Universidad Autónoma de Querétaro
Facultad de Ingeniería

Diseño y construcción de un controlador digital para temperatura basado en FPGA

Tesis
Que como parte de los requisitos
para obtener el título de

Ingeniero
Electromecánico

Presenta
Joel Anaya Sánchez

Asesor:
Dr. Roque Alfredo Osornio Ríos

Agradecimientos

A mi familia, por su apoyo incondicional.

A la Universidad Autónoma de Querétaro, en especial al Dr. Roque por su paciencia y apoyo en la elaboración de este trabajo.

Índice

Índice de Tablas

Índice de Figuras

CAPÍTULO I

INTRODUCCIÓN

El control automático ha desempeñado un papel vital en el avance de la ingeniería y la ciencia. Además de su gran importancia en los sistemas de vehículos espaciales, de guiado de misiles, robóticos y análogos, el control automático se ha convertido en una parte fundamental e integral de los procesos modernos industriales y de fabricación. De igual forma, es esencial en las operaciones industriales como el control de presión, temperatura, humedad, viscosidad y flujo en las industrias de procesos (Ogata, 2004).

Siendo la temperatura una de las variables más importantes para medir y controlar en muchos proyectos científicos y procesos industriales, en esta tesis se desarrolló un sistema de control digital para temperatura, basándose en los fundamentos de la teoría de control clásica y, lo más importante, desarrollar e implementar el controlador en una tarjeta FPGA (*Field Programmable Gate Array*, arreglo de compuertas programables en campo) y periféricos de entrada salida. Esto último involucra también un sólido fundamento en teoría de sistemas digitales y la relativamente nueva aparición de la lógica programable. Para las pruebas de funcionalidad del controlador de temperatura se utiliza una máquina de inyección de plástico que se encuentra dentro de las instalaciones del laboratorio de Ingeniería Electromecánica.

Como elemento de retroalimentación (sensor) se hará uso de un termopar tipo J. La señal de salida del controlador consistirá en un PWM (*Pulse Width Modulation*, modulador de ancho de pulso), y como elemento final de control para las pruebas de funcionalidad, la máquina de inyección de plástico dispone de resistencias térmicas conmutadas por relevadores de estado sólido.

Esta tesis esta estructurada de la siguiente manera: el capítulo uno lleva como título introducción; en él se tratan los antecedentes, objetivos y el planteamiento general del proyecto. En el capítulo dos se efectúa una revisión

literaria de las herramientas teóricas en las que se basa el proyecto, tales como: modelado de sistemas de temperatura, leyes de control, sistemas discretos de control y sintonización de controladores así como los lenguajes de descripción de hardware, dispositivos lógicos programables, moduladores de ancho de pulso y sensores para la implementación del diseño.

En el capítulo tres se presenta la metodología empleada para el diseño e implementación del proyecto, el primer punto es la identificación del sistema de temperatura: la máquina de inyección de plástico antes mencionada. Se presèntan también las metodologías a emplear para la sintonización de las leyes de control (P, PD, PI, PID), y la discretización de estas últimas. Uno de los parámetros más importantes es el diseño digital para la estructura del controlador en donde se contempla el uso de un MAC (multiplicador-acumulador) para efectuar las operaciones matemáticas necesarias por el algoritmo discretizado. Se presentan también el diseño digital del modulador de ancho de pulso (PWM) hacia los relevadores de estado sólido como señal de corrección y etapa de potencia respectivamente, así como también la forma de adquisición de la señal de retroalimentación mediante un convertidor analógico-digital (ADC) y el manejo de una pantalla de cristal líquido (LCD) como elemento de interfaz del dispositivo con el usuario. En el capítulo cuatro se muestran los resultados y análisis obtenidos durante el proyecto que consisten en la identificación del modelo matemático de la planta, selección de la ley de control a emplear y su discretización. Se llevan a cabo las simulaciones de las estructuras digitales del controlador y del PWM mediante la herramienta indicada del software VHDL, la simulación de la interfaz, y una descripción detallada de la construcción del prototipo. Por último se enuncian las conclusiones sobre todo el desarrollo en el diseño y construcción del prototipo así como algunas características que podrían optimizarlo. Se adicionan las referencias citadas y anexos que incluyen los listados de código y diagramas esquemáticos utilizados en la construcción del prototipo.

1.1 Antecedentes

Dentro de la Universidad Autónoma de Querétaro existen diversos trabajos realizados en cuanto a medición y/o control de temperatura y empleo

de FPGA. El trabajo que se constituye como referencia directa con el presente es una tesis de maestría titulada "Sistema de control de temperatura aplicado a máquina de inyección de plástico" (Enriquez, 2006) que consistió en el diseño de todo un controlador de algoritmo PID en un lenguaje de descripción de Hardware (VHDL) y que llegó a la etapa de simulación pero nunca se logró a implementar en un dispositivo programable, en cambio su algoritmo se probó en la máquina de inyección de plástico mediante su desarrollo en software.

Existe otra tesis titulada "Sistema de monitoreo y control de temperatura en el edificio de aulas de posgrado de la facultad de Ingeniería de la UAQ" (Contreras, 2006), el objetivo era controlar la temperatura dentro del recinto teniendo como elemento actuador un equipo de aire acondicionado comercial, este sería controlado mediante el voltaje y frecuencia en las terminales de su motor trifásico, empleando para ello un inversor controlado, sin embargo, el algoritmo de control solo tenia la acción on/off y no se implementaba una ley de control formal (PID y las leyes que de esta se derivan). La interfaz con el usuario y el controlador fueron desarrollados en un lenguaje de programación visual, en este caso Visual Basic. Cabe resaltar el empleo de una tarjeta FPGA comercial Spartan-3, para llevar a cabo las funciones digitales de comunicación de la computadora con los dispositivos externos.

Así mismo, se desarrolló el trabajo "Identificación y control en tiempo real de un horno eléctrico" (Burgos, 2004), en donde se programó un algoritmo PID para controlar la temperatura dentro de un horno eléctrico comercial. El controlador fue programado en un lenguaje de programación visual (Windows CVI). Como elemento de medición se implementó un transductor RTD DIN PT (detector de temperatura resistivo). Un relevador de estado sólido y una resistencia eléctrica formaron la etapa de potencia.

Existe otro trabajo titulado "Medición de temperatura" (Anaya y Leonides, 1998), que consistía en sólo medir la temperatura dentro de un horno y desplegarla en el monitor de una PC. Esto se logró teniendo como elemento sensor un termopar tipo K, un sistema de amplificadores operacionales como acondicionadores de señal y, lo más importante fue el empleo de un sistema

de adquisición de datos llamado PC-Lab Card PLC-812 PG. Para el despliegue en pantalla se utilizó el software comercial Lab View.

A nivel nacional existen proyectos muy relacionados con el presente, por ejemplo, "Diseño y construcción de un controlador de temperatura para incubadora" (San Vicente y Acosta, 2001) del Instituto Tecnológico de Estudios superiores de Monterrey Campus Estado de México, en donde se propuso el diseño de un controlador digital utilizando las acciones de control de todo o nada (On/Off), proporcional (P) y proporcional Integral (PI) para sustituir el controlador analógico comercial con el que contaban las incubadoras. Este controlador fue solicitado por el Instituto Mexicano del Seguro Social, Hospital Gineco-Obstetricia del Centro Médico la Raza, para un programa de rehabilitación de equipos médicos. El algoritmo de control fue programado en un microcontrolador MC68HC11A1, como elemento calefactor es empleada una resistencia de 200 W conmutada por TRIAC y un termistor es empleado para retroalimentar el sistema.

En el ámbito internacional existen diversas propuestas en cuanto controladores digitales se refiere, por ejemplo, resulta interesante otra aplicación biológica como el control en las incubadoras antes mencionado, el cual es "Una aplicación biomédica basada en microcontroladores: control de temperatura en ratas de laboratorio para experimentación quirúrgica" (Hernández et al., 2003), del departamento de Ingeniería telemática y tecnología Electrónica, Universidad Rey Juan Carlos Madrid. La principal necesidad era mantener a ratas de laboratorio vivas durante prácticas quirúrgicas ya que su temperatura corporal desciende al estar anestesiadas. Se utilizó como elemento de retroalimentación un sensor digital LM74. El algoritmo de control fue implementado en un microcontrolador PIC16F876 mediante el lenguaje de programación ensamblador.

Por otro lado, el uso de la tecnología FPGA ha cobrado gran importancia en los últimos años en varias aplicaciones mecatrónicas ejemplo de ello es el trabajo "Instrumentación Climatológica aplicada a granjas de producción animal" (Joshua Mendoza et al., 2007), en donde el propósito era medir todas las variables indispensables para la optimización de una granja

animal, (higiene, enfermedades, crecimiento de los animales), como temperatura, humedad, concentraciones de gas, etc. El sistema de adquisición esta basado en un FPGA A54SX32A de ACTEL, este comandaba convertidores analógico-digitales (ADC) ADS7886, de Texas Instruments, en cada módulo de variable correspondiente. Es de señalarse que a pesar de obtener los datos digitales de cada variable física, no se realiza ningún control para corregir estas últimas.

Otro trabajo desarrollado es el titulado "Ajuste polinomial implementado en FPGA para un termistor Davis-7817" (Soto et al., 2007), donde se propuso el diseño e implementación en Hardware de una aproximación polinomial de la repuesta en un sensor Davis 7817 descrito en VHDL e implementado en el FPGA de la tarjeta comercial Spartan 3.

Existen también aplicaciones a máquinas de control numérico por computadora (CNC), "Aplicación de la lógica reconfigurable al control de fresadoras de alta velocidad" (Osornio et al., 2006) donde los autores aprovechan la lógica digital reconfigurable para diseñar el control de posición de una fresadora de alta velocidad mediante un algoritmo PID y así mejorar la eficiencia de los tiempos de maquinado en ésta.

Otro proyecto de considerable importancia es "Implementación en FPGA de perfiles de aceleración con polinomios de orden superior para la reducción jerk pico en servomotores" (Osornio et al., 2006), donde el "jerk" es analizado considerando su efecto en el desgaste de las partes móviles en máquinas herramienta impulsadas por servomotores. Fueron diseñados perfiles en la aceleración para reducir los picos en servomotores utilizando modelos matemáticos programados sobre FPGA.

Otra aplicación más de la instrumentación con FPGA es el trabajo "Medidor digital de corriente alterna implementado en FPGA" (Avendaño et al., 2007) en donde el principal uso del FPGA era controlar un convertidor analógico digital (ADC 0808) y el despliegue en displays de siete segmentos de la señal analógica de la variable a medir por medio de un transformador de corriente (TC). Se utilizó el FPGA XC3S200 de la tarjeta Spartan 3.

1.2 Objetivos

1.2.1 Objetivos generales

El objetivo principal de este trabajo es diseñar e implementar un controlador digital de algoritmo PID o cualquier otra ley físicamente en Hardware, así como su interfaz de retroalimentación y señal de corrección mediante un FPGA, para su aplicación en proceso térmico.

La funcionalidad del proyecto se podrá confirmar mediante la implementación del Hardware en uno más de los módulos de control de eventos continuos de la máquina de inyección de plástico antes mencionada, específicamente, en el cañón de fundición del plástico para inyección en moldes.

La "programación" en hardware del algoritmo PID se realizará haciendo uso de unas de las herramienta más poderosas en este campo, que son los dispositivos lógicos programables y los lenguajes de descripción de Hardware, específicamente, FPGA y VHDL respectivamente.

1.2.2 Objetivos particulares

- Identificar el sistema de temperatura mediante un método clásico para el conocimiento del modelo.

- Desarrollar la sintonización del controlador mediante el método de respuesta en frecuencia, para la obtención de los coeficientes del controlador.

- Realizar la estructura digital del controlador mediante lógica programable, para su operación independiente a una PC.

- Implementar digitalmente el controlador mediante tecnología FPGA, con la finalidad de tener control en línea.

- Integrar la estructura digital de la interfaz mediante lógica programable, para ser reconfigurable.

- Generar el prototipo de prueba mediante tarjetas y diseños propios en Protel.

Uno de los objetivos particulares más importantes es aplicar los conocimientos adquiridos en esta universidad, especialmente control automático, electrónica digital, analógica, de potencia, así como diseño de sistemas digitales con VHDL.

1.3 Justificación

Lamentablemente en nuestro país, se tiene una enorme dependencia de tecnología industrial extranjera, como en el ramo de control automático de procesos, por ello, el presente trabajo cumple con la responsabilidad que como ingenieros o estudiantes de ingeniería se tiene al recibir educación pública, desarrollar e innovar con diseños propios, dispositivos que cumplan con las exigencias de los cada vez mas complejos procesos industriales.

La medición de temperatura es una de las variables más comunes e importantes que se monitorean en procesos industriales. Casi todos los fenómenos físicos están afectados por ella. La temperatura se utiliza frecuentemente para inferir otras variables de procesos (Creus, 2005). En los procesos de secado, de fusión, de tratamiento térmico, de reacción química, etc., la temperatura es de primordial importancia. La temperatura se ha vuelto una variable crítica en procesos autónomos como los de la industria alimenticia, o en el control de clima artificial en el interior de edificios que se encuentran en lugares de climas extremos donde el desempeño del personal laborando en sus interiores depende en gran medida de la temperatura corporal. Por tanto, surge la necesidad de medir y controlar la temperatura de los fenómenos físicos de la que dependen directa o indirectamente los procesos industriales.

Particularmente, como ya se mencionó en los objetivos generales del proyecto, se tiene la necesidad de controlar de la manera más eficientemente posible (en cuanto a costo y control en línea) la temperatura de fundición de plástico de la máquina de inyección de plástico en el laboratorio de Ingeniería Electromecánica, sustituyendo el controlador comercial del que dispone la máquina. Sin embargo, el diseño del controlador esta pensado para su

aplicación en prácticamente cualquier proceso térmico pues solo se requeriría de un cambio en los sensores, actuadores y programación interna del dispositivo, eliminando así cualquier modificación física del circuito electrónico.

Actualmente, se tiene una tendencia hacia controladores digitales y no análogos, estos últimos presentan grandes desventajas como las no linealidades, saturación y ruido debido a la necesidad de amplificadores operacionales para su implementación. Estas desventajas pueden ser superadas gracias al empleo de los circuitos integrados reconfigurables ya que si se desea modificar el controlador no es necesario rediseñar el circuito en la placa impresa físicamente, siendo necesario solamente reprogramar la lógica de control mediante software. Aunque los elementos mas complejos con los que se pretende construir el dispositivo (tales como todos los semiconductores) no son diseñados ni manufacturados en el país, es posible disminuir el costo en gran medida si se puede diseñar el dispositivo aprovechando la principal ventaja de la lógica programable que es precisamente poder reconfigurar desde el exterior su funcionamiento mediante diversas técnicas de programación. Así, se tiene una cierta independencia tecnológica en cuanto al uso de tecnología extranjera. Por lo mencionado anteriormente, en este proyecto se decidió el empleo de un dispositivo lógico programable, en este caso, un FPGA por las ventajas que este ofrece respecto a otro tipo de procesadores. Dichas ventajas son: alta capacidad de procesamiento, reprogramabilidad, reconfigurabilidad, manejan lenguajes portables y todo a bajo costo. Es importante recalcar la portabilidad del lenguaje de descripción en hardware (HDL) a utilizar, es decir, VHDL que es un lenguaje totalmente estandarizado por el Instituto de Ingenieros Eléctricos y Electrónicos (IEEE por sus siglas en Inglés), y es respaldado por la mayor parte de los organismos que ofrecen hardware digital. Esta es una de sus mayores ventajas en comparación con otros lenguajes, ya que el diseñador tiene que preocuparse poco por la tecnología de implementación, centrándose en la funcionalidad del circuito deseado.

En algunas obras mencionadas en los antecedentes, los algoritmos de control y/o medición fueron programados en lenguajes de programación,

empleando para ello ordenadores personales haciendo que el proceso controlado dependa enteramente de estos. Pero los ordenadores personales comúnmente fabricados para su uso en oficinas, escuelas o laboratorios, no están preparados para su uso en la industria donde se presentan polvo, vibraciones mecánicas, temperaturas extremas, etc., además de que no realizan un control en tiempo real necesario para algunos procesos industriales. Por ello, la necesidad de contar con el controlador en hardware, donde sólo es necesario un despliegue visual de la temperatura de referencia, temperatura alcanzada y botones para ajustar la temperatura requerida por el proceso. En la construcción del prototipo se contempla la similitud de la interfaz hacia el usuario, con la mayoría de los controladores comerciales de temperatura, es decir, una pequeña pantalla donde se visualice la temperatura en tiempo real y la referencia, así como botones para modificar esta última.

1.4 Planteamiento general

La figura 1.1 muestra un diagrama general del sistema construido, el bloque principal es el FPGA donde se encuentra el algoritmo de la ley de control a emplear, el modulador de ancho de pulso, la comunicación con el display LCD y el convertidor analógico digital. Enseguida de este módulo se ubica el relevador de estado sólido que será conmutado por la salida del PWM, es decir, la señal de control, variando así la potencia eléctrica del actuador, en este caso una resistencia de calor para fundición de plástico, la temperatura de ésta última será medida utilizando como elemento transductor un termopar tipo J, es necesario, antes de convertir esta señal analógica en digital por medio del ADC, un acondicionador de señal para obtener así un voltaje lineal de 0 a 5 V, cerrando de esta forma el lazo de control.

Los módulos exclusivos del prototipo por construir son la referencia y la interfaz visual, el FPGA, el ADC y el acondicionador de señal. De esta forma, cuando se requiera conectar el dispositivo al sistema será necesario conectar sólo el termopar y el relevador de estado sólido.

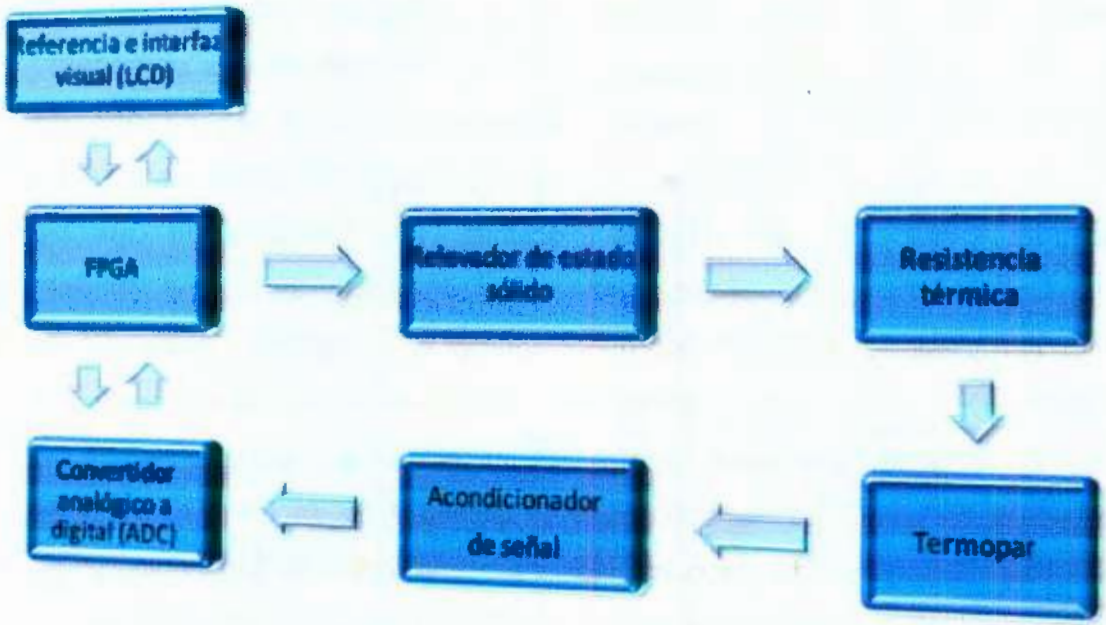


Figura 1.1 Diagrama general del sistema.

CAPÍTULO II

REVISIÓN DE LITERATURA

2.1 Estado del arte

En los inicios de la era industrial, el control de los procesos se llevó a cabo mediante tanteos basados en la intuición y en la experiencia acumulada por el operario. Un caso típico fue el control de acabado de un producto en un horno. El operario era realmente el "instrumento de control" que juzgaba la marcha del proceso por el color de la llama, por el tipo de humo, el tiempo transcurrido y el aspecto del producto y decidía así el momento de retirar la pieza; en esta decisión influía muchas veces la suerte, de tal modo que no siempre la pieza se retiraba en las mejores condiciones de fabricación. Más tarde, el mercado exigió mejor calidad en las piezas fabricadas lo que condujo al desarrollo de teorías para explicar el funcionamiento del proceso, de las que derivaron estudios analíticos que a su vez permitieron realizar el control de la mayor parte de las variables de interés en los procesos (Creus, 2005).

El primer trabajo significativo en control automático fue el regulador de velocidad centrífugo de James Watt para el control de velocidad de una máquina de vapor, en el siglo XVIII. Minorsky, Hazen y Nyquist, entre muchos otros, aportaron trabajos importantes en las etapas iniciales del desarrollo de la teoría de control. En 1922, Minorsky trabajó en controladores automáticos para el guiado de embarcaciones, y mostró que la estabilidad puede determinarse a partir de las ecuaciones diferenciales que describen el sistema. En 1932, Nyquist diseñó un procedimiento relativamente simple para determinar la estabilidad de sistemas en lazo cerrado, a partir de la respuesta en lazo abierto a entradas sinusoidales en estado estacionario. En 1934, Hazen, quien introdujo el término servomecanismos para los sistemas de control de posición, analizó el diseño de los servomecanismos con rele, capaces de seguir con precisión una entrada cambiante (Ogata, 2004).

Durante la década de los cuarenta, los métodos de la respuesta en frecuencia (especialmente los diagramas de bode) hicieron posible que los ingenieros diseñaran sistemas de control lineales en lazo cerrado que cumplieran los requisitos de comportamiento. A finales de los años cuarenta y principios de los cincuenta, se desarrolló por completo el método del lugar de las raíces propuesto por Evans (Ogata, 2004).

La ingeniería de control clásica, que dominó hasta aproximadamente 1960, maneja procesos y señales análogas; es decir, características dinámicas y transitorias que se estudian en tiempo continuo. La descripción matemática se basa en ecuaciones diferenciales, con la transformada de Laplace y de ahí en funciones de transferencia como base primaria para el análisis.

Al desarrollarse las técnicas digitales en los años cincuenta se encontró la necesidad de métodos para un sistema de control en donde los eventos sucedían en tiempos discretos. Una teoría para sistemas muestreados (*sampled systems*) llegó a finales de los años cincuenta, en donde todavía había mucho en común con métodos desarrollados anteriormente para el dimensionamiento de sistemas de control.

Alrededor de los años setenta el desarrollo de componentes digitales no era caro ni con faltas de aplicación para los cálculos numéricos. En su lugar se podía pensar en usar las computadoras directamente en el circuito de control para que en tiempo real se pudieran calcular valores adecuados de las señales de regulación basados en la información dada por la medición. Al mismo tiempo el desarrollo dio pie a que con la ayuda y flexibilidad de las computadoras se pudieran realizar mejores métodos de control (Navarro, 2004).

Ya sea que se use la transformada de Laplace o las ecuaciones diferenciales se encuentra mucha similitud en las herramientas de análisis entre los sistemas de tiempo continuo y los de tiempo discreto. Muchos de los conceptos y métodos de la ingeniería de control en tiempo continuo se pueden transferir a los sistemas de tiempo discreto, tales como la función de transferencia, reducción de diagramas de bloques, estabilidad, respuesta

escalón, función en respuesta en frecuencia, entre otras. Una parte de las soluciones modernas no necesariamente necesitan una computadora, sino que puede usarse para mejorar el diseño convencional análogo.

El rápido crecimiento que tiene la teoría de control continúa a la fecha. Los nuevos métodos requieren modelos relevantes de los procesos y grandes personalidades han hecho interesantes trabajos en esta rama, la cual se llama identificación de procesos, entre ellos cabe mencionar a Karl-Johan Åström, Lennart Ljung y Torsten Söderström.

La combinación de control con identificación de procesos basadas en la información que se obtiene ha dado lugar a nuevos tipos de reguladores, los adaptivos, los cuales ajustan sus características a las dinámicas variables de los procesos. Al mismo tiempo se ha apostado por el desarrollo de reguladores constantes, aquellos que toleran grandes cambios en el proceso sin que por ello la estabilidad y otras prestaciones sean afectadas, a esto se le llama *control robusto*.

Se puede pensar que con todo esto la elaboración y estandarización de los métodos se han logrado, como en el caso de los controladores PID. Aún cuando este no es el caso, el gran avance tecnológico y las demandas por parte de la población al mejoramiento del medio ambiente están logrando que métodos y soluciones con ayuda de los microprocesadores sean un factor importante en la Ingeniería de control (Navarro, 2004).

2.2 Sistemas comerciales para control de temperatura

Existe una muy amplia gama en cuanto a controladores comerciales se refiere, es fácil recurrir a catálogos impresos o en Internet, para su adquisición. A continuación se muestran algunos controladores digitales comerciales. Esta información fue adquirida de las guías de usuario que se requieren para su uso e instalación de la página de internet www.omega.com/prodinfo/ TemperatureControllers.html

La marca estadounidense OMEGA, proveedora de equipo electrónico para la industria, cuenta con controladores digitales para muchos tipos de variables, entre estas, temperatura. Están clasificados en series, una de ellas es la serie CNI8C, este modelo se muestra en la figura 2.1.

Figura 2.1 Controlador comercial de OMEGA serie CNI8C.

Algunas de sus características más importantes son las siguientes:

- **Precisión:** ±0.5 °C; 0.03% en lectura.
- **Resolución:** 1°/0.1° 10μV proceso
- **Estabilidad en temperatura:** RTD: 0.04 °C/°C, Termopar @ 25 °C (77°F): 0.05 °C/°C.
- **Modo de control:** Control de tiempo y amplitud proporcional; selección manual o auto PID, proporcional, proporcional integral, proporcional derivativo y on/off.
- **Autosintonizable:** Operación iniciada desde el panel frontal.

Se encuentran disponibles controladores más complejos de la misma marca, por ejemplo la serie i mostrada en la figura 2.2.

Este dispositivo puede ser configurado como controlador o como un acondicionador de señal. Cada unidad de la serie i permite al usuario seleccionar el tipo de entrada, desde 10 tipos de termopares (J, K, T, E, R, S,

B, C, N), RTDs (100, 500 o 1000 Ω), voltaje DC o corriente CD. El control puede ser ajustado como on/off o PID, puede ser optimizado con una opción de autosintonizado. Las características estándar incluyen tres salidas con opción a salidas relé, SCR, o pulso dc y salida de voltaje o corriente aislado. Opciones que incluyen comunicación serial RS232 programable o excitación RS485. Los precios de esta serie comienzan en los 795 dólares.

Figura 2.2 Controlador comercial OMEGA serie i.

Otra marca comercial disponible es WATLOW, que ofrece también controladores digitales para temperatura, por ejemplo, el mostrado en la figura 2.3 es un controlador también autosintonizable que permite los modos de control on/off, P, PI o PID. El fabricante asegura su funcionamiento en el ambiente industrial. También existen dispositivos de esta marca para comunicación remota como el mostrado en la figura 2.4.

Figura 2.3 Controlador comercial WATLOW.

Figura 2.4 Controlador comercial WATLOW con comunicación remota.

2.3 Modelado de sistemas de temperatura

Dado que el principal objetivo de un sistema de control es manipular una o más de las variables físicas de un sistema, es necesario conocer el comportamiento "natural" de tal sistema mediante un modelo matemático, ya que es la forma más viable para el análisis y diseño del sistema de control.

El modelo más generalizado es la llamada "función de transferencia" que no es más que una representación en el plano s (transformada de Laplace) de un sistema físico que puede ser descrito por ecuaciones diferenciales ordinarias de coeficientes constantes. En algunos casos se puede obtener la función de transferencia de "principios básicos", esto es, aplicar las leyes apropiadas de la física, química o electricidad al sistema físico empezando por obtener las ecuaciones diferenciales que describan al sistema. La transformada de Laplace puede ser usada para obtener la función de transferencia.

En otros casos, puede ser difícil obtener la función de transferencia fundamental a partir de los "principios básicos". Éste a menudo es el caso si el sistema es complejo. Por ejemplo, supóngase que deseamos encontrar el momento de inercia *I* de un objeto de forma irregular, por ejemplo, del transbordador espacial con una carga útil a bordo listo para ser lanzado al espacio. El cálculo del momento de inercia con los principios básicos podría ser complicado. Por consiguiente, a menudo tenemos que recurrir a métodos

experimentales más genéricos para determinar los parámetros de funciones de transferencia (Dorsey, 2005).

Por lo tanto, la forma más común de modelar un sistema de temperatura, es someterlo a entradas de prueba (comúnmente la entrada escalón), para así obtener la "constante de tiempo" que es una variable característica de los sistemas de primer orden, incluyendo a los de temperatura, obteniendo así, la función de transferencia de un sistema de primer orden

$$\frac{C(s)}{R(s)} = \frac{1}{Ts+1} \qquad 1$$

dónde $C(s)$ es la salida del sistema, $R(s)$ la entrada y T la constante de tiempo. Al evaluar $R(s) = \frac{1}{s}$ (señal de prueba escalón) y resolver para $C(s)$, se obtiene

$$C(s) = \frac{1}{Ts+1}\frac{1}{s} \qquad 2$$

al desarrollar $C(s)$ en fracciones simples se obtiene

$$C(s) = \frac{1}{s} - \frac{T}{Ts+1} = \frac{1}{s} - \frac{1}{s+(1/T)} \qquad 3$$

si se toma la transformada inversa de Laplace de la ecuación 3 se obtiene

$$c(t) = 1 - e^{-t/T}, \text{para } t \geq 0 \qquad 4$$

La curva de respuesta exponencial $c(t)$ obtenida mediante la ecuación 4 aparece en la figura 2.5 (Ogata, 2003). Por lo tanto, es posible modelar un sistema de temperatura al someter el sistema a una entrada escalón y evaluar el 63.2% de la entrada escalón para entonces obtener la constante de tiempo y finalmente obtener la función de transferencia.

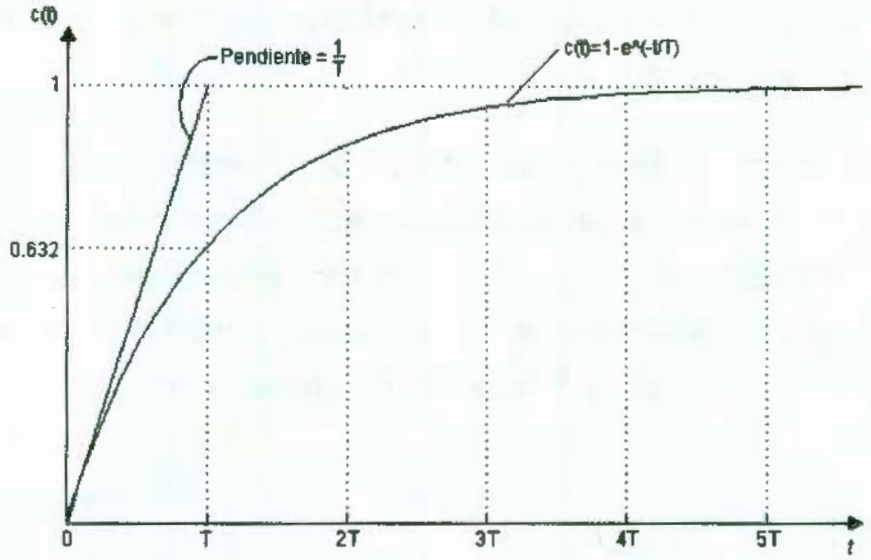


Figura 2.5 Curva de respuesta exponencial.

2.4 Leyes de control

Los controladores industriales se clasifican, de acuerdo con sus acciones de control, como:

- Controladores proporcionales (P)
- Controladores integrales (I)
- Controladores proporcionales-integrales (PI)
- Controladores proporcionales-derivativos (PD)
- Controladores proporcionales-integrales-derivativos (PID)

2.4.1 Ley de control Proporcional

Para un controlador con acción de control proporcional, la relación entre la salida del controlador *y(t)* y la señal de error *x(t)* es:

$$y(t) = K_p x(t) \qquad 5$$

o bien, en cantidades mostradas por el método de Laplace,

$$\frac{Y(s)}{X(s)} = K_p \qquad 6$$

dónde K_p se considera la ganancia proporcional.

Cualquiera que sea el mecanismo real y la forma de la potencia de operación, el controlador proporcional es, en esencia, un amplificador con una ganancia ajustable.

2.4.2 Ley de control Integral

En un controlador con acción de control integral, el valor de la salida del controlador *u(t)* se cambia a una razón proporcional a la señal de error *e(t)*. Es decir,

$$\frac{dy(t)}{dt} = K_i x(t) \qquad 7$$

o bien

$$y(t) = K_i \int_0^t x(t)\, dt \qquad 8$$

dónde K_i es una constante ajustable. La función de transferencia del controlador integral es

$$\frac{Y(s)}{X(s)} = \frac{K_i}{s} \qquad 9$$

2.4.3 Ley de control Proporcional-Integral

La acción de control de un controlador proporcional-integral (PI) se define mediante

$$y(t) = K_p x(t) + \frac{K_p}{T_i} \int_0^t x(t)\, dt \qquad 10$$

o la función de transferencia del controlador es

$$\frac{Y(s)}{X(s)} = K_p \left(1 + \frac{1}{T_i s}\right) \qquad 11$$

dónde T_i se denomina tiempo integral.

2.4.4 Ley de control Proporcional-Derivativa

La acción de control de un controlador proporcional-derivativa (PD) se define mediante

$$y(t) = K_p x(t) + K_p T_d \frac{dx(t)}{dt} \qquad 12$$

y la función de transferencia es

$$\frac{Y(s)}{X(s)} = K_p(1 + T_d s) \qquad 13$$

donde T_d es el tiempo derivativo.

2.4.5 Ley de control Proporcional-Integral-Derivativa

La combinación de la acción de control proporcional, la acción de control integral y la acción de control derivativa se denomina acción de control proporcional-integral-derivativa. Esta acción combinada tiene las ventajas de cada una de las tres acciones de control individuales. La ecuación de un controlador con esta acción combinada esta dada por

$$y(t) = K_p x(t) + \frac{K_p}{T_i} \int_0^t x(t)\, dt + K_p T_d \frac{dx(t)}{dt} \qquad 14$$

o la función de transferencia es

$$\frac{Y(s)}{X(s)} = K_p \left(1 + \frac{1}{T_i s} + T_d s\right) \qquad 15$$

dónde $K_p$es la ganancia proporcional, T_i es el tiempo integral y T_d es el tiempo derivativo (Ogata, 2003).

2.5 Sistemas discretos de control

El diseño de sistemas de control en tiempo discreto es similar en principio al diseño de sistemas de control en tiempo continuo. El objetivo del diseño es básicamente, determinar el controlador para que el sistema tenga un desempeño de acuerdo a las especificaciones. De hecho, en la mayoría de las situaciones, el proceso controlado es el mismo, excepto que en sistemas en tiempo discreto el controlador esta diseñado para procesar datos digitales o muestreados.

En años recientes el análisis y diseño de sistemas de control de datos discretos y digitales ha experimentado avances muy importantes. Estos sistemas han ganado mucha popularidad e importancia en la industria debido, en parte, a los progresos realizados en computadoras digitales de control, en microprocesadores (MP) y procesadores digitales de señal (DSP).

Los sistemas de control de datos discretos y digitales difieren de los sistemas continuos, o analógicos, en que las señales en una o más partes de ellos se encuentran en forma de trenes de pulsos o códigos numéricos. Los términos sistemas de control de datos muestreados, sistemas de control de datos discretos y sistemas de control digital se emplean de manera vaga e intercambiable en la literatura de sistemas de control. En sentido estricto, los datos muestreados son señales cuya amplitud esta modulada por pulsos y que se obtienen mediante el muestreo de señales analógicas. Con frecuencia, la señal con amplitud modulada por pulsos esta presente en la forma de trenes de pulsos, donde la información es transmitida por las amplitudes de estos. En general, los datos digitales son las señales generadas por computadoras o transductores digitales; lo usual es que estos datos se encuentren codificados de alguna manera en formato digital. Por otra parte, los métodos de análisis y diseño son esencialmente los mismos, sin importar que el sistema contenga datos muestreados o codificados de manera digital (Kuo, 1996).

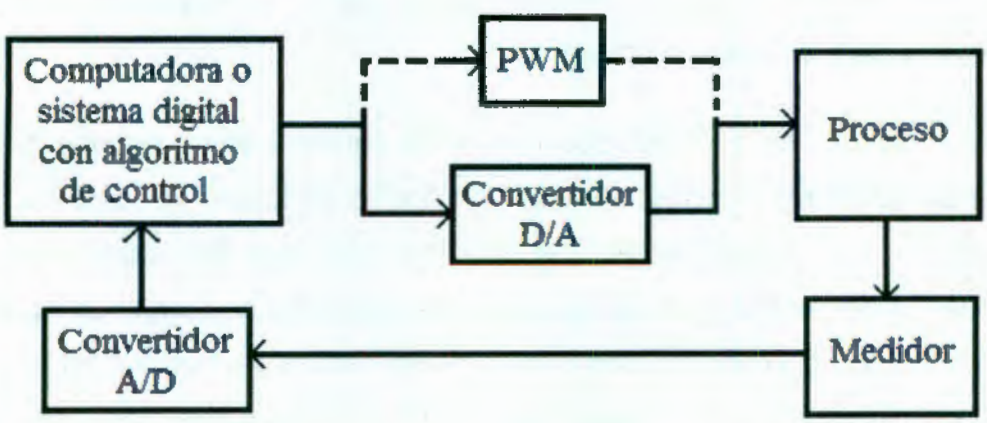


Figura 2.6 Descripción esquemática de un sistema de control en tiempo discreto.

En la figura 2.6 se muestra la configuración básica de un sistema de control de tiempo discreto. Los procesos que se van a regular son del mismo

tipo que aquellos del caso análogo. La diferencia principal entre los sistemas análogos y digitales es el equipo necesario para la regulación.

Para la construcción de un sistema de regulación digital normalmente se necesitan los siguientes componentes:

Convertidor analógico – digital (A/D). Un convertidor analógico – digital, también conocido como codificador es un dispositivo que convierte una señal analógica en una señal digital, usualmente una señal codificada numéricamente. Dicho convertidor se necesita como una interfaz entre un componente analógico y uno digital. Con frecuencia un circuito de muestreo y retención es una parte integral de un convertidor A/D disponible comercialmente. La conversión de una señal analógica en la señal digital correspondiente (número binario) es una aproximación, ya que la señal analógica puede adoptar un número infinito de valores, mientras que la variedad de números diferentes que se pueden formar mediante un conjunto finito de dígitos esta limitada. Este proceso de aproximación se denomina *cuantificación* (Ogata 1996). La exactitud del convertidor depende de la cantidad de cifras binarias (número de bits) que se use para representar cierta medición. Normalmente se habla de convertidores de 8 bits, 12 bits, etc. Un convertidor de 8 bits corresponde a un campo de trabajo del medidor de 2^8 = 256 intervalos. Su exactitud es por lo tanto, 1/256 = 0.4% para un voltaje de referencia de 1 V (Navarro, 2004).

Computadora digital. El componente principal en un sistema de control es la computadora o sistema digital, en donde se hacen los cálculos para la regulación. Las computadoras que son especialmente diseñadas para usarse en cuartos de control para la regulación de procesos se llaman computadoras de procesos. Lo importante es que la computadora se puede comunicar con sus alrededores vía los convertidores A/D y D/A, y que tiene consigo los programas adecuados para el cálculo de las señales de regulación (Navarro, 2004).

Convertidores Digital – Analógico (D/A). Se usa para convertir las señales binarias de la salida de la computadora en una señal análoga. A la salida del convertidor existe normalmente un circuito que mantiene la señal de regulación del último valor calculado hasta que llega un nuevo valor, es decir, mantiene el valor entre los intervalos de tiempo, esto hace que la señal sea de pulsos constantes (cambios escalón entre cada intervalo) (Navarro, 2004).

Modulador de ancho de pulso. De manera alterna al empleo de DACs, la computadora o sistema digital puede proveer un tren de pulsos que regulen la potencia eléctrica entregada al actuador siempre y cuando éste lo permita. En la sección 2.8 se explica con mayor detalle el funcionamiento de este bloque.

Actuador. Es en principio del mismo tipo que se usa en un sistema de control análogo. Aun cuando a veces viene con el convertidor D/A integrado, lo cual significa que pueden regular directamente en forma binaria (Navarro, 2004).

Medidor o transductor. Un transductor es un dispositivo que convierte una señal de entrada en una señal de naturaleza diferente a la de entrada, tal como los dispositivos que convierten una señal de presión en una salida de voltaje. En general, la señal de salida depende de la historia de la entrada. Los transductores se pueden clasificar como transductores analógicos, transductores de datos muestreados o transductores digitales. Un transductor analógico es aquel en el que las señales de entrada y salida son funciones continuas del tiempo. Las magnitudes de estas señales pueden tomar cualquier valor dentro de las limitaciones físicas del sistema. Un transductor de datos muestreados es aquel en el que las señales de entrada y salida se presentan en valores discretos de tiempo (normalmente periódicos), pero las magnitudes de las señales, como en el caso de los transductores analógicos, no están cuantificadas. Un transductor digital es aquel en el que las señales de entrada y salida se presentan solo en valores discretos de tiempo y las magnitudes de las señales están cuantificadas (esto es, solamente pueden adoptar ciertos valores discretos) (Ogata, 1996).

2.5.1 Transformada Z

En general, los métodos para el análisis y diseño de sistemas de control pueden clasificarse en *convencionales* o *modernos*. Los métodos convencionales, o clásicos, se caracterizan por el empleo de técnicas basadas en transformadas y funciones de transferencia, mientras que la teoría de control moderna se basa en el modelado de sistemas con variables y ecuaciones de estado. La transformada de Laplace es la herramienta básica en el análisis y diseño convencional de sistemas de control de los datos continuos. En principio, esta transformada también puede utilizarse para modelar sistemas de control digital. Sin embargo, las expresiones correspondientes a las transformadas de Laplace de los sistemas donde aparecen señales digitales o muestreadas, contienen términos exponenciales de la forma e^{Ts}. Esto dificulta inevitablemente el manejo de las transformadas. Desde un punto de vista heurístico, lo anterior puede considerarse como motivación para introducir la transformada z.

En la práctica, es preferible la aproximación dada por la transformación bilineal para la simulación digital de sistemas de control, debido a que mapea el eje $j\omega$ del plano s sobre el circulo unitario del plano z. Esto significa que, dada cualquier función de transferencia racional $G(s)$, los polos y ceros de los semiplanos izquierdo y derecho del plano s se mapean sobre los correspondientes polos y ceros dentro y fuera del plano z, respectivamente, para $G(z)$, cuando se aplica la transformación bilineal dada por la ecuación 16 (Kuo, 1996).

$$s = \frac{2}{T}\frac{z-1}{z+1} \qquad 16$$

2.5.2 Ecuaciones en diferencias

En un sistema de procesamiento de señales en tiempo discreto existe la entrada de una secuencia de pulsos y una salida de pulsos. En cierto instante la salida la calcula el sistema como resultado de procesar el pulso presente y los pulsos previos en la entrada y quizá los pulsos previos en la salida. Por ejemplo, en un instante particular se podría tener una entrada de pulsos $x[k]$ de una secuencia. El programa que se usa con el microprocesador

podría leer este valor y adicionarlo al valor de salida previo $y[k-1]$ para dar la salida requerida, $y[k]$. Esta operación se puede representar mediante la ecuación

$$y[k] = y[k-1] + x[k] \qquad 17$$

Dicha ecuación se denomina *ecuación en diferencias* y proporciona la relación entre la salida y la entrada para un sistema en tiempo discreto; es comparable a la ecuación diferencial que se usa con sistemas en tiempo continuo para relacionar la entrada y la salida (Bolton, 2001).

2.6 Sintonización de controladores

En la sección 2.4 se mostraron los diferentes tipos o leyes de control, los mas conocidos y popularmente usados son los del tipo proporcional – integral – derivativo (PID), también se encuentran filtros de adelanto o atraso de fase, los reguladores Otto – Smith, los basados en lógica difusa (*fuzzy logic*), etc.

Una pregunta que se debe hacer es: ¿Cuál ley se debe usar?, la respuesta a esto depende de su aplicación y los requisitos de regulación establecidos para el sistema. El 80 % de los procesos controlados hoy en día tiene una implementación de reguladores PID. Es por ello que en primera instancia hay que dedicarse a los controladores PID (Navarro, 2004).

El uso del controlador proporcional sólo requiere la elección de una variable: la ganancia proporcional, k_p, para que el sistema de control tenga el comportamiento dinámico requerido. El uso de un controlador PI requiere la selección de dos variables: la ganancia proporcional k_p y la ganancia integral, k_i, Con un controlador PID se deben seleccionar tres variables: la ganancia proporcional k_p, la ganancia integral, k_i, y la ganancia derivativa, k_d. La selección de estas variables permite localizar los polos y ceros que introduce el controlador a ser determinados y, por lo tanto, afectan la estabilidad del sistema de control.

Para describir el proceso de selección de los mejores valores para el controlador se usa el término *sintonización*, existen varios métodos para lograrlo, a continuación se muestran algunos.

2.6.1 Métodos de Ziegler y Nichols

Ambos métodos se basan en experimentación y análisis, estos son "recetas de cocina" útiles que se usan con mucha frecuencia. El primer método a menudo se denomina *método de la curva de reacción del proceso.* El procedimiento con este método consiste en abrir el lazo de control de modo que no se presentan acciones de control. En general, la ruptura del lazo se hace entre el controlador y la unidad de corrección. Se aplica, entonces, una señal de prueba a la unidad de corrección y se determina la repuesta de la variable de proceso medida, es decir, la señal de error. La señal de prueba deberá ser tan pequeña como sea posible. La figura 2.7 muestra la forma de la señal de prueba y una respuesta típica. La gráfica de la señal medida se gráfica contra el tiempo y se conoce como la *curva de reacción del proceso.*

La señal de prueba, *P*, se expresa como el porcentaje de cambio en la unidad de corrección. La variable medida se expresa como el porcentaje del rango a escala completa. Para dar el máximo gradiente de la gráfica se traza una tangente. Para la figura 2.7 el máximo gradiente *R* es *M / T*. El tiempo entre la aplicación de la señal de prueba y cuando esta tangente interseca el eje de tiempo de la gráfica se denomina atraso *L*. La tabla 2.1 proporciona los criterios recomendados por Ziegler y Nichols para los valores del controlador con base en los valores de *P, R* y *L*.

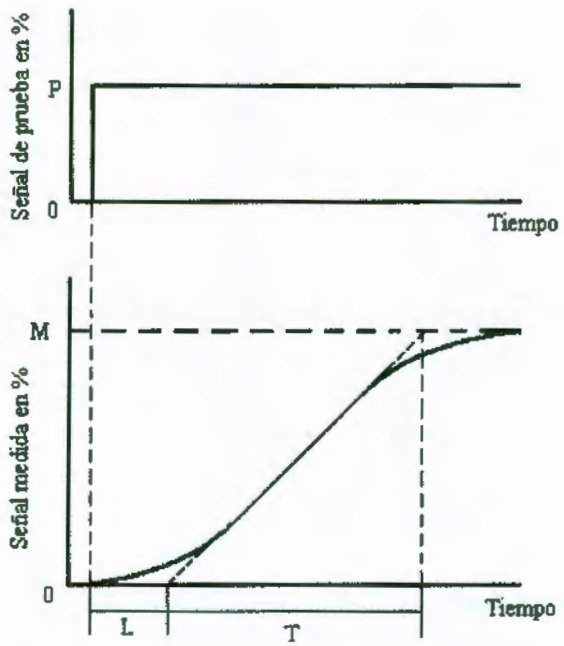


Figura 2.7 Curva de reacción de proceso.

Tabla 2.1 Criterios de Ziegler y Nichols para la curva de reacción del proceso

Ley de control	k_p	k_i	k_d
Proporcional	P/RL		
Proporcional + integral	0.9P/RL	$1/3.33L$	
Proporcional + integral + derivativo	1.2P/RL	½L	$0.5L$

El otro método se conoce como el *método de la última ganancia.* Primero, las acciones integral y derivativa se reducen a sus valores mínimos. La constante proporcional, k_p, se fija en un valor bajo y, entonces, se incremente en forma gradual. Esto es lo mismo que decir que la banda proporcional se hace más angosta de manera gradual. Mientras esto sucede, al sistema se le aplican pequeñas perturbaciones. El proceso continua hasta que se presentan oscilaciones. Se anota el valor crítico de la banda proporcional, k_{pc}, en la que se presentan las oscilaciones, así como el tiempo, T_c, de estas. La tabla 2.2 muestra los criterios de Ziegler y Nichols sobre como se relacionan los valores de k_{pc} y T_c. Para establecer los valores del controlador. La banda proporcional critica es $100/k_{pc}$ (Bolton, 2001).

Tabla 2.2 Criterios de Ziegler y Nichols para la última ganancia

Ley de control	k_p	k_i	k_d
Proporcional	$0.5k_{pc}$		
Proporcional + integral	$0.45k_{pc}$	$1.2T_c$	
Proporcional + integral + derivativo	$0.6k_{pc}$	$2.0T_c$	$T_c/8$

2.6.2 Método del lugar geométrico de las raíces (LGR)

Este método consiste en el trazado y conocimiento del diseñador del LGR y los pasos para su trazado son los siguientes:

1. Dibujar el LGR de la planta.
2. En base a los parámetros de diseño, seleccionar la acción de control que permita cumplirlos o aproximarlos.
3. Proponer la ubicación de los polos y/o ceros del controlador seleccionado en el punto anterior. Quedando todo en función de un parámetro del controlador.
4. Trazar el lugar geométrico de la planta mas el controlador y en base a su descripción (pasos del método), seleccionar el parámetro en el que finalmente quedará trabajando el controlador. En función de este, se determinan el resto de las ganancias del controlador (Osornio, 2007).

2.6.3 Método de respuesta en frecuencia

Este método se basa en parámetros importantes del sistema: la estabilidad, la rapidez de respuesta y su dinámica. La estabilidad indica la repuesta natural del sistema. El grado de estabilidad es casi siempre expresado en margen de fase, entre más grande mejor será la estabilidad.

La velocidad de respuesta indica que tanto le toma la ejecución de pequeñas correcciones que no causen saturación. En la mayoría de los casos, la respuesta del sistema se caracteriza por el parámetro de frecuencia de cruce ω_c el cual esta directamente relacionado con el tiempo de respuesta τ. Entonces existen dos parámetros de diseño en la sintonización de controladores: el margen de fase y la frecuencia de cruce.

a) El margen de fase es estable, cuando es positivo e inestable cuando es negativo. En estudios de análisis, se ha demostrado que valores de margen de fase en el intervalo entre 30° y 45° indican una respuesta amortiguada; valores de fase mas pequeños indican respuesta de bajo amortiguamiento.

b) En cuanto a la frecuencia, indica la velocidad de respuesta, un valor mas alto de ω_c indican respuestas mas rápidas. En general, la respuesta de tiempo τ, se aproxima mediante

$$\tau = \frac{1}{\omega_c} \qquad 18$$

Procedimiento para la sintonización

1. Se obtien la función $L(s) = G(s)H(s)$ función de transferencia en lazo abierto. Se reemplaza "s" por "$j\omega$" función sinusoidal, incluyendo el controlador.

2. Se propone o fija la frecuencia de cruce ω_c en base a los requerimientos de respuesta del sistema.

3. Se determina el margen de fase deseado para el sistema.

4. Basado en el margen de fase, se encuentra la fase para el sistema de $L(j\omega)$ y se obtiene de ahí uno de los parámetros de sintonización usando la ω_c y mediante las siguientes ecuaciones:

$$\varphi = arg|L(j\omega_c)| \qquad 19$$

$$M\varphi = 180 + \varphi \qquad 20$$

5. Como la frecuencia de cruce ω_c es aquella a la cual la magnitud de $L(j\omega) = 1$, se utiliza esta última ecuación sustituyendo ω_c para encontrar el parámetro del controlador.

Como máximo se podrán encontrar dos parámetros. El resto, de ser necesarios, pueden ser propuestos por el diseñador en base a "algún criterio". (Osornio, 2007).

2.7 VHDL Y FPGA

2.7.1 La lógica programable y los lenguajes de descripción en Hardware (HDL)

Como consecuencia de la creciente necesidad de integrar un mayor número de dispositivos en un solo circuito integrado, se desarrollaron nuevas herramientas de diseño que auxilian al ingeniero a integrar sistemas de mayor complejidad. Esto permitió que en la década de los cincuentas aparecieran los lenguajes de descripción en hardware (HDL) como una opción de diseño para el desarrollo de sistemas electrónicos elaborados. Estos lenguajes alcanzaron mayor desarrollo durante los años setenta, lapso en que se desarrollaron varios de ellos como IDL de IBM, TI-HDL de Texas Instruments, ZEUS de General Electric, etc., todos orientados al área industrial, así como los lenguajes en el ámbito universitario (AHPL, DDL, CDL, ISPS, etc.). Los primeros no estaban disponibles fuera de la empresa que los manejaba, mientras que los segundos carecían de soporte y mantenimiento adecuados que permitieran su utilización industrial. El desarrollo continuó y en la década de los ochenta surgieron lenguajes como VHDL, Verilog, ABEL, AHDL, etc., considerados lenguajes de descripción en hardware por que permitieron abordar un problema lógico a nivel funcional (descripción de un problema conociendo sólo las entradas y salidas), lo cual facilita la evaluación de soluciones alternativas antes de iniciar un diseño detallado (Maxinez y Alcalá, 2007).

Una de las principales características de estos lenguajes radica en su capacidad para describir en distintos niveles de abstracción (funcional, transferencia de registros RT y lógico o nivel de compuertas) en cierto diseño. Los niveles de abstracción se emplean para clasificar modelos HDL según el grado de detalle y precisión de sus descripciones.

Los niveles de abstracción descritos desde el punto de vista de simulación y síntesis del circuito pueden definirse como sigue:

- **Algorítmico:** se refiere a la relación funcional entre las entradas y salidas del circuito o sistema, sin hacer referencia a la realización del sistema.

- **Transferencia de registros (RT):** Consiste en la partición del sistema en bloques funcionales sin considerar a detalle la realización final de cada bloque.

- **Lógico o de compuertas:** el circuito se expresa en términos de ecuaciones lógicas o de compuertas (Maxinez y Alcalá, 2007).

2.7.2 Lenguaje de descripción de Hardware VHDL

En la década de 1980, los rápidos avances en la tecnología de los circuitos integrados impulsaron el desarrollo de prácticas estándar de diseño para los circuitos digitales. VHDL (*Very High Speed Integrated Circuit Hardware Description Language*: lenguaje de descripción de hardware de circuitos integrados de muy alta velocidad) se creó como parte de tal esfuerzo y se convirtió en el lenguaje estándar industrial para describir circuitos digitales, principalmente, porque es un estándar oficial de la IEEE. En 1987 se adoptó la norma original para VHDL, llamada IEEE 1076. En 1993 se adoptó una norma revisada, la IEEE 1164.

En sus orígenes, VHDL tenía dos propósitos centrales. Primero, servía como lenguaje de documentación para describir la estructura de circuitos digitales complejos. Como estándar oficial del IEEE, ofreció una forma común de documentar los circuitos diseñados por varias personas. Segundo, VHDL aportó funciones para modelar el comportamiento de un circuito digital, lo que permitió emplearlo como entrada para programas que entonces se usaban para simular la operación del circuito.

En años recientes, aparte de usarlo para documentación y simulación, VHDL también se volvió popular para el ingreso de diseño en sistemas CAD. Las herramientas CAD se utilizan para sintetizar el código de VHDL en una implementación de hardware del circuito descrito.

VHDL es un lenguaje complejo y refinado. Aunque aprender todas sus funciones es una tarea atemorizante, para usarlo en la síntesis basta conocer un subconjunto de ellas (Brown y Vranesic, 2006).

2.7.3 Arreglos de compuertas de campos programables (FPGA)

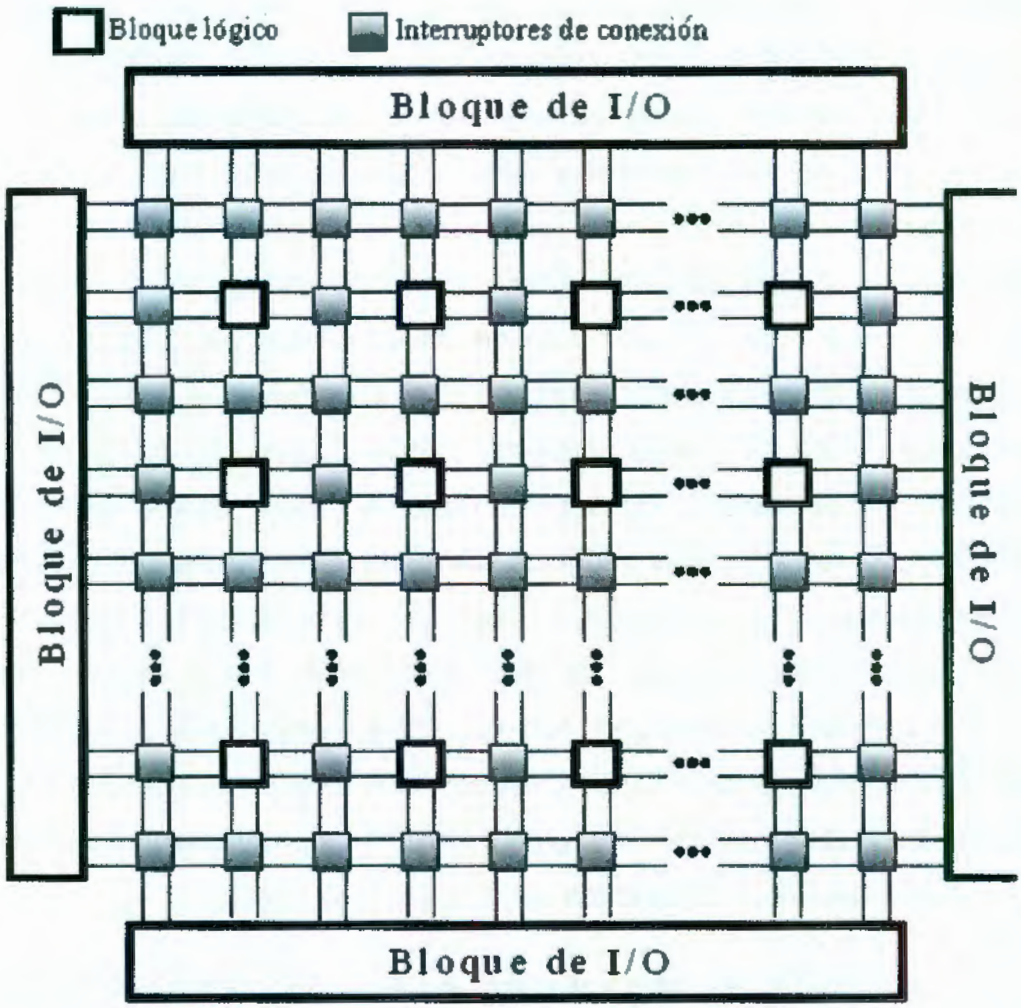
Los tipos de chips de la serie 74LS (TTL *transistor-transistor logic*: lógica transistor-transistor), los PLD (*Programmable Logic Device*: Dispositivo Lógico Programable) y los CPLD (*Complex Programable Logic Device*: Dispositivo Lógico Programable Complejo) son útiles para programar una amplia variedad de circuitos lógicos. Excepto por el CPLD, estos dispositivos son mas bien pequeños y adecuados sólo para aplicaciones hasta cierto punto simples. Incluso para los CPLD, nada más pueden acomodarse circuitos lógicos moderadamente grandes en un solo chip. Por razones de costo y rendimiento, es prudente implementar el circuito lógico deseado empleando cuantos menos chips sea posible, de modo que tanto la cantidad de circuitos en un chip como su capacidad funcional son significativas. Una forma de cuantificar el tamaño de un circuito es suponer que se construirá usando sólo compuertas lógicas simples y luego estimar cuántas de ellas se necesitan. Una medida que suele usarse es el número total de compuertas NAND de dos entradas que se necesitarían para construir el circuito; esta medida se llama número de compuertas equivalentes.

Si se utiliza la métrica de compuertas equivalentes es fácil medir el tamaño de un chip de la serie 74LS por que cada uno contiene únicamente compuertas simples. En el caso de los PLD y los CPLD la medida típica usada es que cada una de sus macroceldas represente aproximadamente 20 compuertas equivalentes. Por ende, un gran CPLD que tenga 500 macroceldas puede implementar circuitos de hasta más o menos 10,000 compuertas equivalentes.

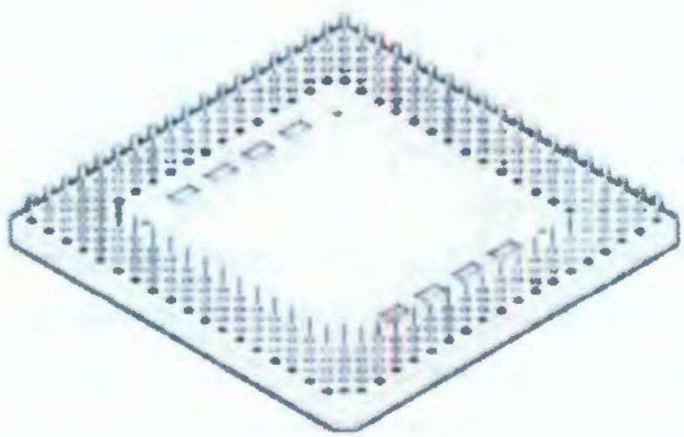
Según los estándares modernos, un circuito con 10,000 compuertas no es grande. Para implementar circuitos mayores conviene usar un tipo diferente de chip con una capacidad lógica mayor. Un *arreglo de compuertas de campos programables* (FPGA, *field-programmable gate array*) es un dispositivo lógico programable que soporta la implementación de circuitos lógicos hasta cierto punto grandes. Los FPGA son muy diferentes de los PLD y los CPLD, ya que no contienen planos AND y OR. En vez de ello, los FPGA ofrecen *bloques*

lógicos para la implementación de las funciones requeridas. La estructura general de un FPGA se ilustra en la figura 2.8. Contiene tres tipos principales de recursos: bloques lógicos, bloques de I/O para conectar a los pines del paquete, y cables de conexión e interruptores. Los bloques lógicos están dispuestos en un arreglo bidimensional, en tanto que los cables de interconexión están organizados como *canales de enrutamiento* horizontales y verticales entre filas y columnas de bloques lógicos. Los canales de enrutamiento contienen cables e interruptores programables que permiten que los bloque lógicos se interconecten de muchas formas. En la figura 2.8a se muestran dos ubicaciones de los interruptores programables; los recuadros grises adyacentes a los bloque lógicos sostienen interruptores que conectan las terminales de entrada y salida del bloque lógico a los cables de interconexión, y los recuadros grises que están diagonalmente entre bloques lógicos conectan un cable de interconexión con otro (un cable vertical con otro horizontal). También hay conexiones programables entre los bloques de I/O y los cables de interconexión. El verdadero número de interruptores programables y cables que hay en un FPGA varía en los chips disponibles en el comercio.

Los FPGA sirven para implementar circuitos lógicos con un tamaño de más de un millón de compuertas equivalentes. Los chips FPGA están disponibles en una variedad de paquetes, incluidos los paquetes PLCC (*plastic-leaded chip carrier*) y QFP (*quad flat pack*). En la figura 2.8b se ilustra otro tipo de paquete, llamado *arreglo de pines en retícula* (PGA, *pin grid array*). Un paquete PGA puede tener hasta unos cuantos cientos de pines en total, que se extienden rectos hacia afuera desde la parte inferior del paquete, en un patrón de retícula. Incluso otra tecnología de empacado que ha surgido se conoce como *ball grid array* (BGA). El BGA es similar al PGA excepto que los pines son pequeñas bolas redondas en vez de postes. La ventaja de los paquetes BGA es que los pines son muy pequeños; por tanto, un paquete hasta cierto punto pequeño puede tener más (Brown y Vranesic, 2006).



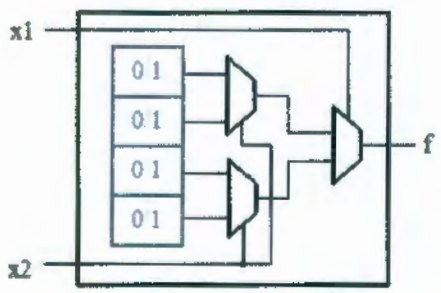
a) Estructura general de un FPGA.



b) Paquete de arreglo de pines en retícula (PGA) (vista inferior).

Figura 2.8 Arreglo de compuertas de campos programables.

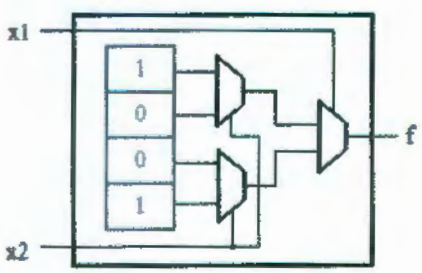
Cada bloque lógico en un FPGA tiene un pequeño número de entradas y salidas. En el mercado hay varios productos FPGA, que presentan diferentes tipos de bloques lógicos. El más usado de éstos es una *tabla de consulta* (LUT, *lookup table*), que contiene *celdas de almacenamiento* que sirven para implementar una pequeña función lógica. Cada celda puede contener un solo valor lógico 0 o 1. El valor almacenado se produce como la salida de la celda de almacenamiento. Pueden crearse LUT de varios tamaños, en los que el tamaño se define mediante el número de entradas. En la figura 2.9a se muestra la estructura de una pequeña LUT. Tiene dos entradas, x_1 y x_2, y una salida, *f*. Es capaz de implementar cualquier función lógica de dos variables. Como una tabla de verdad de dos variables tiene cuatro filas, esta LUT posee cuatro celdas de almacenamiento. Una celda corresponde al valor de salida de cada fila de la tabla de verdad. Las variables de entrada x_1 y x_2 se usan como las entradas de selección de tres multiplexores, los cuales, según la valoración de x_1 y x_2, eligen el contenido de una de las cuatro celdas como la salida de la LUT (Brown y Vranesic, 2006).



a) Circuito para una LUT de dos entradas

x1	x2	f1
0	0	1
0	1	0
1	0	0
1	1	1

b) $f_1 = \bar{x}_1\bar{x}_2 + x_1x_2$



c) Contenido de las celdas de almacenamiento en la LUT

Figura 2.9 Tabla de contenido LUT de dos entradas.

Para ver cómo se realiza una función lógica en la LUT de dos entradas considérese la tabla de verdad de la figura 2.9b. La función f_1 de esta tabla puede almacenarse en la LUT como se ilustra en la figura 2.9c. El ordenamiento de los multiplexores de la LUT realiza correctamente la función f_1. Cuando $x_1 = x_2 = 0$, la salida de la LUT la dirige la celda de almacenamiento superior, que en la tabla de verdad representa la entrada para $x_1x_2 = 00$. De manera similar, para todos los valores de x_1 y x_2, el valor lógico almacenado en la celda de almacenamiento correspondiente a la entrada en la tabla de verdad elegida por el valor particular aparece en la salida de la LUT. Dar acceso al contenido de las celdas de almacenamiento es sólo una forma en la que los multiplexores pueden usarse para implementar funciones lógicas.

En la figura 2.10 se muestra como incluir un flip-flop en un bloque lógico FPGA. El flip-flop se usa para almacenar el valor de su entrada D bajo el control de su *reloj*.

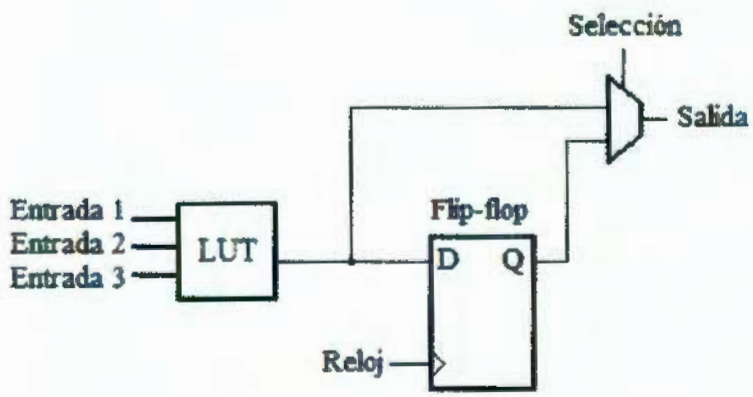


Figura 2.10 Inclusión de un flip-flop en un bloque lógico de un FPGA.

Una LUT de tres entradas tiene ocho celdas de almacenamiento porque una tabla de verdad de tres variables tiene ocho filas. En los chips comerciales de FPGA, las LUT tienen cuatro o cinco entradas, que requieren 16 y 32 celdas de almacenamiento, respectivamente.

Para realizar un circuito lógico en un FPGA, cada función lógica del circuito ha de ser lo suficientemente pequeña para encajar un solo bloque lógico. En la práctica, el circuito de un usuario se traduce de manera automática a la forma requerida con las herramientas CAD. Cuando se

implementa un circuito en un FPGA, los bloques lógicos se programan para cumplir las funciones necesarias y los canales de enrutamiento para realizar las interconexiones requeridas entre bloque lógicos. Los FPGA se configuran con el método ISP (*in-system programming, programación en el sistema*), es decir, el FPGA está unido a su tarjeta de circuito para su programación. Las celdas de almacenamiento en las LUT de un FPGA son *volátiles*, lo que significa que pierden el contenido que almacenan siempre que la fuente de poder para el chip se apague. Por tanto, el FPGA debe programarse cada vez que se aplique potencia. Con frecuencia, en la tarjeta de circuito que alberga el FPGA se incluye un pequeño chip de memoria que conserva permanente los datos, llamado *memoria programable de sólo lectura* (PROM, *programable read-only memory*). Las celdas de almacenamiento del FPGA se cargan de modo automático del PROM cuando se aplica potencia a los chips (Brown y Vranesic, 2006).

2.8 PWM

2.8.1. Modulación de ancho de pulso analógica

Existe una técnica para operar conmutadores de estado sólido y controlar así la potencia eléctrica entregada a una carga, denominada modulación por ancho de pulso, o duración de impulsos. Hay tres partes esenciales en cualquier modulación por ancho de pulso, que se ven en la figura 2.11:

- Un generador de ondas en rampa, que en general funciona a frecuencia constante.

- Un comparador, para detectar cuando el voltaje de rampa ha rebasado el voltaje de la señal de control.

- Un dispositivo electrónico que conecta la corriente a la carga, en el momento en que el comparador detecta el punto crítico en la onda rampa.

En la figura 2.11, el comparador se implementa con un amplificador operacional y el interruptor electrónico es un transistor bipolar que funciona en configuración de emisor común (Maloney, 2006).

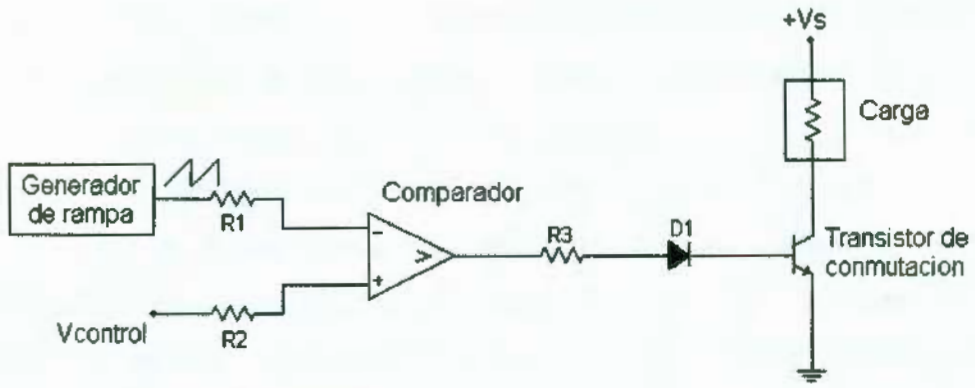


Figura 2.11 Pwm Analógico.

2.8.2. Modulación de ancho de pulso digital

Existe una técnica alternativa para evitar el empleo de amplificadores operacionales al implementar un PWM. Consiste en aprovechar el tipo de salida del PWM que es una onda cuadrada con características digitales (es decir sólo dos estados) y "programar" mediante descripción de hardware el PWM, para así integrarlo junto con todos los bloques digitales del sistema que lo requiera como señal de corrección.

Enríquez (2006) propuso el diseño mostrado en la figura 2.12, donde mediante un módulo de referencia, un controlador digital de algoritmo PID envía la señal que es almacenada en un registro. El reloj del sistema incrementa un contador, de manera que se lleva la cuenta de cuantos ciclos de reloj han transcurrido desde que inicialmente se encontraba a cero. Mediante un comparador se comprueba si el número de ciclos de reloj recibidos es menor que el valor especificado por la referencia. Este valor, que es en realidad el ancho de pulso de la señal PWM, se expresa también en ciclos de reloj, por lo que mientras que los ciclos de reloj contados sean menores que los ciclos de reloj del ancho del pulso, la señal de salida del comparador, permanecerá a nivel alto. En cuanto los ciclos de reloj superen este valor, la señal de salida del comparador se pone a cero, activando y desactivando un actuador.

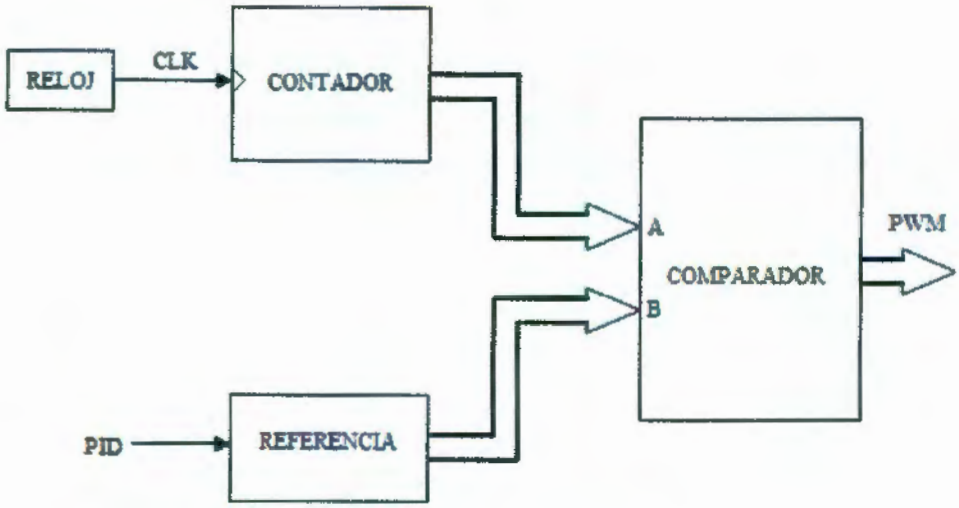


Figura 2.12 Diagrama a bloques de PWM digital.

2.9 Sensores para temperatura

Todo control industrial depende de la habilidad de medir con precisión y velocidad el valor de una variable controlada. En general, se ha encontrado que la mejor forma de medir el valor de una variable controlada es convertirla a una señal eléctrica de alguna clase y detectar ésta con un dispositivo de medición eléctrico.

Como ya se mencionó en la sección 2.5 los dispositivos que convierten el valor de una variable controlada a una señal eléctrica se denominan transductores eléctricos. El número de transductores eléctricos diferentes es muy grande. Se han inventado transductores eléctricos para medir virtualmente toda variable física, sin importar que tan complicada sea. En la industria, las variables físicas más importantes que se encuentran son la posición, velocidad, aceleración, fuerza, potencia, presión, velocidad de flujo, temperatura, intensidad luminosa, y humedad (Maloney, 2006).

2.9.1. Termopares

El dispositivo más común para medir temperaturas de procesos industriales es el termopar. Un termopar es un par de cables de metales diferentes unidos en un lazo completo, como se muestra en la figura 2.13a. Los cables diferentes tienen dos puntos de unión, uno a cada extremo del lazo. Una

unión denominada *unión caliente*, esta sujeta a una alta temperatura, la otra unión, denominada *unión fría*, esta sujeta a una baja temperatura. Cuando esto se realiza, un voltaje neto pequeño se crea en el lazo; este voltaje es proporcional a la *diferencia* entre las dos temperaturas de unión.

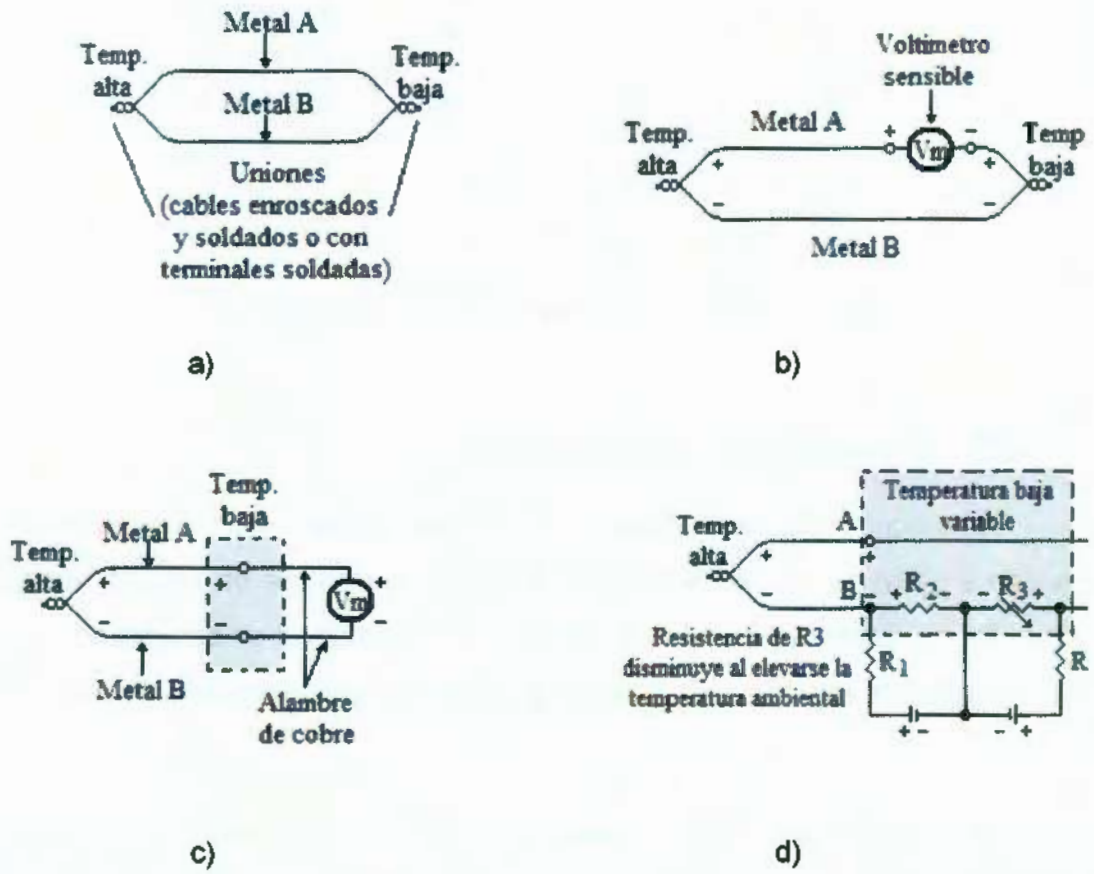


Figura 2.13 a) Termopar básico. b) Termopar con un voltímetro insertado en el lazo. c) Lazo de termopar sin unión fría entre el metal A y el metal B. d) Lazo de termopar el cual está compensado contra las variaciones en la temperatura de la unión fría.

Lo que pasa dentro de un lazo termopar es que un voltaje pequeño se produce en cada unión de metales diferentes debido a un fenómeno poco conocido denominado *efecto Seebeck*. Cuanto mayor sea la temperatura en la unión, mayor será el voltaje producido por esa unión. Además, la relación entre el voltaje y la temperatura es aproximadamente lineal; es decir, un incremento determinado en la temperatura produce un incremento determinado en el voltaje. La constante de proporcionalidad entre el voltaje y la temperatura depende de cual de los dos metales esta siendo utilizado. Debido a que un lazo completo siempre tiene dos uniones, se producen dos voltajes. Estos voltajes

se oponen entre sí en el lazo, como lo muestra la figura 2.13b. El voltaje neto disponible para conducir la corriente a través de la resistencia del lazo es la diferencia entre los dos voltajes individuales de la unión, lo cual depende de la diferencia entre las dos temperaturas de unión.

Para medir la diferencia de temperatura, sólo es necesario romper el lazo abierto en un punto conveniente (en una ubicación fría) e insertar un voltímetro. El voltímetro debe ser sensible, debido a que el voltaje producido por un lazo termopar está en el rango de los milivoltios. La lectura de voltaje se puede entonces convertir a medición de temperatura mediante la referencia a tablas o gráficas estándares que relacionan estas dos variables. Las gráficas de voltaje en función de la diferencia de temperatura para varios termopares industriales populares están dadas en la figura 2.14. En cada caso el primer metal o aleación de metal mencionada en el termopar es la terminal positiva, y el segundo metal o aleación de metal es la terminal negativa.

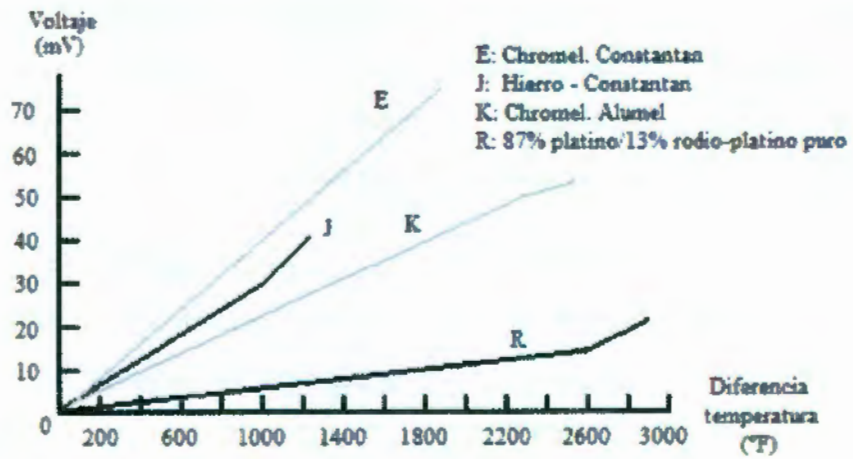


Figura 2.14 Voltaje en función de curvas de temperatura para los tipos de termopares E, J, K y R. Las palabras Chromel, Constantant y Alumel son nombres de marcas registradas de los fabricantes individuales del cable termopar.

Para evitar el problema de identificar los termopares por nombres de marcas registradas, se adoptó un código de letra para los tipos de termopares. Por tanto, los termopares de tipo *J* tienen la respuesta mostrada en la figura 2.14 sin importar el nombre particular usado para identificar la aleación de metal. Lo mismo sucede para los termopares de tipo *K* y *R*, y para otros tipos no graficados en la figura 2.14.

Cuando se inserta un voltímetro dentro del lazo termopar, por lo general es más conveniente insertarlo como lo muestra la figura 2.13c. En esa figura, el metal A y el metal B en realidad no se tocan entre sí en la unión fría. En lugar de ello, ambos metales se colocan en contacto con cables de conducción de cobre estándares. Las conexiones normalmente se hacen sobre una banda terminal. Los cables de cobre entonces se conectan al voltímetro sensible. Puede parecer que esto desestabilizaría el voltaje neto total generado por el lazo termopar, pero eso no sucede. El voltaje del lazo neto permanece siendo el mismo debido a que ahora hay dos uniones frías, una entre el metal A y el cobre, y otra entre el metal B y el cobre. La suma de los dos voltajes de unión producida por estas uniones frías es igual al voltaje que habría sido producido por la sola unión fría del metal A con el metal B. Por supuesto, las dos uniones frías deben mantenerse a la misma temperatura que la unión única habría captado. Esto no representa problema, dado que los cables de cobre y las terminales están siempre dentro de algún recubrimiento que está térmicamente aislado del proceso que esta siendo medido y que está sujeto a la misma temperatura a la que una unión sola habría sido sujeta, es decir, la temperatura ambiente en la ubicación industrial. Por tanto el circuito en la figura 2.13c producirá la misma lectura que el circuito en la figura 2.13b.

Otro asunto es importante tener en cuenta cuando se usan los termopares en la industria. Esto concierne a la variación en la temperatura ambiente en las uniones frías. He aquí la situación: se sabe de antemano la temperatura de las uniones frías, entonces, en lugar de relacionar la lectura del voltímetro con la *diferencia* de temperatura, se puede relacionar con la propia temperatura de la unión caliente. Esto sería posible debido a que se podria construir las tablas de temperatura en función del voltaje para reflejar el hecho de que las uniones frías se encuentran a una cierta *temperatura de referencia*, como se conoce.

Como ejemplo, el termopar de tipo *J* de la figura 2.14. La gráfica muestra que a una diferencia de temperatura de 400 °F, el voltaje de lazo del termopar es de 12 mV. Si se sabe que la unión fría estaba siempre a 75 °F por ejemplo, entonces se puede concluir que un voltaje de lazo de 12 mV

representó una temperatura de unión caliente de 475 °*F* (475 °*F* – 75 °*F* = 400 °*F*). Por tanto tiempo como la unión fría fue mantenida constantemente en la temperatura de referencia de 75 °*F*, se puede tomar como guía la tabla del termopar y agregar 75 °*F* a cada lectura de diferencia de temperatura. El valor de la temperatura resultante representará la temperatura en la unión caliente.

De hecho, esto es exactamente lo que se hace en las tablas de termopar industriales. La cifra de 75 °*F* se eligió debido a que representa una ponderación muy razonable de la temperatura ambiente promedio en un ambiente industrial. (En las tablas de termopar para uso de *laboratorio*, la temperatura de referencia por lo general se considera de 32 °*F*, el punto de congelación del agua.)

Para que este método trabaje con mayor precisión, la unión fría debe ser constantemente mantenida en la temperatura de referencia de 75 °F. Esto es por lo general poco práctico a menos que el instrumento de medición de la temperatura se ubique en un cuarto con control de aire acondicionado. No obstante, con toda probabilidad, el instrumento de medición se localiza fuera con el equipo y la maquinaria industrial. La temperatura ambiente puede variar fácilmente de cerca de 50 °F en el invierno a cerca de 100 °F en el verano; estos cambios en la temperatura del ambiente son comunes. Debido a esta variación en la temperatura de unión fría, los lazos de termopar deben ser *compensados*.

Un método de compensación automática simple se ilustra en la figura 2.13d. Las dos alimentaciones de voltaje de cd y los cuatro resistores están dispuestos de manera que los voltajes a través de R_2 y R_3 son opuestos. Las polaridades del voltaje a través de R_1 y R_4 no tienen importancia, debido a que R_1 y R_4 están fuera el lazo del termopar. R_3 es un resistor sensible a la temperatura, que tiene un coeficiente de temperatura negativo. Esto significa que su resistencia desciende cuando aumenta la temperatura. El circuito esta diseñado de modo que a 75 °F el pequeño voltaje a través de R_2 equivale al pequeño voltaje a través de R_3. Los voltajes a través de los dos resistores exactamente se cancelan entre sí, y la lectura del voltímetro no se ve afectada. Ahora, si la temperatura de unión fría se eleva por encima de 75 °F, la lectura

del voltímetro tendería a disminuir debido a la diferencia más pequeña entre las uniones frías y calientes. Esto tendería a producir una lectura de temperatura medida que es *más baja* que la temperatura real en la unión caliente. Sin embargo, la resistencia de R_3 disminuye al elevarse la temperatura de unión fría, lo que da como resultado un voltaje más pequeño a través de sus terminales. El voltaje R_3 ya no iguala al voltaje de R_2. Por tanto, la combinación de $R_2 - R_3$ introduce un voltaje neto al interior del lazo lo cual tiende a incrementar la lectura del voltímetro. Una revisión cuidadosa de las polaridades de los voltajes en la figura 2.14d probará que esto es así. Debido al diseño del circuito de compensación, el voltaje neto introducido por la combinación $R_2 - R_3$ cancela exactamente la disminución en el voltaje de lazo causada por la elevación de la temperatura en la unión fría.

Si la temperatura de unión fría cayera por debajo de 75 °*F*, la combinación de $R_2 - R_3$ introduciría un voltaje neto en la dirección opuesta. Esto compensa el incremento en el voltaje de lazo causado por la diferencia más grande de temperatura entre las uniones fría y caliente.

Muchos instrumentos de registro/medición de temperaturas industriales usan un puente de balanceo automático para indicar temperaturas. El voltaje de lazo de termopar es balanceado mediante el movimiento del selector de un potenciómetro en un circuito puente Wheatstone. El eje del potenciómetro está unido a otro eje, el cual opera la aguja indicadora de temperatura. Por tanto, por cada valor de voltaje del lazo de termopar, existe una posición correspondiente de la aguja indicadora de temperatura. Entonces se marca una escala de temperatura detrás de la aguja (Maloney, 2006).

2.9.2. Detector de resistencia de temperatura (RTDs)

La resistencia de la mayoría de los metales se incrementa, sobre un rango límite de temperatura, en una proporción linealmente razonable con la temperatura (figura 2.15). Tal relación esta dada por:

$$R_t = R_0(1 + at) \qquad 21$$

donde R_t es la resistencia a la temperatura t °C, R_0 la resistencia a 0 °C y a una constante igual al coeficiente de resistencia de temperatura para

determinado metal. Los detectores de resistencia por temperatura son simples elementos resistivos en forma de carbones de alambre de metales como el platino, níquel, o aleaciones de níquel-cobre; el platino es el más ampliamente usado. Elementos como películas delgadas de platino son a menudo hechas depositando el metal en un sustrato apropiado, elementos de alambre embobinado envuelven un alambre de platino sostenido por un cristal adhesivo a alta temperatura dentro de un tubo cerámico. Tales detectores son altamente estables y mantienen respuestas repetitivas sobre largos periodos de tiempo. Suelen responder en tiempos de 0.5 a 5 s o más (Bolton, 2003).

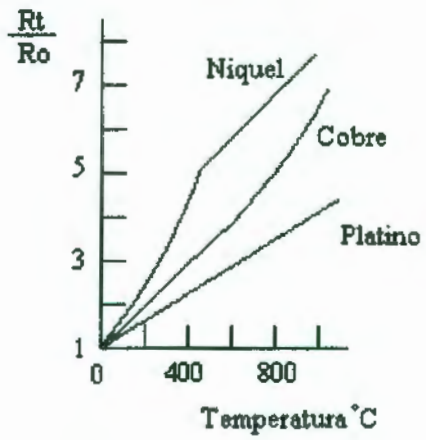


Figura 2.15 Variación de la resistencia con la temperatura para algunos metales.

2.9.3. Termistores

Los termistores son pequeñas piezas de material hechas de mezclas de óxidos de metal, tales como cromo cobalto, hierro, manganeso y níquel. Estos óxidos son semiconductores. El material es formado en elementos de varias formas, tales como molduras, discos y cilindros (figura 2.16). La resistencia de los termistores convencionales de óxido de metal decrece de una manera poco lineal con un incremento en la temperatura, como la ilustrada en la figura 2.17. Tales termistores tienen un coeficiente de temperatura negativo. Los termistores con coeficiente de temperatura positivo, están sin embargo, disponibles. El cambio en la resistencia por cambio de grado en temperatura es considerablemente más grande del aquel que ocurre con metales. La relación resistencia-temperatura para un termistor puede ser descrita por una ecuación de la forma

$$R_t = Ke^{\beta/t} \qquad 22$$

donde R_t es la resistencia a la temperatura t, con K y β siendo constantes. Los termistores tienen muchas ventajas cuando son comparados con otros tipos de sensores de temperatura. Son resistentes y pueden ser muy pequeños. Debido a su pequeño tamaño, responden muy rápidamente a cambios en temperatura. Pueden dar grandes cambios en resistencia por grado de temperatura. Su principal desventaja es que no son lineales (Bolton, 2003).

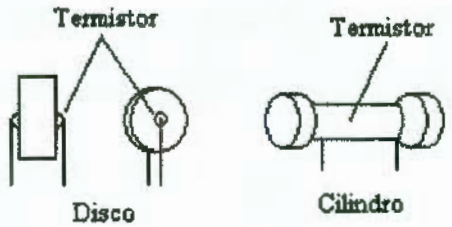


Figura 2.16 Termistores.

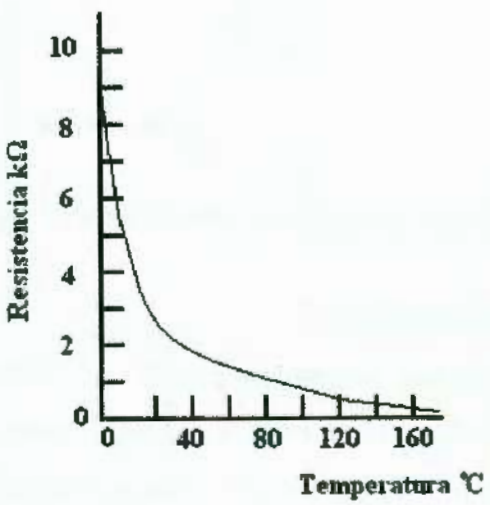


Figura 2.17 Variación de la resistencia con temperatura para un termistor típico.

2.9.4. Termodiodos y transistores

Un diodo semiconductor de unión es ampliamente usado como un sensor de temperatura. Cuando la temperatura del semiconductor dopado cambia, la modalidad de su carga sufre cambios y esto afecta la taza a la cual los electrones y vacios pueden difundirse a través de una unión p-n. Así,

cuando una unión p-n tiene una diferencia de potencial V a través de ella, la corriente *I* a través de la unión es una función de la temperatura, dada por

$$I = I_0\left(e^{\frac{eV}{KT}} - 1\right) \qquad 23$$

donde T es la temperatura en escala Kelvin, e la carga de un electrón y k e I_0 son constantes. Tomando logaritmos podemos escribir la ecuación en términos del voltaje como

$$V = \left(\frac{kT}{e}\right) ln\left(\frac{I}{I_0} + 1\right) \qquad 24$$

Así, para una corriente constante, V es proporcional a la temperatura en escala Kelvin, entonces, una medición de la diferencia de potencial en el diodo, a una corriente constante puede ser usada como una medición de la temperatura. Tal sensor es compacto como un termistor pero tiene la gran ventaja de dar una repuesta lineal en función del tiempo. Diodos como sensores de temperatura junto con acondicionadores de señal están disponibles como circuitos integrados, por ejemplo, el LM3911, un sensor de tamaño muy compacto. El voltaje de salida del LM3911 es proporcional a la temperatura en una proporción de 10 mV/°C.

De forma similar al termodiodo, para un termo-transistor el voltaje a través de la unión entre la base y el emisor depende de la temperatura y puede ser usado como una medición de la temperatura. Un método común es usar dos transistores con diferentes corrientes de colector y determinar la diferencia en los voltajes base emisor entre ellos, esta diferencia es directamente proporcional a la temperatura en escala Kelvin. Tales transistores pueden ser combinados con otros componentes de circuitos en un simple chip para disponer de un sensor con su acondicionador de señal, por ejemplo, el LM35 (figura 2.18). Este sensor puede ser usado en un rango de -40 °C a 110 °C y provee una salida de 10 mV/°C.

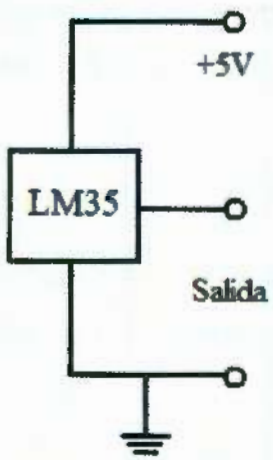


Figura 2.18 Circuito sensor de temperatura LM35.

CAPÍTULO III

METODOLOGÍA

3.1 Identificación del sistema de temperatura

Un sistema térmico como es el caso de hornos de resistencias eléctricas, hornos de llama para combustión, zonas de temperatura para fundición de materias primas etc., representan modelos de primer orden de acuerdo a la teoría de control clásica. A fin de sintonizar un controlador de cualquier tipo es necesario entonces modelar o "identificar" matemáticamente el sistema a ser controlado, una vez que se cuenta con el modelo matemático del sistema, se puede recurrir a los métodos de sintonización para determinar los parámetros necesarios (ganancias) que representarán matemáticamente a la ley de control propuesta. Por estas razones, a continuación se describe el proceso llevado a cabo para identificar al sistema donde se probará la funcionalidad del controlador.

3.1.1 Máquina de inyección de plástico

Como se mencionó en el primer capítulo, el laboratorio de la facultad de Ingeniería cuenta con una máquina de inyección de plástico para vaciado en moldes, éste plástico es fundido por resistencias eléctricas conmutadas por relevadores de estado sólido. Ya que las resistencias térmicas de la máquina de inyección de plástico son un sistema de temperatura, éstas pueden ser modeladas como un sistema de primer orden al adquirir la constante de tiempo T de la ecuación 4, cuando se les somete a una entrada escalón. Con la ayuda de un sistema de adquisición de datos se obtiene la gráfica de la figura 3.1 para el sistema de una sola zona con la variable de milivolts en el eje de las ordenadas y tiempo en segundos para las abscisas. La señal de milivolts corresponde a un factor de 10 mV/°C debido al tipo de acondicionador de señal para el elemento transductor y el tiempo en segundos, que es de hecho, el período de muestreo. La gráfica obtenida es bastante similar a la de la figura

2.5, sin embargo, es afectada por ruido eléctrico al momento de realizar la adquisición.

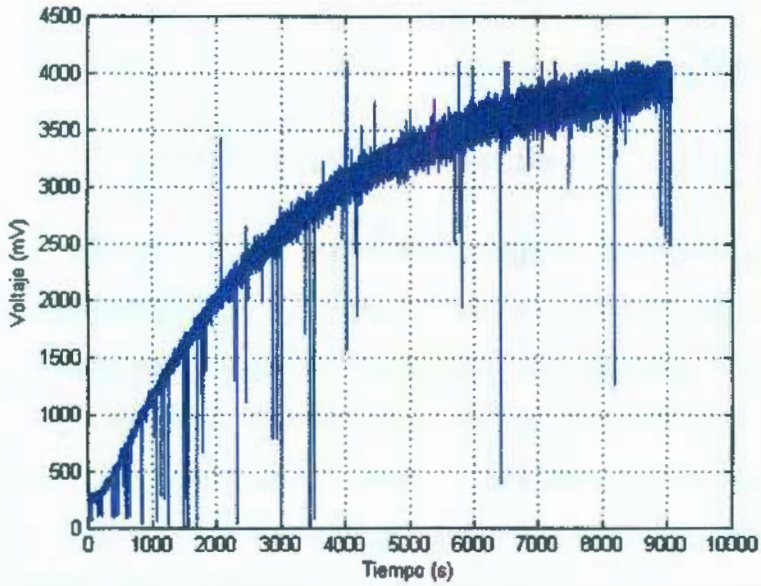


Figura 3.1 Gráfica obtenida de adquisición contaminada por ruido.

El objetivo es obtener una ecuación similar a la ecuación 4, pero de escalón no unitario, por tanto, son dos parámetros a obtener: *A* y *T*, como aparece en la siguiente ecuación en el dominio del tiempo:

$$f(t) = A(1 - e^{-\frac{t}{T}}) \qquad 25$$

o, en función de s:

$$f(s) = \frac{A}{Ts + 1} \qquad 26$$

Con el fin de obtener la constante de tiempo *T* y la magnitud *A*, es necesario visualizar una gráfica con más similitud a la de la figura 2.5, y de esta manera localizar el 63.3% de la magnitud e intersecar este punto con el tiempo, obteniendo la constante *T* en segundos. Dado que la señal esta considerablemente contaminada por ruido eléctrico, es necesario de alguna manera filtrarla para visualizarla de una forma más legible, existen muchas técnicas de filtrado para ello, dos de los mas simples son el filtro por promedio móvil y el filtro por mediana.

El filtro de mediana encuentra adición en la eliminación del ruido por impulso aleatorio aditivo, el cual se presenta como errores repentinos grandes en la señal alterada (Mitra, 2007). Por la razón anterior, es entonces seleccionado el filtro de mediana para suavizar la curva generada por las más de 9000 muestras de la figura 3.1.

El filtro de mediana se implementa deslizando una ventana de longitud impar sobre la secuencia de entrada $\{x[n]\}$ una muestra a la vez. En cualquier instante, la salida del filtro corresponde al valor de la mediana de las muestras de entrada dentro de la ventana. De manera más específica, la muestra de salida $y[n]$ en el n-ésimo instante del filtro de mediana con una ventana de longitud ($2K$+1) esta dada por (Mitra, 2007):

$$y[n] = med\{x[n-K], \ldots, x[n-1], x[n], x[n+1], \ldots, x[n+K]\} \qquad 27$$

3.2 Sintonización de leyes de control

Una vez obtenido el modelo de la planta, es necesario seleccionar una clase de ley de control así como los parámetros (ganancias) de ésta en base a los requisitos de diseño. A continuación se describen la selección de las ganancias para las leyes de control P, PD, PI y PID por el método de respuesta en frecuencia, en base a los parámetros de diseño del margen de fase $M\varphi$ y tiempo de respuesta τ deseados.

3.2.1 Proporcional (P)

Dado que en el análisis del dominio de la frecuencia el controlador proporcional no permite modificar la fase de la planta, resulta entonces no satisfactorio implementar esta ley al sistema en cuestión, además de que éste es de tipo cero y esta ley no permite eliminar el error en estado estable. Sin embargo, el controlador P mueve la curva para la ganancia de la planta, permitiendo así, modificar el margen de fase de ésta, pero en el mejor de los casos el margen de fase se situaría en un intervalo de 90° a 180°, no permitiendo así satisfacer el criterio de estabilidad en base al margen de fase de 45°.

3.2.2 Proporcional derivativo (PD)

De acuerdo al procedimiento de sintonización de la sección 2.6.3, la función L(s) para la figura 3.2 es:

$$L(s) = \frac{A(k_p + k_d s)}{Ts + B} \qquad 28$$

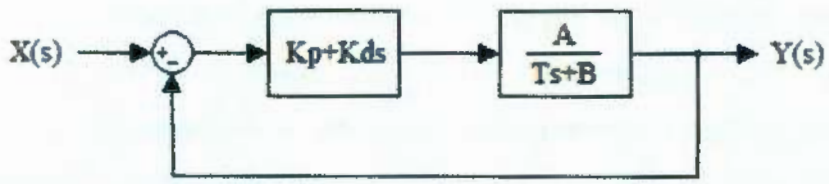


Figura 3.2 Planta con controlador PD

En el dominio de la frecuencia la magnitud y la fase para la función son respectivamente:

$$|L(j\omega)| = \frac{A\sqrt{k_p^2 + k_d^2\omega^2}}{\sqrt{B + j\omega T}} \qquad 29$$

$$\angle L(j\omega) = tan^{-1}\frac{k_d}{k_p}\omega - tan^{-1}\frac{T}{B}\omega \qquad 30$$

Al obtener el margen de fase $M\varphi$ e igualarlo al deseado:

$$180° + \angle L(j\omega) = M\varphi \qquad 31$$

$$180° + tan^{-1}\frac{k_d}{k_p}\omega - tan^{-1}\frac{T}{B}\omega = M\varphi$$

$$k_d = \frac{k_p}{\omega}tan\left(M\varphi - 180° + tan^{-1}\frac{T}{B}\omega\right)$$

Igualando a uno la magnitud del sistema

$$|L(j\omega)| = 1 \qquad 32$$

$$\frac{A\sqrt{k_p^2 + k_d^2\omega^2}}{\sqrt{B + j\omega T}} = 1$$

Se obtienen dos ecuaciones con dos incógnitas, al resolver para k_p y k_d con la frecuencia $\omega_c = {}^{1}/_{\tau}$, se obtiene:

$$k_p = \frac{\sqrt{B^2 + \frac{T^2}{\tau^2}}}{A\sqrt{1 + tan^2\left(M\varphi - 180 + tan^{-1}\frac{T}{B\tau}\right)}} \quad 33$$

$$k_d = \frac{\tau\sqrt{B^2 + \frac{T^2}{\tau^2}}\, tan\left(M\varphi - 180° + tan^{-1}\frac{T}{B\tau}\right)}{A\sqrt{1 + tan^2\left(M\varphi - 180 + tan^{-1}\frac{T}{B\tau}\right)}} \quad 34$$

3.2.3 Proporcional Integral (PI)

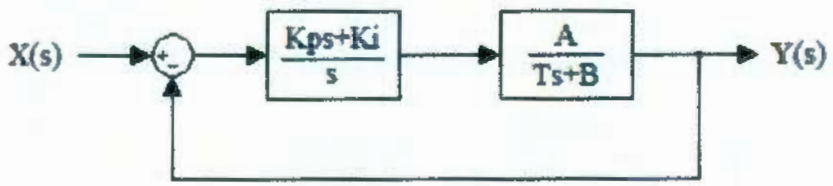


Figura 3.3 Planta con control PI

Las funciónes L(s) y L(jω) son:

$$L(s) = \frac{A(K_p s + K_i)}{s(Ts + B)} \quad 35$$

$$L(j\omega) = \frac{A(k_i + j\omega K_p)}{j\omega(B + j\omega T)}$$

La magnitud y el ángulo son:

$$|L(j\omega)| = \frac{A\sqrt{k_i^2 + k_p^2\omega^2}}{\omega\sqrt{B^2 + \omega^2 T^2}} \quad 36$$

$$\angle L(j\omega) = tan^{-1}\frac{k_p}{k_i}\omega - 90 - tan^{-1}\frac{T}{B}\omega \quad 37$$

Con la ecuación 20:

$$180° + \angle L(j\omega) = M\varphi \quad 36$$

$$180° + tan^{-1}\frac{k_p}{k_i}\omega - 90 - tan^{-1}\frac{T}{B}\omega = M\varphi$$

Al igualar la magnitud a uno:

$$|L(j\omega)| = \frac{A\sqrt{k_i^2 + k_p^2\omega^2}}{\omega\sqrt{B^2 + \omega^2 T^2}} = 1 \quad 37$$

De las ecuaciones 36, 37 y $\omega_c = {}^{1}/_{\tau}$, se obtienen:

$$k_i = \frac{\sqrt{B^2 + \frac{T^2}{\tau^2}}}{A\tau\sqrt{1 + tan^2\left[M\varphi - 90 + tan^{-1}\frac{T}{B\tau}\right]}} \quad 38$$

$$k_p = \frac{\sqrt{B^2 + \frac{T^2}{\tau^2}}\, tan\left[M\varphi - 90 + tan^{-1}\frac{T}{B\tau}\right]}{A\sqrt{1 + tan^2\left[M\varphi - 90 + tan^{-1}\frac{T}{B\tau}\right]}} \quad 39$$

3.2.4 Proporcional Integral Derivativo (PID)

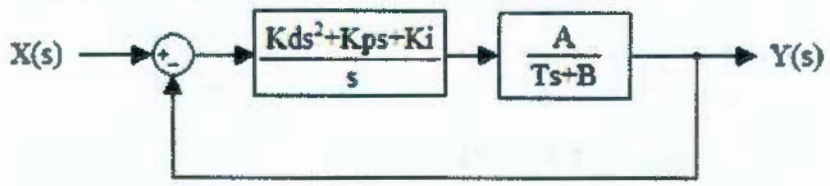


Figura 3.4 Planta con control PID

El controlador PID puede escribirse de la siguiente manera:

$$G_{PID} = K_p + K_d s + \frac{K_i}{s} = \frac{K_d\left(s^2 + \frac{K_p}{K_d}s + \frac{K_i}{K_d}\right)}{s} = \frac{K_d(s+a)(s+b)}{s} \quad 40$$

Siguiendo el procedimiento de la sección 2.6.3:

$$L(s) = \frac{Ak_d(s+a)(s+b)}{s(Ts+B)} \quad 41$$

$$L(j\omega) = \frac{Ak_d(a+j\omega)(b+j\omega)}{j\omega(B+j\omega T)}$$

$$|L(j\omega)| = \frac{AK_d\sqrt{a^2+\omega^2}\sqrt{b^2+\omega^2}}{\omega\sqrt{B^2+\omega^2T^2}} \quad 42$$

$$\angle L(j\omega) = tan^{-1}\frac{\omega}{a} + tan^{-1}\frac{\omega}{b} - 90° - tan^{-1}\frac{\omega T}{B} \quad 43$$

Con la ecuación 20 se obtiene:

$$180° + \angle L(j\omega) = M\varphi$$

$$180° + tan^{-1}\frac{\omega}{a} + tan^{-1}\frac{\omega}{b} - 90° - tan^{-1}\frac{\omega T}{B} = M\varphi$$

$$tan^{-1}\frac{1}{a\tau} + tan^{-1}\frac{1}{b\tau} = M\varphi - 90° + tan^{-1}\frac{T}{B\tau} \quad 44$$

Igualando la magnitud a la unidad:

$$|L(j\omega)| = 1$$

$$\frac{AK_d\sqrt{a^2+\omega^2}\sqrt{b^2+\omega^2}}{\omega\sqrt{B^2+\omega^2T^2}} = 1$$

$$\left(a^2+\frac{1}{\tau^2}\right)\left(b^2+\frac{1}{\tau^2}\right) = \left(\frac{1}{\tau A k_d}\right)^2\left(B^2+\frac{T^2}{\tau^2}\right) \qquad 45$$

Existen tres incógnitas (*a*, *b* y K_d) que deben ser resueltas con las ecuaciones 44 y 45, es entonces necesario proponer un valor para *a* o *b* y resolver para *b* o *a* y K_d en base al margen de fase deseado. Al proponer un valor para *b*, *a* y K_d son:

$$a = \frac{1}{\tau tan\left[M\varphi - 90° + tan^{-1}\frac{T}{B\tau} - tan^{-1}\frac{1}{b\tau}\right]} \qquad 46$$

$$K_d = \frac{\sqrt{B^2+\omega^2T^2}}{A\tau\sqrt{a^2+\omega^2}\sqrt{b^2+\omega^2}} \qquad 47$$

K_p y K_i son:

$$K_p = (a+b)K_d \qquad 48$$

$$K_i = K_d ab \qquad 49$$

3.3 Discretización de controladores

Ya que una ecuación diferencial describe el comportamiento de un algoritmo de control sobre una señal, es necesario entonces, implementar físicamente mediante dispositivos electrónicos esta ecuación diferencial, sin embargo, no es posible que un sistema digital (a diferencia de un sistema análogo) efectúe cálculos continuos en el tiempo, es en este punto cuando surge la necesidad de "discretizar" el algoritmo o ley de control en cálculos que se efectúen de manera no continua, esto es, cada cierto tiempo, siendo así posible que un sistema digital realice cálculos en estos intervalos de tiempo. Una manera muy eficiente es transformar una ecuación diferencial en una ecuación en diferencias donde se requieren sólo de operaciones aritméticas (sección 2.5.2).

A continuación se describe el procedimiento matemático llevado a cabo para modelar mediante una ecuación en diferencias los controladores tipo P, PD y PID.

3.3.1 Proporcional (P)

Llevado a cabo el método de Laplace, la relación entre la salida del controlador *y(t)* y la señal de error *x(t)* es (sección 2.4.1, ecuación 6):

$$\frac{Y(s)}{E(s)} = K_p$$

al efectuar la transformada bilineal:

$$\frac{Y(z)}{E(z)} = K_p \qquad 50$$

la ecuación en diferencias es:

$$Y(k) = K_p E(k) \qquad 51$$

Como se observa en la ecuación 48, la salida *Y(k)* es simplemente, la señal de error *E(K)* amplificada en un factor K_p.

3.3.2 Proporcional derivativo (PD)

La ecuación en el dominio de Laplace es:

$$\frac{Y(s)}{E(s)} = K_p + K_d s \qquad 52$$

Después de aplicar la transformada bilineal se obtiene:

$$\frac{Y(z)}{E(z)} = K_p + K_d\left[\frac{2(z-1)}{T(z+1)}\right]$$

$$\frac{Y(z)}{E(z)} = \frac{\left(K_p + \frac{2}{T}K_d\right)z + K_p - \frac{2}{T}K_d}{z+1} \qquad 53$$

Multiplicando numerador y denominador del lado derecho de la ecuación 53 por $\frac{z^{-1}}{z^{-1}}$:

$$\frac{Y(z)}{E(z)} = \frac{\left(K_p - \frac{2}{T}K_d\right)z^{-1} + K_p + \frac{2}{T}K_d}{1 + z^{-1}}$$

La ecuación en diferencias para el controlador PD es:

$$Y(k) = \left(K_p - \frac{2}{T}K_d\right)E(k-1) + \left(K_p + \frac{2}{T}K_d\right)E(k) - Y(k-1) \qquad 54$$

donde:

Y(K) = señal de salida del controlador en la iteración K

Y(K-1) = señal de salida del controlador en la iteración K-1

E(K) = señal de entrada al controlador en la iteración K

E(K-1) = señal de entrada al controlador (error) en la iteración K-1

K_p = ganancia proporcional

K_d = ganancia derivativa

T = período de muestreo

3.3.3 Proporcional Integral (PI)

El controlador PI en el dominio de Laplace es:

$$\frac{Y(s)}{E(s)} = K_p + \frac{K_i}{s}$$

Al aplicar la transformada bilineal evaluando $s = \frac{2(z-1)}{T(z+1)}$

$$\frac{Y(z)}{E(z)} = \frac{(K_p - {}^1/_2\, K_iT)z^{-1} - K_p - {}^1/_2\, K_iT}{z^{-1} - 1} \qquad 55$$

Como ecuación en diferencias

$$Y(k) = ({}^1/_2\, K_iT - K_p)E(k-1) + (K_p + {}^1/_2\, K_iT)E(k) + Y(k-1) \qquad 56$$

donde:

Y(K) = señal de salida del controlador en la iteración K

Y(K-1) = señal de salida del controlador en la iteración K-1

E(K) = señal de entrada al controlador en la iteración K

E(K-1) = señal de entrada al controlador (error) en la iteración K-1

K_p = ganancia proporcional

K_i = ganancia integral

T = período de muestreo

3.3.4 Proporcional Integral Derivativo (PID)

La expresión en el dominio de z para el controlador PID se obtiene de la siguiente manera:

$$\frac{Y(s)}{E(s)} = K_p + K_d s + \frac{K_i}{s}$$

$$\frac{Y(z)}{E(z)} = K_p + K_d\left[\frac{2(z-1)}{T(z+1)}\right] + \frac{TK_i(z+1)}{2(z-1)}$$

$$= \frac{\left(K_p + \frac{2}{T}K_d + \frac{T}{2}K_i\right)z^2 + \left(-\frac{4}{T}K_d + K_iT\right)z - K_p + \frac{2}{T}K_d + \frac{T}{2}K_i}{z^2 - 1}\left(\frac{z^{-2}}{z^{-2}}\right)$$

$$= \frac{\left(-K_p + \frac{2}{T}K_d + \frac{T}{2}K_i\right)z^{-2} + \left(-\frac{4}{T}K_d + K_iT\right)z^{-1} + K_p + \frac{2}{T}K_d + \frac{T}{2}K_i}{1 - z^{-2}} \qquad 57$$

Por último, la ecuación en diferencias que describe el comportamiento de un controlador PID es:

$$Y(k) = \left(-K_p + \frac{2}{T}K_d + \frac{T}{2}K_i\right)E(k-2) + \left(-\frac{4}{T}K_d + K_iT\right)E(k-1) + \left(K_p + \frac{2}{T}K_d + \frac{T}{2}K_i\right)E(k) + Y(k-2) \qquad 58$$

donde:

Y(K) = señal de salida del controlador en la iteración K

Y(K-2) = señal de salida del controlador en la iteración K-2

E(K) = señal de entrada al controlador en la iteración K

E(K-1) = señal de entrada al controlador (error) en la iteración K-1

E(K-2) = señal de entrada al controlador (error) en la iteración K-2

K_p = ganancia proporcional

K_d = ganancia derivativa

K_i = ganancia integral

T = período de muestreo

3.4 Diseño digital de estructuras para controlador

En la figura 3.5 aparece un diagrama mejor detallado al presentado en la figura 1.1, la diferencia principal, es que muestra los elementos internos dentro del bloque FPGA de la figura 1.1, dentro de éste se encuentra el control de una pantalla LCD así como introducción de la referencia por medio del mismo, el punto suma, la ley de control, el PWM y el control del ADC, así como de un reloj con un tiempo al cual estarán trabajando los módulos conectados a él. Cabe señalar que los módulos de la ley de control, el reloj, el PWM y el punto de suma están programados digitalmente dentro de la FPGA, pero la pantalla LCD y el ADC se encuentran físicamente fuera de ésta, aunque el control de éstos también esta programado digitalmente dentro de la FPGA. Los bloques dentro de las líneas segmentadas (pantalla, LCD y acondicionador de señal) así como el FPGA, se ubican dentro del prototipo. En esta sección se describe el diseño y funcionamiento del bloque principal (controlador) y en las siguientes secciones del resto.

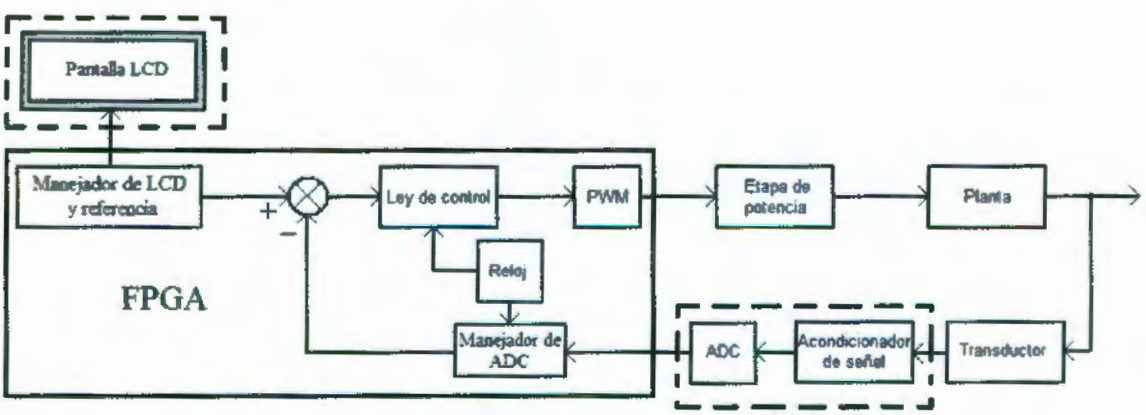


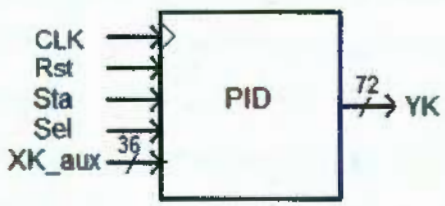
Figura 3.5 Esquema general del sistema de control

3.4.1 Estructura del controlador PID en FPGA

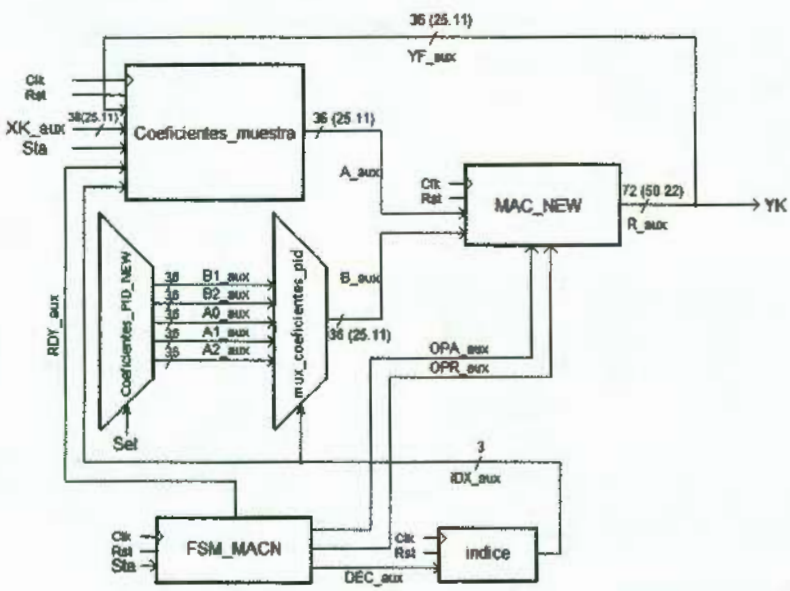
Por lo visto en la sección 3.3, es posible modelar mediante una ecuación en diferencias cualquier ley de control, a continuación se explica el diseño de los bloques digitales que pueden llevar a cabo las operaciones aritméticas exigidas por la siguiente ecuación:

$$Y(k) = a_2E(k-2) + a_1E(k-1) + a_0E(k) + b_2Y(k-2) + b_1Y(k-1) \qquad 59$$

La ecuación 59 es denominada ecuación en diferencias lineal de segundo orden, cualquier ley de control de las discretizadas en la sección 3.3 es del tipo similar, sólo que algunos de los coeficientes son cero, por ejemplo, para la ecuación en diferencias del la ley de control PID, $b_2 = 0$ y b_1 siempre es igual a la unidad. De esta manera, son necesarios bloques digitales que sumen y multipliquen, además de elementos de memoria para almacenar resultados anteriores y de esta forma la ecuación 59 pueda ser evaluada.



a) Entradas y salidas de módulo PID



b) Bloques internos de PID

Figura 3.6 Módulo digital PID.

El módulo PID de la figura 3.6 tiene como entradas a "CLK" que es el reloj maestro de 50 MHz, "RST" el reset asíncrono, "STA" para iniciar una nueva iteración, "sel" que selecciona cinco de diez coeficientes para permitir de este modo elegir entre dos ecuaciones en diferencias diferentes, es decir, dos controladores diferentes, esto con la finalidad de que cuando el controlador este funcionando, los coeficientes de la ecuación en diferencias puedan ser cambiados desde el exterior, esto al menos para dos juegos de coeficientes diferentes. Finalmente "XK_aux" y "YK" donde entra la señal de error y sale la señal de corrección hacia el PWM respectivamente.

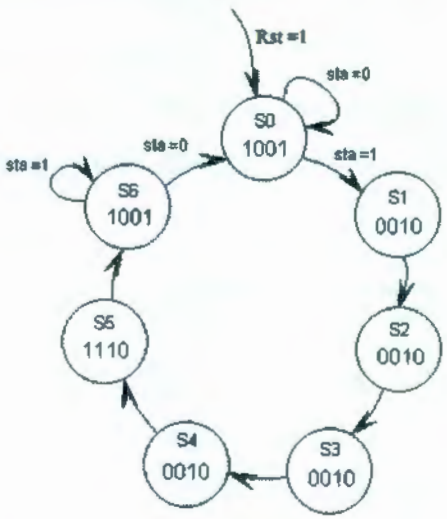


Figura 3.7 FSM "FSM_MACN".

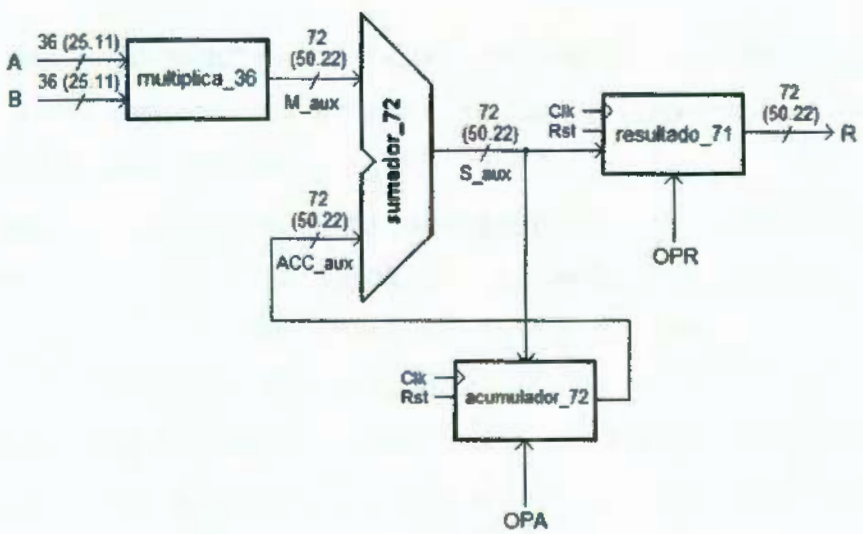
El módulo PID consta de seis bloques, un multiplicador acumulador "MAC_NEW" que multiplica y suma dos números con formato 25.11 y el resultado es un número con formato 50.22, éstas son las únicas dos operaciones aritméticas que se necesitan para evaluar la ecuación en diferencias, el multiplexor "coeficientes_PID_NEW" selecciona 5 de entre 10 coeficientes y el multiplexor "mux_coeficientes_PID" selecciona uno de estos cinco para enviarlo a la MAC. El bloque "coeficientes_muestra" selecciona el otro factor a entrar en la MAC, el orden de multiplicación y suma de estos factores es gobernado por una FSM llamada "FSM_MACN" (figura 3.6) y el contador "índice".

La máquina de estados "FSM_MACN" tiene como entradas a "CLK", "Rst" y "STA", las salidas son: "OPA", "OPR", "DEC" y "RDY", las dos primeras

cargan los bloques de la MAC "acumulador_72" y "resultado_71" con bajo y alto respectivamente, la tercera salida activa el contador "indice" para que decremente y la última salida esta dirigida a la máquina de estados "Maquina_muestra". Durante el estado s0, los dos registros en la MAC mantienen su dato, el contador "índice" mantiene su cuenta y con "RDY"=1 se mantienen los resultados de iteraciones y errores anteriores en el bloque "Coeficientes_muestra", al inicializarse la máquina se permite que el registro "acumulador_72" en la MAC carge los resultados que se van generando, por la multiplicación y suma en la ecuación en diferencias, después de que se ha efectuado una iteración completa, en s5 se carga el registro "resultado_71" y así se obtiene el resultado, con "RDY" = 1 "maquina_muestra" a su vez actualiza los registros "registros_muestra", esto significa que se preparan nuevamente los siguientes datos de errores y resultados anteriores.

3.4.1.1 MAC

a) Entradas y salidas

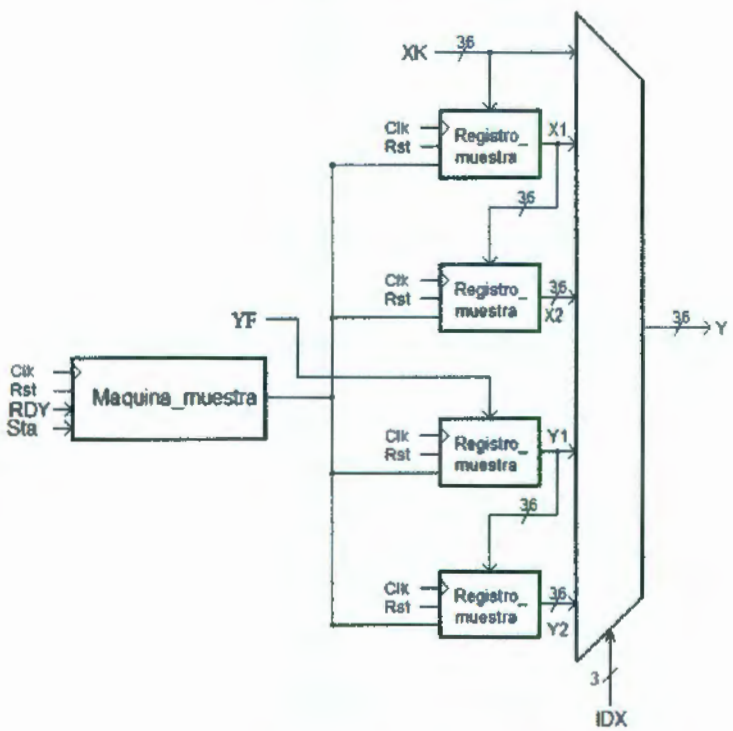


b) Bloques internos

Figura 3.8 Multiplicador – acumulador "MAC_NEW".

La figura 3.8 muestra los bloque internos de el bloque "MAC_NEW", un multiplicador "multiplica_36" multiplica dos números A y B y el resultado es sumado a la multiplicación de dos números anteriores por medio de el sumador "sumador_72", el resultado es almacenado en "acumulador_72" este es un registro de 72 bits, una vez que se han efectuado las 5 sumas (inicialmente suma cero a la primera multiplicación), el registro restante "resultado_71" carga el resultado de una iteración. Dado que el sistema funciona a una velocidad de 50 MHz, lleva tan solo 100ns realizar una iteración completa.

a) Entradas y salidas



b) Bloques internos

Figura 3.9 Registro de coeficientes.

3.4.1.2 Registro de coeficientes

El bloque "coeficientes_muestra" de la figura 3.9 tiene como tarea almacenar dos de los resultados anteriores $y(k-1)$ y $y(k-2)$ así como los errores $e(k)$, $e(k-1)$ y $e(k-2)$, y seleccionar uno de estos por medio de un multiplexor para dirigirlo hacia la MAC, el selector de este multiplexor es la cuenta de el contador "índice" visto anteriormente. Cuando "FSM_MACN" indica a "maquina_muestra" de la figura 3.10 que se ha efectuado una iteración, ésta última realiza un corrimiento en los registros "registro_muestra" y prepara de esta forma los nuevos datos para la siguiente iteración. El código VHDL para el módulo PID y sus bloques se encuentra en apéndice A.1.

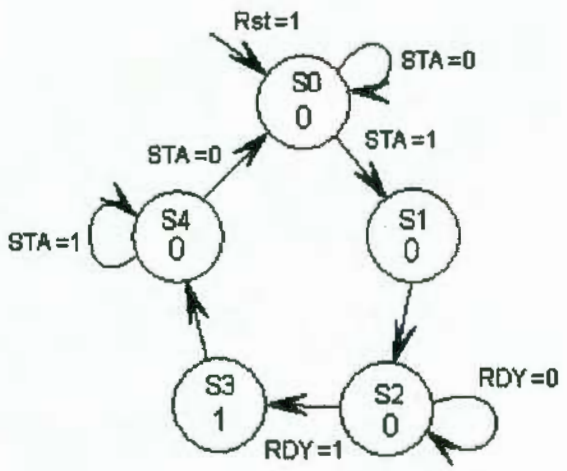


Figura 3.10 FSM "Maquina_muestra".

3.5 Diseño digital FPGA de módulos adicionales

Es necesario tener un panorama general de todas las partes digitales denominadas "módulos" en esta tesis, estos módulos son, de hecho, las estructuras digitales dentro del bloque "FPGA" mostrado en la figura 1.1. El bloque PID de la figura 3.11 es el explicado en la sección 3.4, los bloque restantes son considerados módulos adicionales, para completar el sistema de control. En la figura 3.11 se ilustra un diagrama de todos los módulos que conforman el sistema de control, dentro de estos módulos, se encuentran otros bloques, que son explicados en los apartados siguientes.

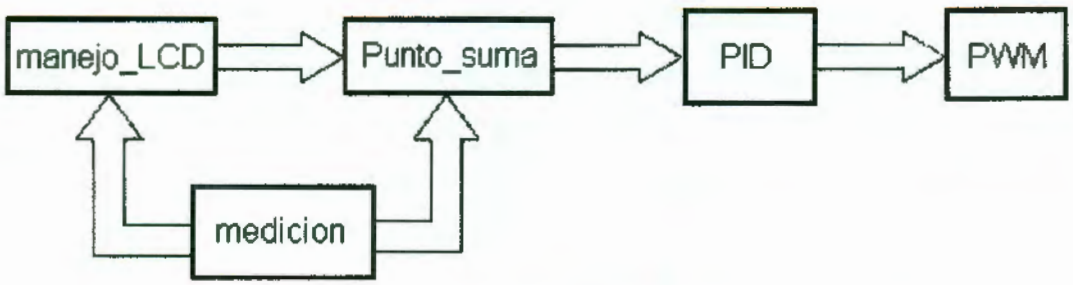


Figura 3.11 Diagrama simplificado de módulos digitales.

Ya que gran parte de los procesos térmicos industriales son monitoreados con termopares tipo J como elementos transductores, el dispositivo es diseñado teniendo esto en cuenta. Al seleccionar un termopar tipo J como sensor de temperatura, es entonces necesario un "acondicionador de señal" que amplifique la señal de salida del termopar tipo J, además de que provea un circuito compensador de unión fría, para así obtener un voltaje lineal proporcional a la temperatura dentro del proceso. Existe un acondicionador comercial con las características requeridas, es el caso del AD596 de "ANALOG DEVICES" (figura 3.12) que funciona como transductor de temperatura y acondicionador de señal para termopares tipo J, éste entrega un voltaje CD proporcional a la temperatura en 10mV/°C. De esta manera, es posible conectar directamente la señal de salida del acondicionador al convertidor analógico digital. La hoja de datos aparece en A.6.

Figura 3.12 AD596 de ANALOG DEVICES

3.5.1 Diseño digital de estructura para ADC

Se seleccionó el convertidor analógico a digital ADS 7816 de BURR-BROWN, sus principales características son las siguientes: convertidor de 12 bits con tiempo de muestreo máximo de 200KHz, operación de baja potencia

con modo de bajo consumo automático, interface serial síncrona y una entrada diferencial. El voltaje de referencia puede ser variado de 100mV a 5 V, con una correspondiente resolución de 24µV a 1.22mV. Este ADC aparece en la figura 3.13, la hoja de datos en apéndice A.5.

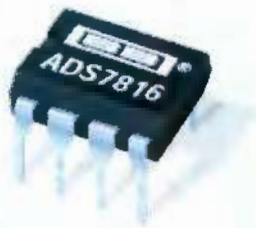


Figura 3.13 ADC ADS7816

Hasta este punto es importante definir un factor numérico característico de la señal de retroalimentación, la ganancia, esto es, en cuantos bits se incrementará la salida del ADC por cada grado centígrado aumentado en el proceso monitoreado. Se emplea un voltaje de referencia de 5 V, por tanto, la resolución del ADC es la siguiente:

$$Resolución\ ADC = \frac{V_{ref}}{2^{\#bits}-1} = \frac{5}{2^{12}-1} = 1.221mV \qquad 60$$

Esto quiere decir que la salida del ADC aumenta 1 bit por cada 1.22 mV en su entrada. Dado que el acondicionador de señal del termopar aumenta 10mV/°C (que de hecho es su resolución), la ganancia *h* es:

$$h = \left(10\frac{mV}{°C}\right)\left(\frac{1}{1.22}\frac{bit}{mV}\right) = 8.19\frac{bit}{°C} \qquad 61$$

Esto significa que el número binario proporcionado por el ADC aumenta en una 8.19 unidades por cada °C incrementado en el sistema. La utilidad de este factor es importante cuando se programan los bloques digitales de interfaz con el usuario, como se verá en la sección del diseño para manejo de LCD.

Como se mencionó al principio de esta sección, es ideal dividir todos los diseños digitales en módulos, uno de los módulos es el de "medición", dentro de este módulo se encuentran los bloques de control del ADC "ADCADS7816", un bloque para dividir entre *h* "factor" y un convertidor de código binario a BCD "Bin_BCD_conv". Este módulo se muestra en la figura

3.14, su código VHDL y el de sus bloques está en apéndice A.2. El módulo medición tiene como entradas el reloj principal "CLK" de 50 MHz, un reset asíncrono "Rst", "Inicio" que indica realizar una conversión y "Din", el bit de datos serial proveniente del ADC. Las salidas son "CS" y "DCLK" que gobiernan el control del ADC, "med_BCD" 12 bits correspondientes a la conversión realizada en código BCD para su envío a pantalla LCD y "med_bin", que también es la conversión pero en código binario simple, estos doce bits son enviados al módulo "punto suma".

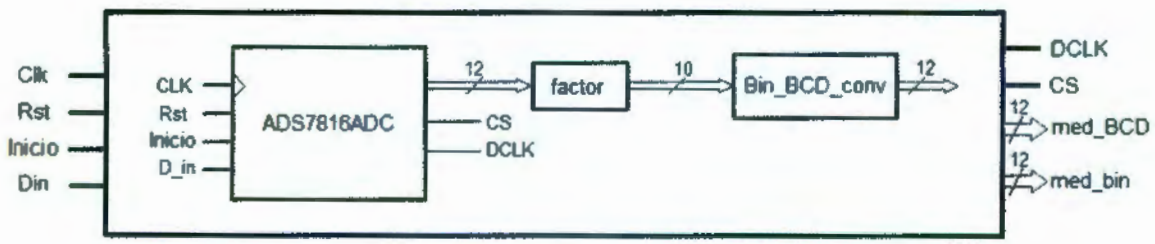


Figura 3.14 Módulo digital "medición" para adquisición de datos.

El convertidor ADS7816 tiene tres terminales digitales, dos para su modo de operación y una es la señal análoga convertida en digital, éstas son CS, DCLK y D_{OUT}. El diagrama de ondas para su operación, se muestra en la figura 3.15. La señal DCLOCK sincroniza la transferencia de datos con cada bit que es trasmitido con el flanco negativo de DCLOCK. Un flanco negativo de la señal CS inicia la conversión y la transferencia de datos. Los primeros 1.5 a 2.0 periodos del ciclo de conversión son usados para muestrear la señal de entrada. Después del segundo flanco negativo de DCLOCK, D_{OUT} es habilitado y una señal de bajo aparecerá en esta terminal por un período de DCLOCK. En los siguientes 12 períodos de DCLOCK, el resultado de la conversión aparecerá por D_{OUT}, el bit más significativo primero. Después de que el bit menos significativo ha sido sacado, si CS permanece en bajo, subsecuentes señales en DCLOCK repetirán el dato de salida pero en formato del menos significativo primero. Para simplificar código, se selecciona la opción de la figura 3.15, cuando los doce bits de la conversión han sido transferidos (MSB primero), CS regresa a alto.

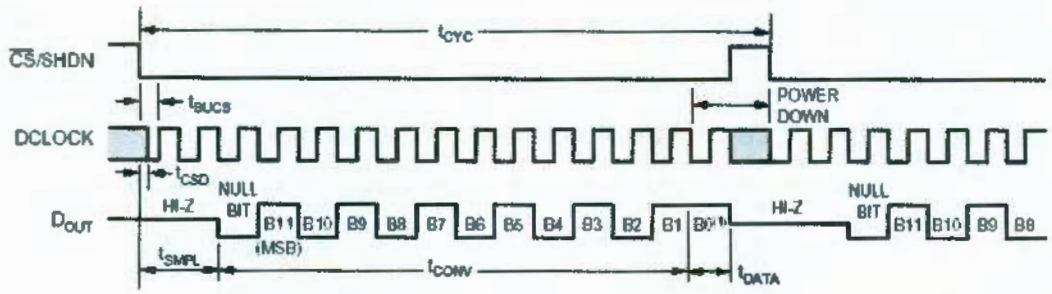
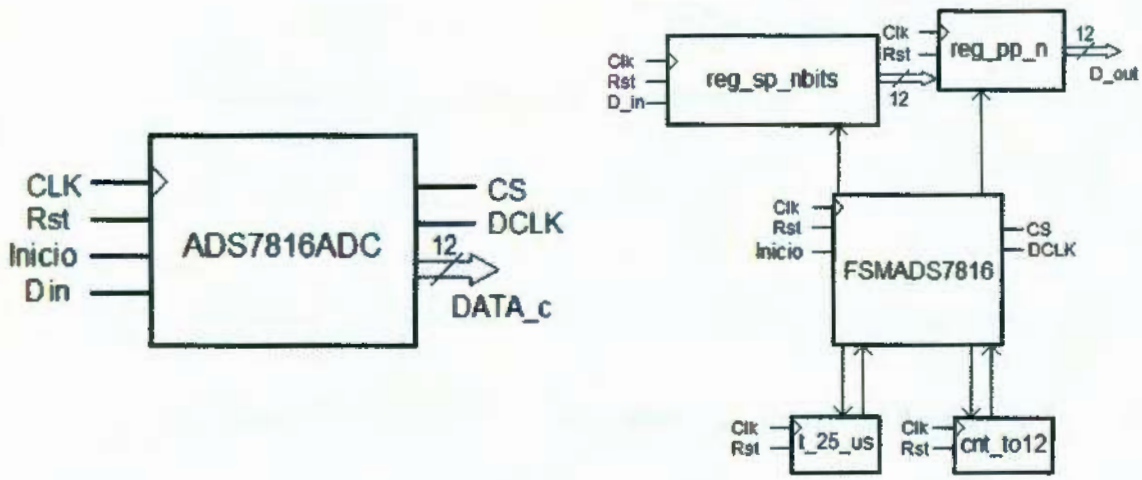


Figura 3.15 Diagrama de tiempo básico para el ADS7816.



a) Bloque ADC_ADS_7816 b) Bloques de ADC_ADS_7816

Figura 3.16 Bloques "ADS7816ADC".

En la figura 3.16a) aparece el bloque ADS7816ADC, su función es la de generar las señales requeridas por el ADC además de capturar los 12 bits seriales, de esta manera el bloque ADS7816ADC esta a la vez compuesto de otros seis bloques con el propósito de efectuar estas tareas. Estos bloques aparecen en la figura 3.16b).

El principal bloque es una máquina de estados nombrada "FSMADS7816", el diagrama de estados se muestra en la figura 3.17, las salidas son CS, DCLK, L_sp, L_pp, T_a, cnt_a y aparecen dentro de los círculos de cada estado con el mismo orden. Esta máquina genera las señales CS y DCLK hacia el ADC y sincroniza otros dos bloques, que son un registro serie-paralelo con corrimiento de derecha a izquierda de 12 bits y un registro paralelo-paralelo de 12 bits. Esta máquina contiene 13 estados, básicamente, estos estados son los de la señal DCLK de la figura 3.15 para un tiempo de t_{cyc}. La señal DCLK es generada con un periodo de 50 µs, esto es posible con una

señal enviada al bloque "t_25_us", se trata de un contador síncrono que regresa otra señal a la cuenta de 1249 pulsos del reloj maestro de 50 MHz. El bloque "reg_sp_nbits" es actualizado inmediatamente después de los flancos positivos de DCLK esto a partir del cuarto (estado 8), a fin de evitar mas estados por cada flanco positivo de DCLK, es posible ciclar la máquina en el estado 10 por 12 veces correspondientes al número de bits, cuando se envía la señal de corrimiento al registro serie paralelo también se incrementa en una unidad un contador llamado "cnt_to12", cuando este llega a 12 cuentas retorna una señal para que sea evaluada durante el estado 10 y finalice el ciclo s10-s7, durante el estado 11 es actualizado el registro paralelo-paralelo y el contador reiniciado a cero.

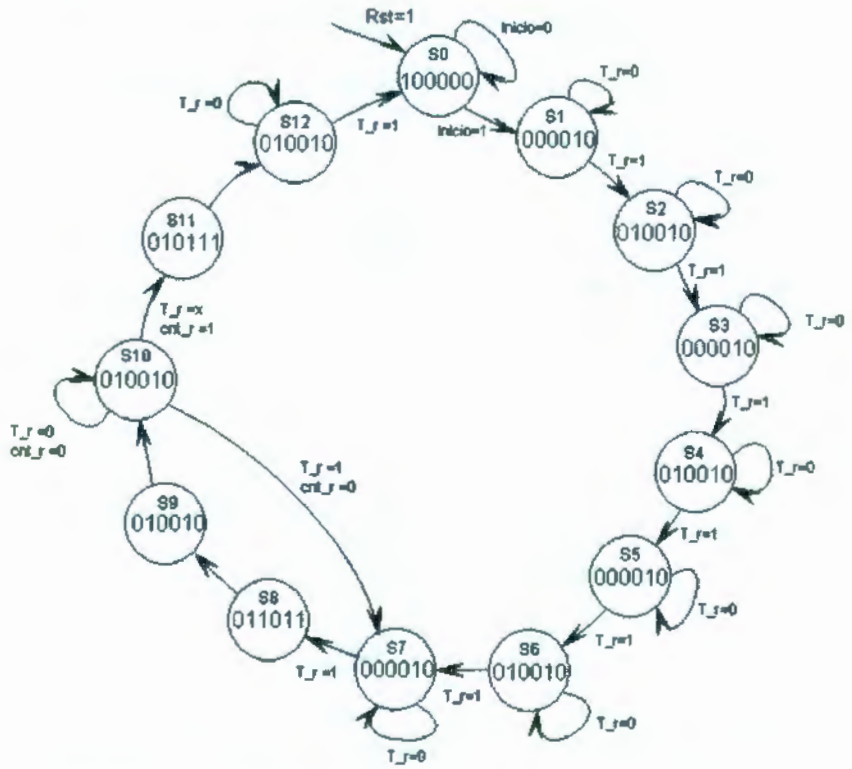


Figura 3.17 Diagrama de estados para "FSMADS7816".

Al terminar el ADC de enviar los doce bits de la conversión completa, el bloque "reg_pp_n" es actualizado y almacena los doce bits de esta conversión.

Los doce bits resultado de la conversión analógico-digital pueden entonces ser enviados directamente al punto de suma del controlador, pero necesitan ser convertidos a código BCD para su visualización en el display

LCD, antes de ello, esta cantidad necesita ser dividida entre la ganancia *h* (como se mencionó al principio de esta sección), esto es posible al multiplicar el número de doce bits por el reciproco de la ganancia, para esto es necesario establecer un número de punto fijo. El inverso de la ganancia *h* es igual a 0.1221001221 en formato decimal, con un formato binario de punto fijo 12.11 es suficiente para expresar el inverso de la ganancia *h* como

$$\frac{1}{h} = 0.1221001221_{10} \approx 000000000000.0001111101_2 \qquad 58$$
$$= 0.1220703125_{10}$$

El bloque toma 10 bits de la parte entera del número resultante de multiplicar la conversión por el reciproco de *h* y éste resultado es enviado al siguiente bloque que es un convertidor de código binario a BCD nombrado "Bin_BCD_conv", el método utilizado es un proceso utilizado en forma de tabla donde se puede calcular manualmente la conversión, que es en realidad un algoritmo con los siguientes pasos:

1. Recorrer el número binario una posición a la izquierda, hasta que el número del número binario en alguna columna de BCD sea mayor o igual a 5.

2. Sumar tres a esa columna.

3. Volver al paso 1 hasta que las posiciones con los bits del número binario inicial se encuentren vacías.

El algoritmo anterior es aplicado en la tabla 3.1.

VHDL permite programar de manera que al compilar el código escrito, este se ejecute de manera secuencial con el uso del tipo de variable "variable". El código VHDL está en apéndice A.2.4.

Tabla 3.1 Conversión binario a BCD de 10 bits

	Millares	Centenas	Decenas	Unidades	Binario
Bin					9 8 7 6 5 4 3 2 1 0
Inicio					1 1 1 1 1 1 1 1 1 1
Cambio1				1	1 1 1 1 1 1 1 1 1
Cambio2				1 1	1 1 1 1 1 1 1 1
Cambio3				1 1 1	1 1 1 1 1 1 1
Suma 3				1 0 1 0	1 1 1 1 1 1 1
Cambio4			1	0 1 0 1	1 1 1 1 1 1
Suma 3			1	1 0 0 0	1 1 1 1 1 1
Cambio5			1 1	0 0 0 1	1 1 1 1 1
Cambio6			1 1 0	0 0 1 1	1 1 1 1
Suma 3			1 0 0 1	0 0 1 1	1 1 1 1
Cambio7		1	0 0 1 0	0 1 1 1	1 1 1
Suma 3		1	0 0 1 0	1 0 1 0	1 1 1
Cambio8		1 0	0 1 0 1	0 1 0 1	1 1
Suma 3		1 0	1 0 0 0	1 0 0 0	1 1
Cambio9		1 0 1	0 0 0 1	0 0 0 1	1
Suma 3		1 0 0 0	0 0 0 1	0 0 0 1	1
Cambio1 0	1	0 0 0 0	0 0 1 0	0 0 1 1	
Número BCD	1	0	2	3	
BCD	12	1 1 8	7 4	3 0	
Z	22	2 1 1 8	1 1 7 4	1 1 3 0	9 0

3.5.2 Diseño digital de estructura para manejo de LCD

El propósito de agregar una pantalla LCD es la de establecer una interfaz entre el usuario y el dispositivo (controlador), de manera que el usuario pueda introducir de manera digital la temperatura deseada para el sistema térmico, así como visualizar la temperatura en tiempo real. Una pantalla de dos líneas y dieciséis caracteres es suficiente para los dos datos a visualizar. Las principales entradas al prototipo son tres botones UP, DN y ENTER que son para subir y bajar la referencia y actualizarla hacia el controlador, estos botones estarán directamente conectados al módulo "manejo_LCD".

El proceso de visualización en la pantalla es gobernado por un microcontrolador incorporado a ella, siendo en este caso el JHD1611. De esta manera es necesario enviar comandos de control a esta pantalla como el

manejo de cualquier dispositivo digital. Ya que estos comandos son enteramente digitales, es fácil generarlos con circuitos implementados sobre el mismo FPGA.

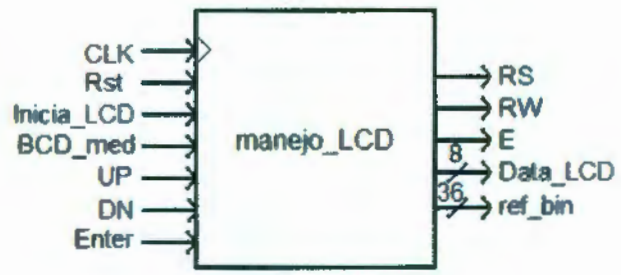
En la tabla 3.2 se describen los pines para el funcionamiento de la pantalla LCD. La pantalla tiene modos de funcionamiento de 4 y 8 bits en el bus de datos, el control con cuatro bits ahorra pines de conexión, pero es más complejo, mientras que con 8 bits se necesitan las 8 conexiones disminuyendo la necesidad de programación de bloques digitales.

Tabla 3.2 Asignación de pines de pantalla LCD

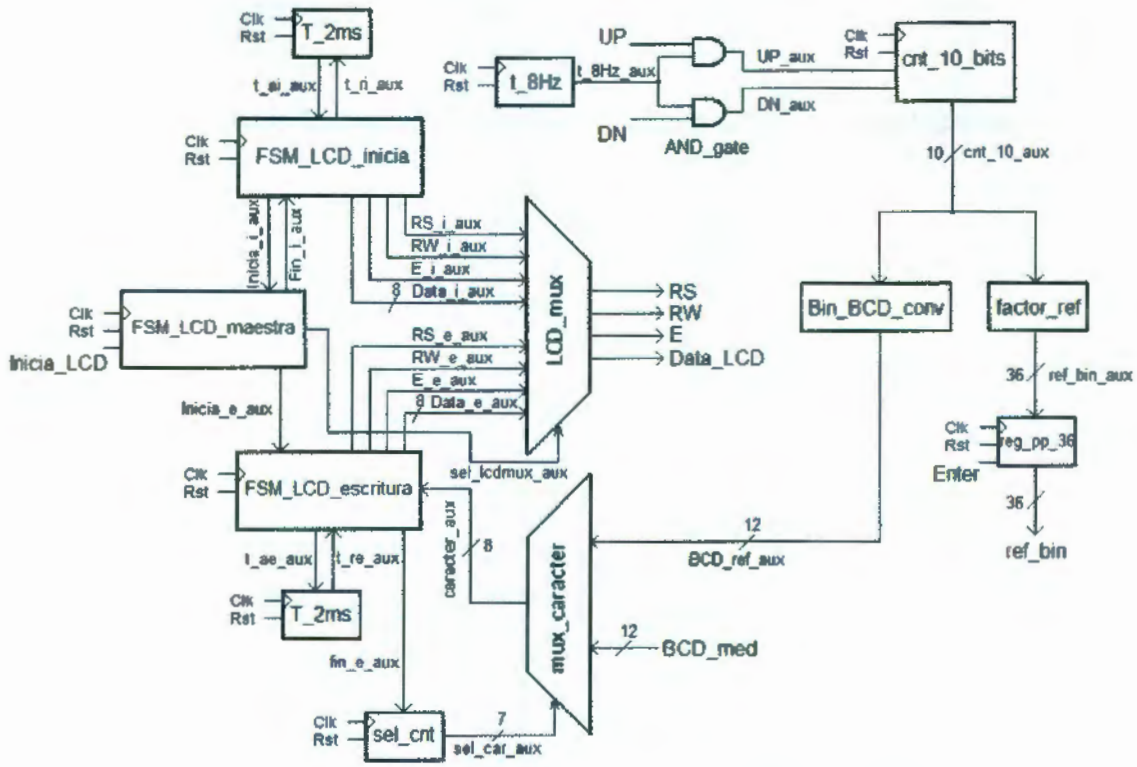
N° de PIN	Símbolo	Descripción
1	VSS	Masa
2	VDD	Alimentación
3	VO	Voltaje de ajuste del contraste
4	RS	Selección de registro
5	RW	Lectura/escritura
6	E	Enable
7	D0	Bit de datos menos significativo
8	D1	Bit de datos
9	D2	Bit de datos
10	D3	Bit de datos
11	D4	Bit de datos
12	D5	Bit de datos
13	D6	Bit de datos
14	D7	Bit de datos mas significativo
15	LED+	Alimentación de iluminación
16	LED-	Masa

Con el fin de implementar el menor número posible de circuitos sobre el FPGA, el display funcionará solamente en modo de escritura, es decir, nunca se lee información de éste. En la figura 3.18 aparece el diagrama completo del módulo "manejo_LCD", en éste una FSM denominada "FSM_LCD_maestra" (figura 3.20) activa la secuencia de inicialización de la pantalla llevada a cabo por otra FSM llamada "FSM_LCD_inicia" (figura 3.21) dónde solo se dan tres comandos de instrucción hacia la pantalla, éstos son: el modo de funcionamiento, control de encendido y apagado y limpiar pantalla. Estos comandos son enviados de acuerdo al diagrama de ondas de la figura 3.18 y

con un tiempo de 2 ms que es el tiempo máximo correspondiente al comando de limpiar pantalla.



a) Entradas y salidas



b) Bloques digitales

Figura 3.18 Módulo digital "manejo_LCD".

Al terminarse la secuencia de inicialización (modo de instrucciones), se comienzan a escribir caracteres por medio de la FSM llamada "FSM_LCD_escritura" (figura 3.22), durante el estado s3 de ésta, se incrementa en una unidad el contador "sel_cnt", los bits correspondientes a su cuenta son a la vez el selector del multiplexor "mux_caracter", éste multiplexor selecciona que carácter es el próximo a imprimir en pantalla y es en su entrada donde esta conectada la salida "med_BCD" del módulo "medición". "FSM_LCD_maestra"

también selecciona por medio del multiplexor "LCD_mux" los comandos a enviar hacia la pantalla, si son instrucciones de inicialización o comandos de caracteres a imprimir.

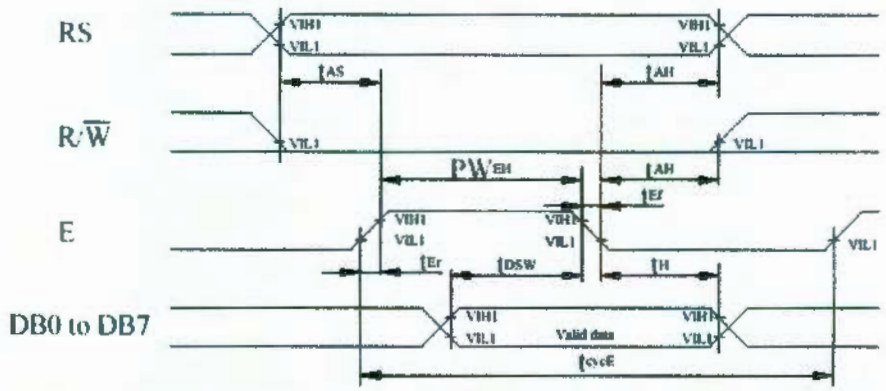


Figura 3.19 Diagrama de ondas requeridas por la pantalla LCD.

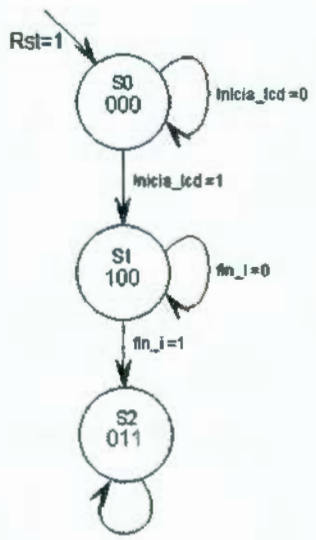


Figura 3.20 FSM de control para LCD "FSM_LCD_maestra".

Con el fin de eliminar los rebotes de los interruptores momentáneos de ajuste de referencia, una señal de 8 Hz es enviada a un contador de 10 bits ("cont_10bits") pero a través de una compuerta AND, de esta manera, no importan los rebotes mecánicos de los interruptores UP o DN, el contador se incrementara o decrementará sólo una vez, sin embargo, si se mantiene pulsado el interruptor, el contador conmutara 8 veces por segundo, de esta manera, el usuario puede ajustar con mayor rapidez referencias grandes.

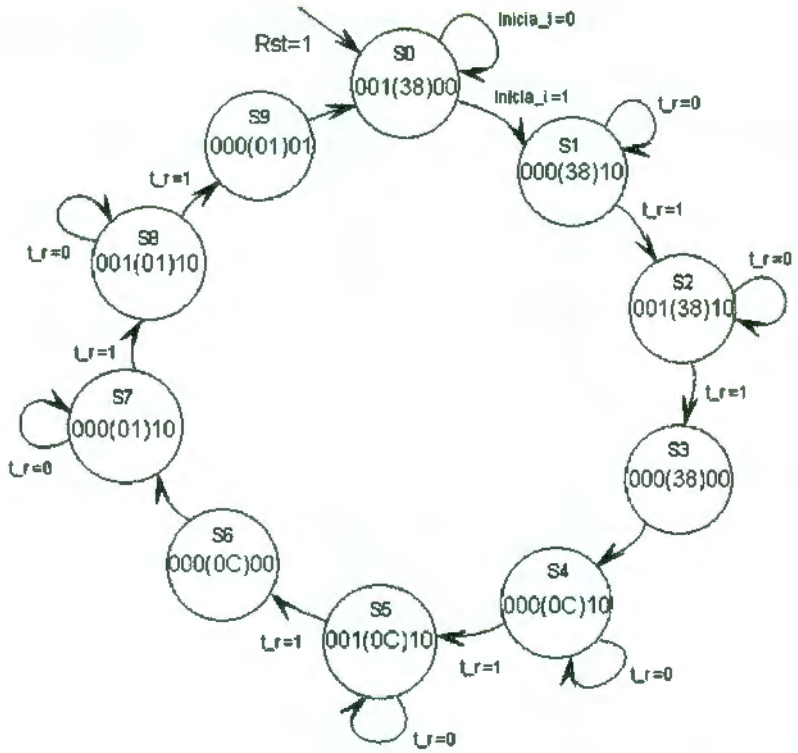


Figura 3.21 Inicialización de pantalla con FSM "FSM_LCD_inicia".

Debido a la ganancia h, es también necesario un bloque que multiplique la referencia por esta ganancia y, de esta manera, el resultado se convierta en el minuendo del punto de suma. Es adicionado también un registro de 36 bits para que actualice por medio del interruptor momentáneo "enter" la referencia y el módulo PID comience a efectuar iteraciones con una nueva referencia, de esta forma, el operador primero ajusta la referencia deseada para después oprimir "enter".

3.5.3 Diseño digital de estructura para PWM

Ya que la planta (resistencias térmicas) es energizada por corriente alterna, su control debería ser capaz de modificar la amplitud o la frecuencia del voltaje al que esta conectada, pero dado que esta conexión es efectuada por relevador de estado sólido, es entonces necesario modular el ancho de pulso del tiempo de conexión. Esto se logra con un PWM digital donde el ancho de pulso de la onda cuadrada depende de la magnitud del número binario proveniente del controlador.

En la figura 3.23 se muestran los bloques del diseño propuesto, el módulo "PWM" tiene como entradas a "CLK", "Rst", "D" que se trata de la señal

proveniente del algoritmo de control y a "Inicio_PWM" para inicializar el modulo. La única salida es "PWM_s", ésta es la señal que esta dirigida a los relevadores electrónicos y es, de hecho, la señal de corrección al actuador.

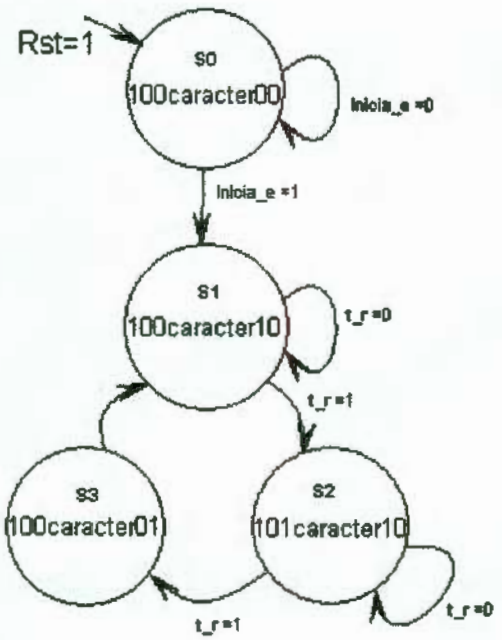
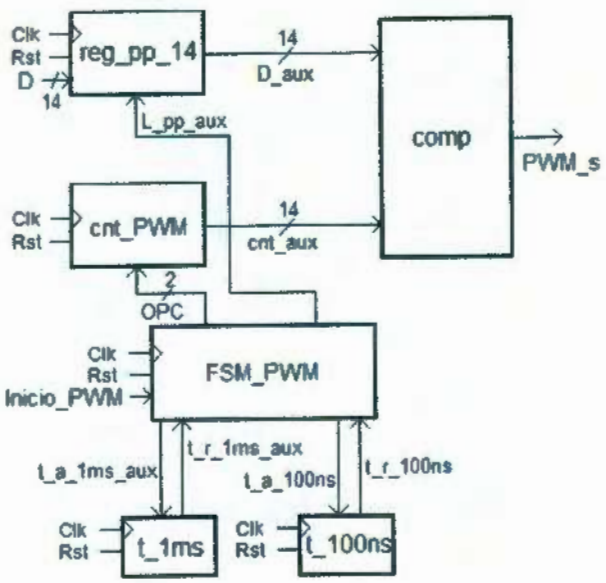


Figura 3.22 Máquina de escritura "FSM_LCD_escritura".

En la figura 3.24 se muestra la máquina de estados llamada "FSM_PWM", sus entradas son: Clk, Rst, e Inicio_PWM, las salidas "t_a_1ms" y "t_a_100ns" controlan a los temporizadores "t_1ms" y "t_100ns", la salida "L_pp" es la señal de carga del registro "reg_pp_14", y "OPC" que es una señal de dos bits, controla al contador "cnt_PWM". Durante el estado de reset s0 se mantiene en cero a los dos temporizadores, se mantiene el dato de entrada en el registro de 14 bits y se limpia a cero al contador "cnt_PWM" con OPC = 1x, al iniciarse el módulo PWM se carga el dato actual en el registro, durante el estado s2 se activan los temporizadores de 1ms y 100 ns además de que se mantiene la cuenta del contador con OPC = 00, transcurridos 100 ns la máquina conmuta al estados s3 y se incrementa en una unidad el contador con OPC = 01, al permanecer en s4, se evalúa la entrada t_r_1ms, si ha pasado un 1ms que se trata del ciclo de trabajo del PWM se vuelve a cargar otro dato en s1 y se realiza el ciclo nuevamente, si no ha pasado 1ms se vuelve a incrementar en 1 al contador "cnt_PWM". La señal de salida es generada por el

comparador "comp", y es alta si el dato "D" es más grande que la cuenta del contador, si "D" es menor a esta cuenta, la salida "PWM_s" es baja.

a) Entradas y salidas



b) Bloques internos de "PWM"

Figura 3.23 Módulo digital "PWM".

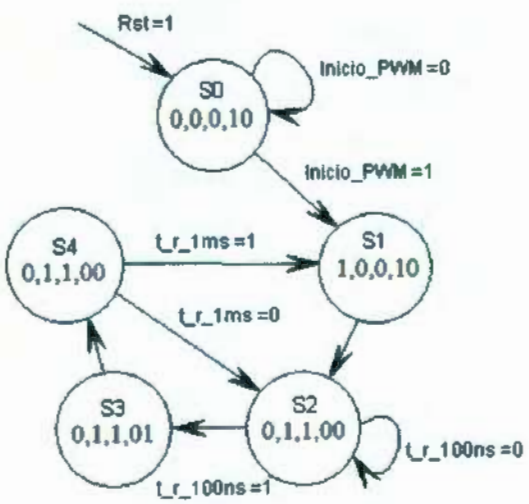


Figura 3.24 Máquina de estados "FSM_PWM".

3.6 Diseño de PCB

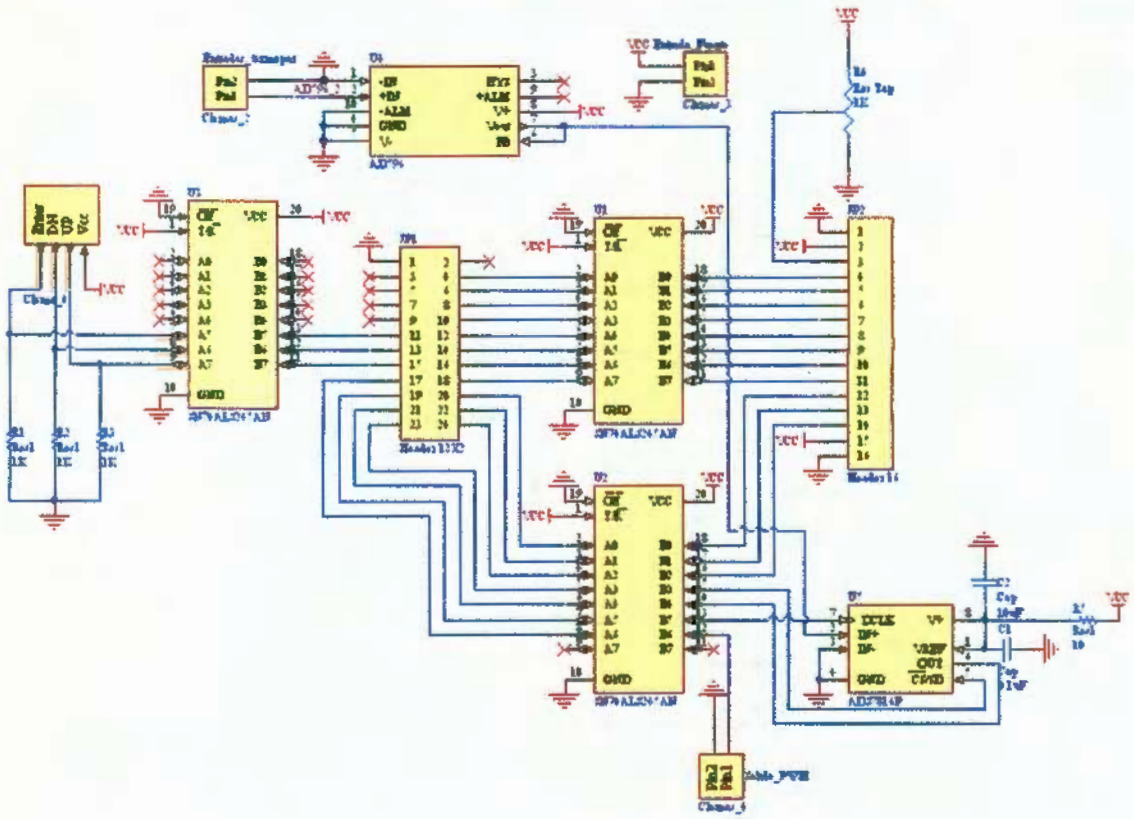


Figura 3.25 Esquemático para PCB.

En la figura 3.25 aparece el diagrama esquemático para la tarjeta construida. Sobre la tarjeta fueron requeridos espacios para el ADC ADS7816, el acondicionador AD596, un resistor variable para el contraste de la pantalla LCD, además de resistencias, capacitores y terminales de conexión hacia el FPGA, la pantalla, los relevadores de estado sólido y desde el termopar.

En la figura 3.26 se muestra el enrutado de la tarjeta para su construcción.

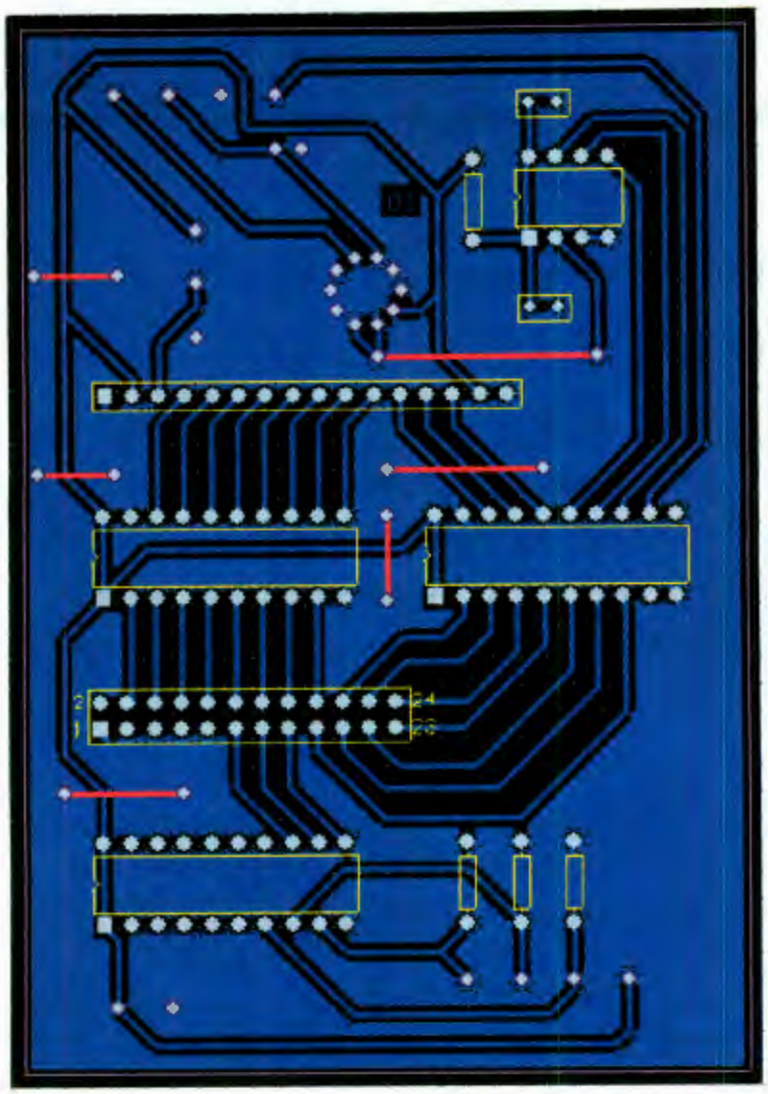


Figura 3.26 Diseño PCB de tarjeta.

CAPÍTULO IV

RESULTADOS Y ANÁLISIS

4.1 Identificación

MATLAB dispone de un comando directo para filtrar un vector de muestras por filtro de mediana, se trata de *medfilt1*, la figura 4.1 muestra la señal de la figura 3.1 después de implementar el filtro de mediana con ayuda de MATLAB y utilizando una ventana (número de muestras) de 21 elementos, se facilita entonces, obtener gráficamente A (temperatura en estado estable) y T (constante de tiempo).

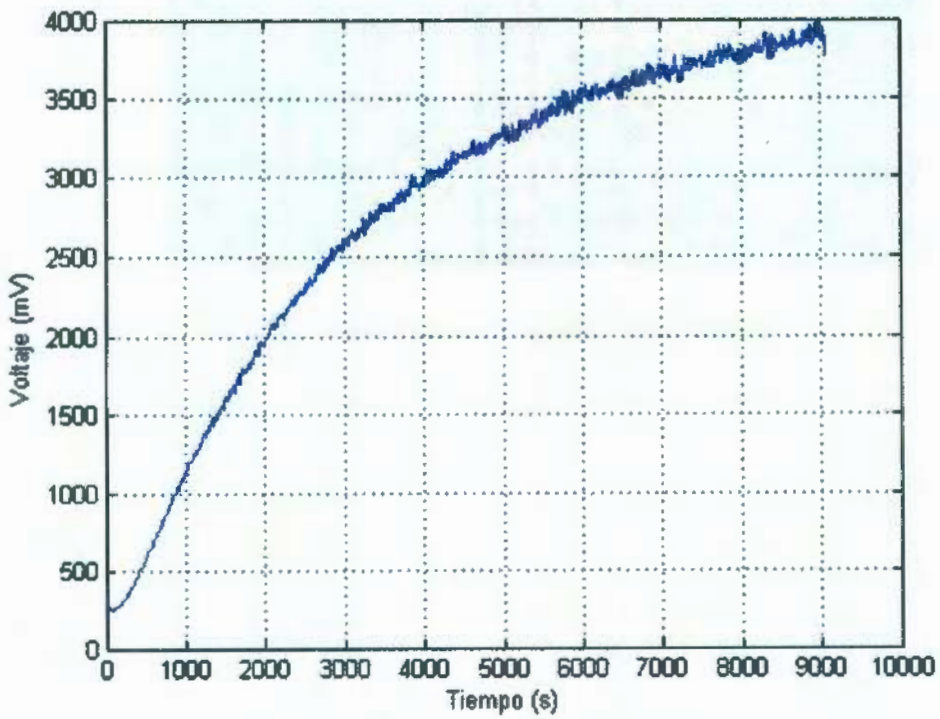


Figura 4.1 Señal de adquisición filtrada.

Dado que la temperatura de inicio fue la temperatura ambiente, estos son aproximadamente 22°C, es necesario restarlos de la temperatura de estado estable (*A*), aproximar el 63.2% de este resultado e intersecar con el eje

del tiempo, dando como resultado la constante (T), se obtiene finalmente la siguiente función de transferencia para la planta:

$$f(s) = \frac{3800}{2553s + 1} \qquad 62$$

o

$$f(s) = \frac{1.436}{s + {}^{1}/_{2553}} \qquad 63$$

En forma de salida/entrada:

$$\frac{C(s)}{R(s)} = \frac{1}{2553s + 1} \qquad 64$$

En la figura 4.2 aparece la planta graficada por Matlab para su validación, la curva roja es similar a la de la figura 4.1, pero en este caso también aparece la curva azul que se graficó con la función de transferencia de la ecuación 64 sometida a un escalón.

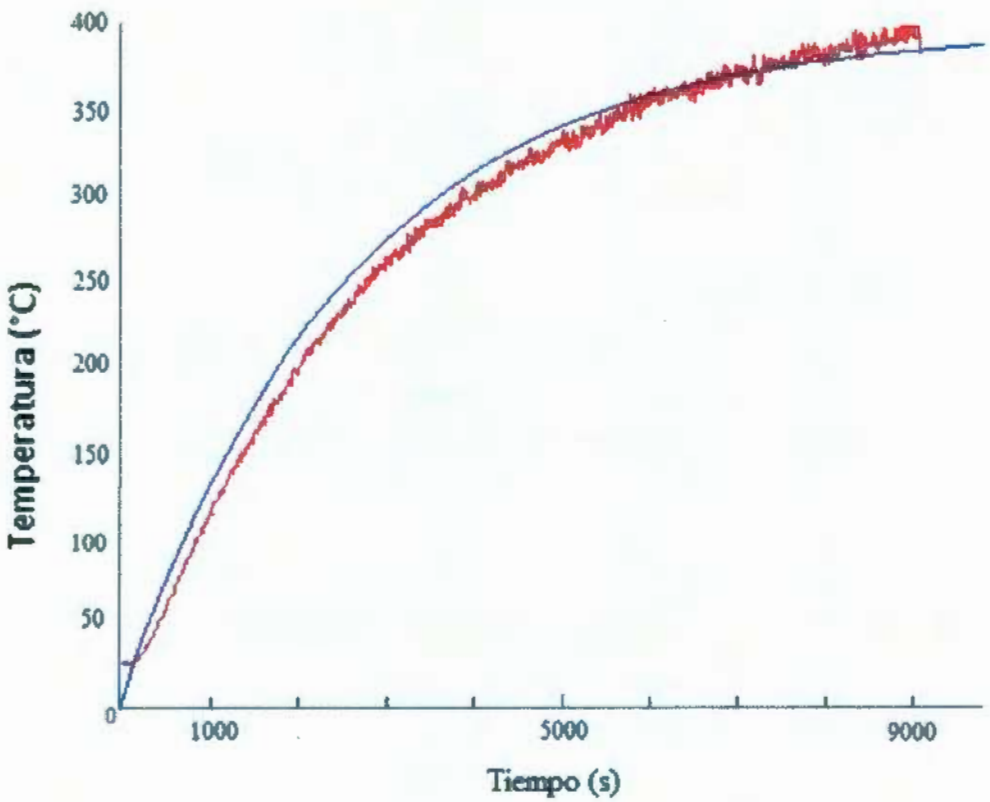


Figura 4.2 Validación para planta.

4.2 Sintonización

Ya que el controlador P y PD no eliminan el error en estado estable para un sistema de primer orden, a continuación se calculan las ganancias de los controladores PI y PID.

4.2.1 Controlador PI

Con base en las ecuaciones 38 y 39, un tiempo de respuesta $\tau = 5000\ s$, margen de fase $M\varphi = 45°$ para la planta con la ecuación 61 se obtiene:

$$k_i = \frac{\sqrt{B^2 + \frac{T^2}{\tau^2}}}{A\tau\sqrt{1 + tan^2\left[M\varphi - 90 + tan^{-1}\frac{T}{B\tau}\right]}}$$

$$k_i = \frac{\sqrt{1 + \left(\frac{2553}{5000}\right)^2}}{5000\sqrt{1 + tan^2\left[45° - 90° + tan^{-1}\frac{2553}{5000}\right]}} = 2.13 \times 10^{-4} \qquad 65$$

$$k_p = \frac{\sqrt{B^2 + \frac{T^2}{\tau^2}}\, tan\left[M\varphi - 90 + tan^{-1}\frac{T}{B\tau}\right]}{A\sqrt{1 + tan^2\left[M\varphi - 90 + tan^{-1}\frac{T}{B\tau}\right]}}$$

$$k_p = \frac{\sqrt{1^2 + \frac{2553^2}{5000^2}}\, tan\left[45° - 90° + tan^{-1}\frac{2553}{5000}\right]}{\sqrt{1 + tan^2\left[45° - 90° + tan^{-1}\frac{2553}{5000}\right]}} = -3.8348 \times 10^{-3} \qquad 66$$

4.2.2 Controlador PID

A partir de las ecuaciones 46, 47 48 y 49 se obtienen las ganancias del algoritmo PID, ya que son de hecho 2 ecuaciones y 3 incógnitas, se propone un valor a *b* o *a* en base al margen de fase deseado, con $M\varphi = 45°$, $\tau = 5000\ s$, *b*=1, se escribe:

$$a = \left\{5000tan\left[45° - 90° + tan^{-1}\frac{2553}{5000} - tan^{-1}\frac{1}{5000(1)}\right]\right\}^{-1} \qquad 67$$

$$a = -6.1690 \times 10^{-4}$$

Se obtienen así las ganancias:

$$k_d = \frac{\sqrt{1+\left(\frac{2553}{5000}\right)^2}}{5000\sqrt{(2.13\times10^{-4})^2+(5000^2)^{-1}}\sqrt{(1)^2+(5000^2)^{-1}}} \qquad 68$$

$$k_d = 0.34621$$

$$k_p = (a+b)k_d = (-6.1690\times10^{-4}+1)(0.34621) \qquad 69$$

$$k_p = 0.345996$$

$$k_i = k_d ab = 0.34621(-6.1690\times10^{-4})(1) \qquad 70$$

$$k_i = -0.000214$$

4.3 Discretización

4.3.1 Controlador PI

Al tener las ganancias numéricas, éstas se sustituyen en la ecuación en diferencias 56, obteniendo así la ecuación en diferencias para el controlador PI:

$$Y(k) = 3.8454\times10^{-3}E(k-1) - 3.8241\times10^{-3}E(k) + Y(k-1) \qquad 71$$

4.3.2 Controlador PID

De igual forma, sustituyendo las ganancias de las ecuaciones 68, 69 y 70 en la ecuación 58 se obtiene:

$$Y(k) = 6.578191E(k-2) - 13.848417E(k-1) + 7.270184E(k) + Y(k-2) \qquad 72$$

4.4 Simulación de estructuras digitales del controlador

Con el fin de probar la funcionalidad del controlador del sistema, esto es, el módulo PID de la sección 3.4.1, a continuación se somete a éste a una entrada escalón unitario. La ecuación en diferencias que el diseño puede resolver es la ecuación 59, con los coeficientes de la ecuación para el controlador PID de la ecuación 72. Siendo el objetivo de la prueba comparar los resultados analíticos contra digitales, en la tabla 4.1 se dan los coeficientes de la ecuación en diferencias (59) en formato decimal y en binario (formato 25.11), así como en sistema complemento a2 en caso necesario. Estas conversiones binarias son aproximadas a sus equivalentes decimales.

Tabla 4.1 Coeficientes para la ecuación en diferencias

	Formato	
Coeficientes	*Decimal*	*Binario*
a_2	*6.578191*	*00000000000000000000000110.10010100000*
a_1	*-13.848417*	*11111111111111111111110010.00100110111*
a_0	*7.270184*	*00000000000000000000000111.01000101001*
b_2	*1*	*00000000000000000000000001.00000000000*
b_1	*0*	*00000000000000000000000000.00000000000*

En las siguientes operaciones, se calculan los resultados hasta K=5 con el uso de la ecuación 72:

$$Y(k=1) = 6.578191(0) - 13.848417(0) + 7.270184(1) = 7.270184$$
$$Y(k=2) = 6.578191(0) - 13.848417(1) + 7.270184(1) = -6.578233$$
$$Y(k=3) = 6.578191(1) - 13.848417(1) + 7.270184(1) + 7.270184 = 7.270184$$
$$Y(k=4) = 6.578191(1) - 13.848417(1) + 7.270184(1) - 6.578233 = -6.578275$$
$$Y(k=5) = 6.578191(1) - 13.848417(1) + 7.270184(1) + 7.270184 = 7.2701$$

Mediante el simulador de VHDL se obtienen los siguientes resultados para igual número de iteraciones (se muestran en formato hexadecimal 50.22):

$$Y(K=1) = 000000000001D14800_{16(50.22)} = 7.2700195313_{10}$$
$$Y(K=2) = FFFFFFFFFFFE5B0000_{16(50.22)} = -6.878125$$
$$Y(K=3) = 000000000001D14800_{16(50.22)} = 7.2700195313_{10}$$
$$Y(K=4) = FFFFFFFFFFFE5B0000_{16(50.22)} = -6.878125$$
$$Y(K=5) = 000000000001D14800_{16(50.22)} = 7.2700195313_{10}$$

En las figuras 4.3, 4.4 y 4.5 se muestra la simulación de la estructura digital del controlador para las cinco iteraciones.

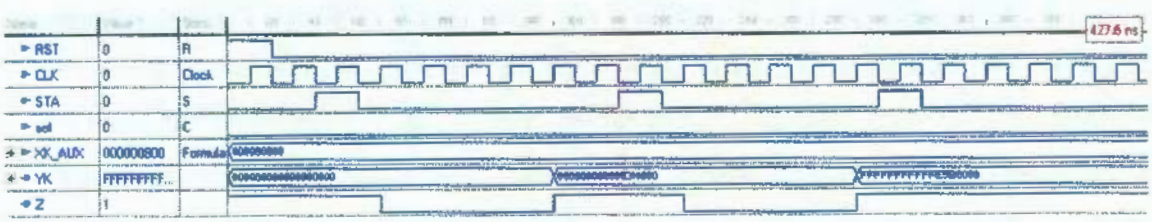


Figura 4.3 Iteración para K=1 y K=2

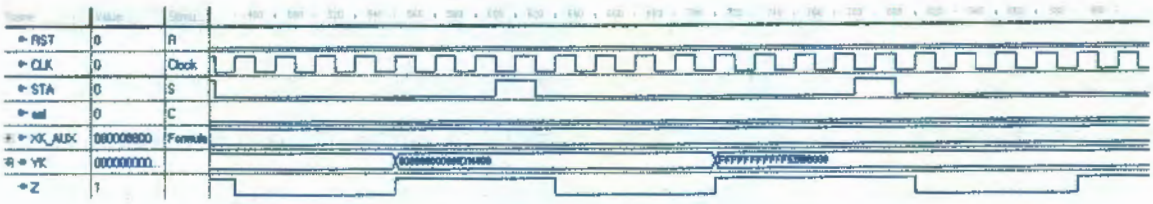


Figura 4.4 Iteración para K=3 y K=4

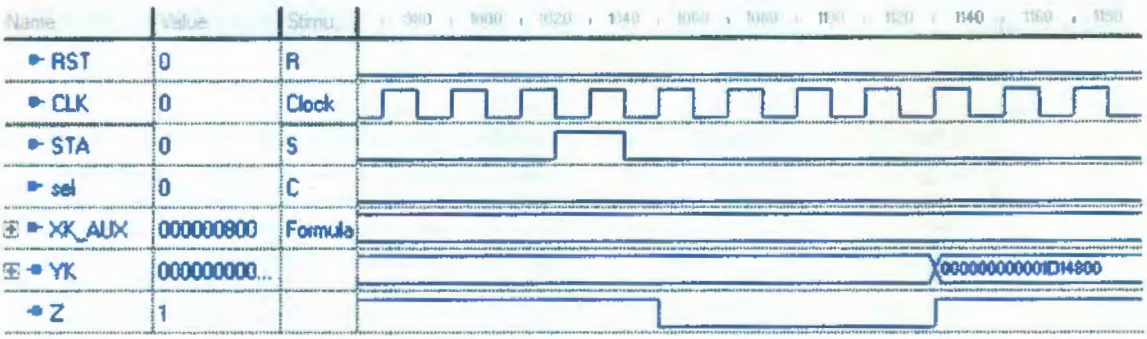


Figura 4.5 Iteración para K=5

4.5 Simulación de estructura digital PWM

En la figura 4.6 aparece el diagrama de ondas para la simulación del modulador de ancho de pulso digital, se muestran cuatro tiempos de ancho de pulso, 250 µs, 500 µs, 750 µs y 1 ms que son respectivamente los ciclos de trabajo 25%, 50%, 75% y 100%.

Figura 4.6 Forma de ondas para simulación del bloque "PWM".

4.6 Simulación de la interfaz

4.6.1 ADC

A continuación se muestra un ciclo de captura del convertidor ADC para una entrada análoga de 2.5 V, esto es, 800 en hexadecimal. El diagrama de ondas aparece en la figura 4.7, que corresponde a la forma de ondas de la figura 3.15. Al levantarse la entrada Inicio, se inicia el ciclo de adquisición de

señal, durante el tercer flanco negativo de DLCK aparece el primer bit de conversión que es el MSB, el sistema toma los datos restantes durante los flancos positivos de DCLK, y finalmente durante el último de estos flancos la conversión 800 aparece en "Data_c", otro ciclo se inicia al bajar y subir la entrada de Inicio. En la figura 4.8 se repite el diagrama de ondas de la figura 3.15 de la hoja de datos para fines de comparación con las generadas mediante VHDL.

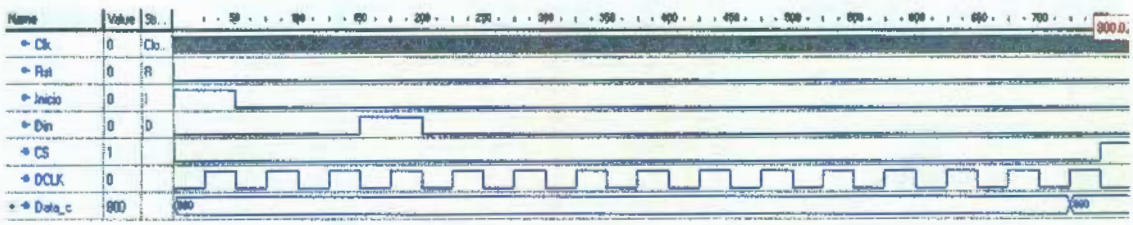


Figura 4.7 Diagrama de ondas para simulación del bloque "ADS7816ADC".

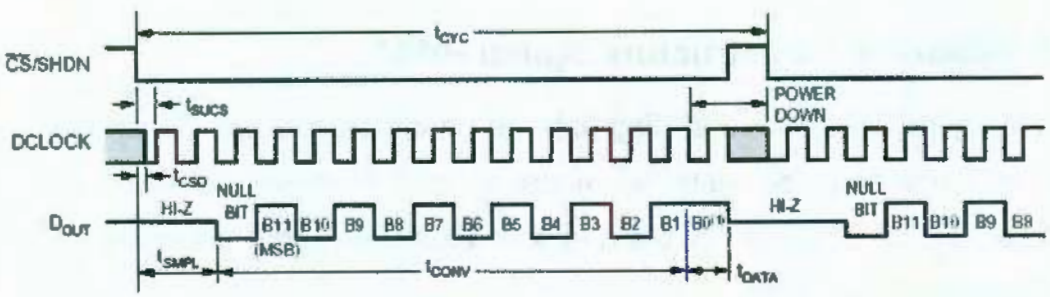


Figura 4.8 Diagrama de tiempo básico para el ADS7816.

4.6.2 LCD

En la figura 4.9 se muestra el diagrama de ondas obtenido del diseño de la sección 3.5.2, es básicamente el diagrama de ondas de la figura 3.18, para las señales RS, RW, E y la palabra de 8 bits. Se observa mientras RS esta en bajo los tres comandos para inicialización de la pantalla, cuando RS conmuta a alto son enviados los códigos para la palabra "Referencia", por razones de espacio solo aparece hasta la segunda letra "e" que corresponde a un "65" en hexadecimal.

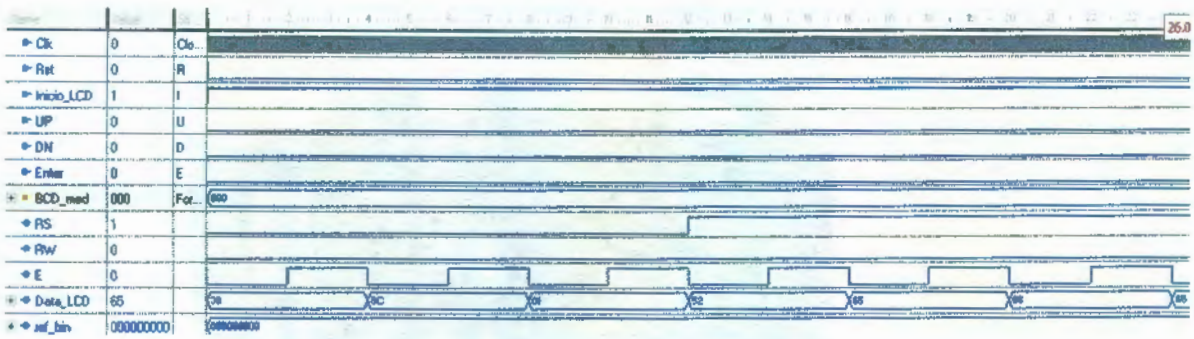


Figura 4.9 Diagrama de ondas para simulación del bloque "manejo LCD".

4.7 Prototipo

Con el objetivo de que el prototipo sea similar a uno manufacturado comercialmente, además de los botones de UP, DOWN y ENTER para el ajuste de la referencia, la pantalla con el desplegado de ésta última y la temperatura en tiempo real, fueron añadidas terminales para la conexión del termopar tipo J y cables de conexión hacia los relevadores de estado sólido. En el interior están sujetos la tarjeta diseñada en la sección 3.6, una fuente de alimentación de DC y el FPGA. Se incluye también un interruptor general.

En la figura 4.10 aparece una fotografía del prototipo completo, dispone de un cable para la conexión a AC monofásica, para alimentación de la fuente. En la figura 4.11 se observan los botones de ajuste y el display LCD. Las terminales de conexión para el termopar tipo J y la salida PWM hacia los conmutadores electrónicos aparecen en la figura 4.12.

La tarjeta diseñada en la sección 3.6 se muestra en la figura 4.13 y en la figura 4.14 aparece el prototipo con todos los elementos en su interior.

Figura 4.10 Prototipo.

Figura 4.11 Interruptores momentáneos y display LCD

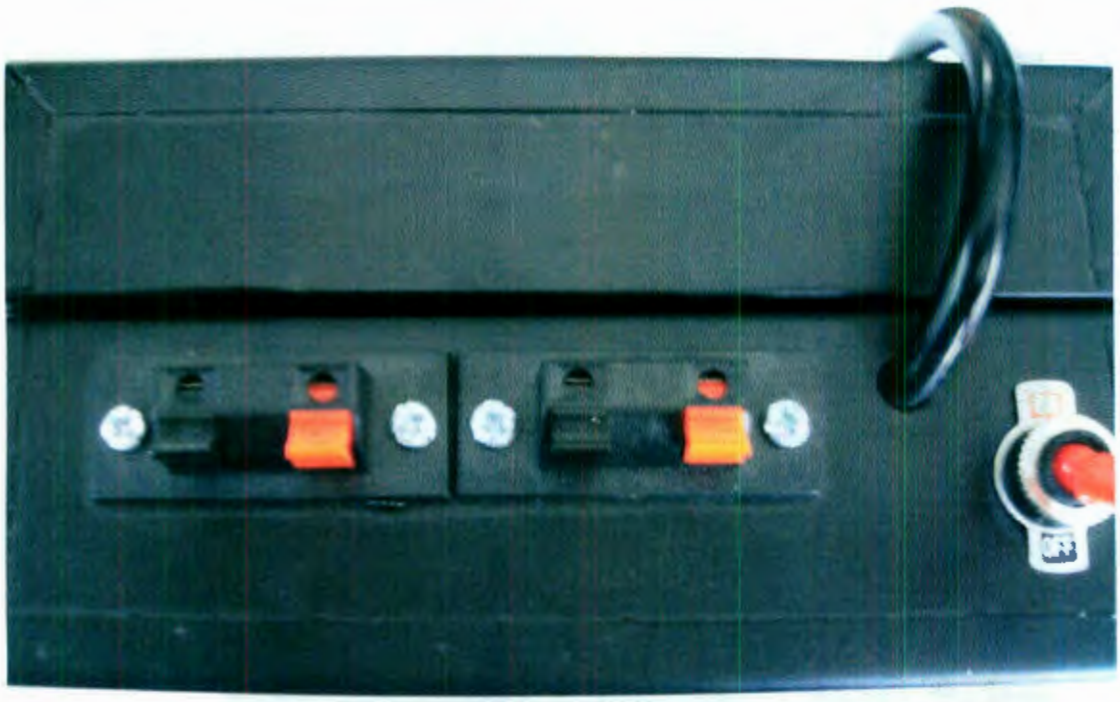


Figura 4.12 Terminales de conexión.

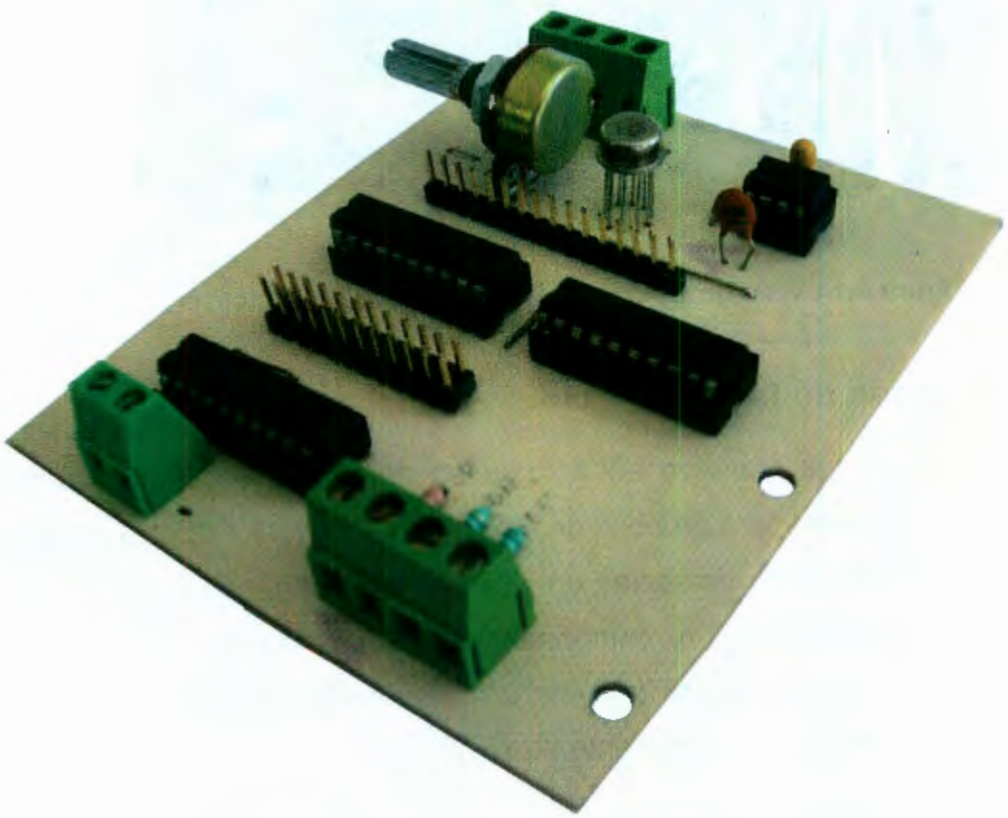


Figura 4.13 Tarjeta para prototipo.

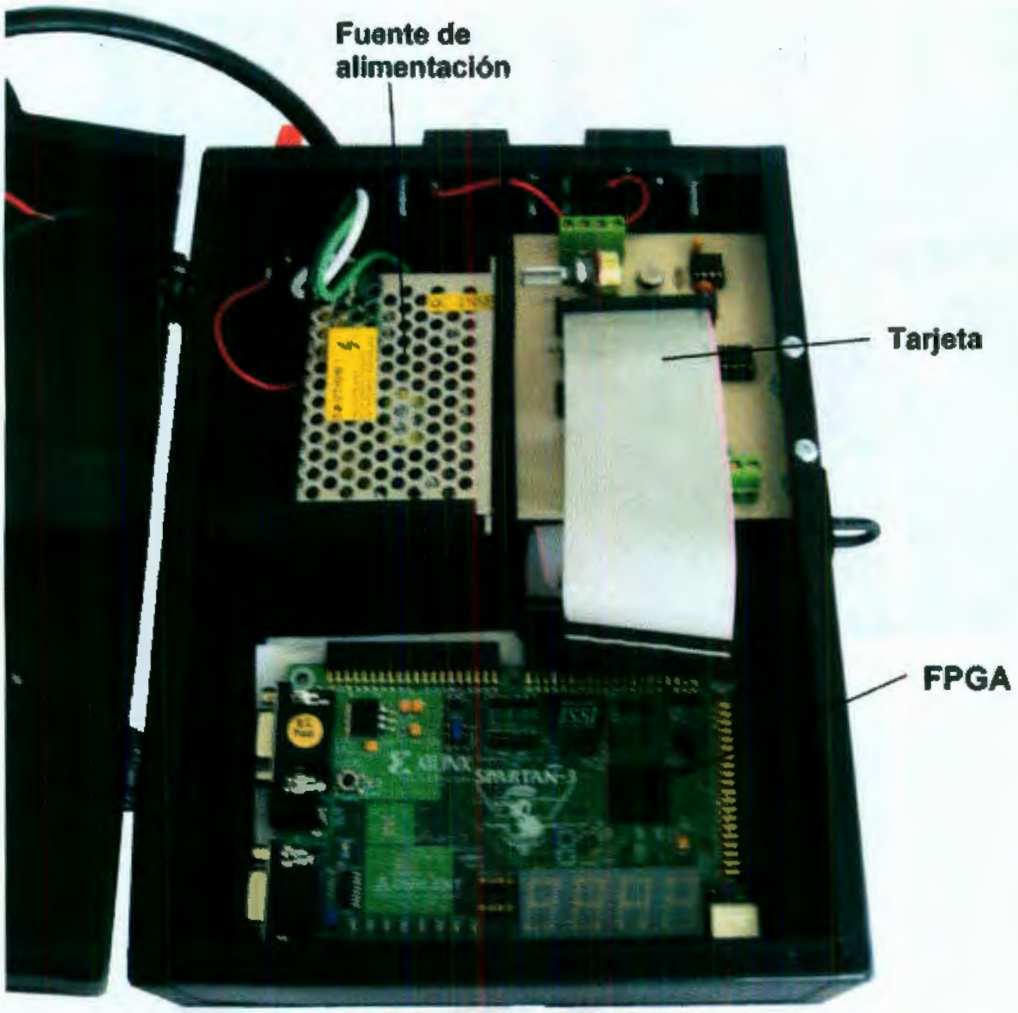


Figura 4.14 Vista de los componentes internos del prototipo.

4.8 Pruebas de funcionalidad

El prototipo fue montado en la máquina de inyección de plástico del laboratorio de Ingeniería para pruebas de funcionalidad (figura 4.15). El sistema se sometió a 4 diferentes referencias en el siguiente orden: 50 °C, 75°C, 150 °C y 100 °C, mostrando un comportamiento satisfactorio, ya que la temperatura de la zona alcanzó las referencias correspondientes y esta se estabilizó con un error máximo aparente de ±5 °C.

Con el fin de comparar la temperatura de referencia para la zona térmica con la registrada por un dispositivo comercial, en las figuras 4.16 y 4.17 se tomó lectura con un termómetro por infarrojo para las temperaturas de 75 °C y 150 °C respectivamente.

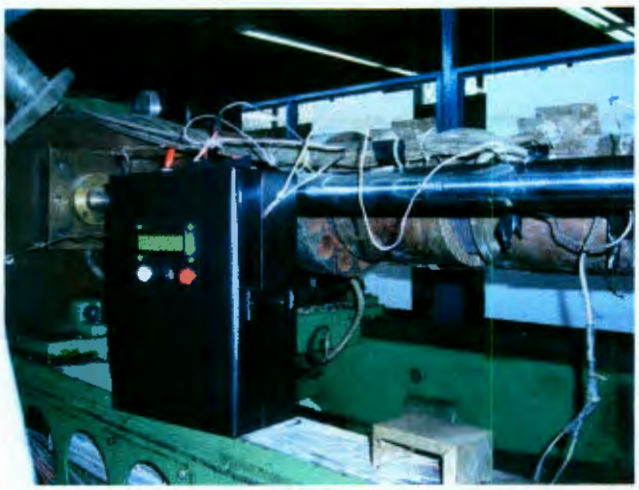


Figura 4.15 Prueba del prototipo

Figura 4.16 Medición de zona con termómetro infrarrojo para 75°C.

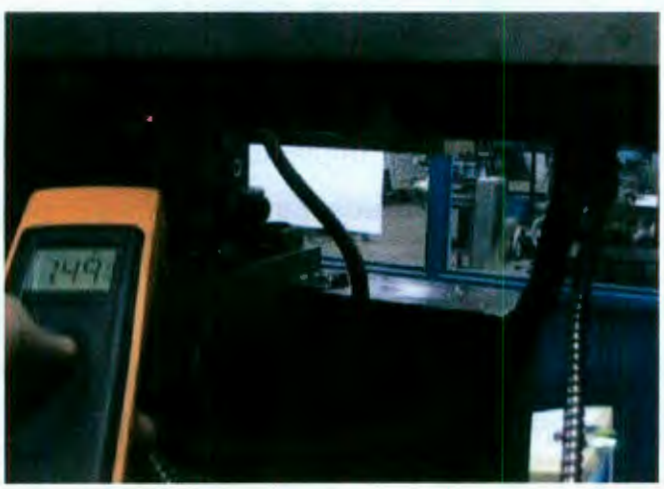


Figura 4.17 Medición de zona con termómetro infrarrojo para 150°C.

CAPÍTULO V

CONCLUSIONES

Un diseño de controlador con aplicación específica, debería ser capaz de lograr la misma funcionalidad pero a un costo mucho menor, comparado con uno adquirido comercialmente, ya que un sistema de control disponible comercialmente es inecesariamente versátil. En éste caso el sistema a controlar fueron las resistencias que funden el plástico a ser inyectado en moldes en una máquina de inyección de plástico, es posible de alguna forma modernizar este tipo de máquinas con controladores en sus diferentes procesos que sean mas eficientes y actuales, además de que se evita adquirir tecnología extranjera, por lo anterior, este proyecto contribuye al control en la parte de temperatura permitiendo la posibilidad de reducir el costo de actualización de la máquina.

El prototipo fue construido utilizando componentes estándar de relativa fácil adquisición en tiendas de componentes electrónicos. Otra ventaja del prototipo y de la lógica reconfigurable empleada es que permite controlar diferentes sistemas de temperatura (teniendo como sensor termopares tipo J) con sólo cambiar los coeficientes de la ecuación en diferencias del algoritmo de control (ecuación 69) ya que estos últimos dependen de la identificación y sintonización del sistema a controlar.

El diseño del controlador esta implementado en FPGA permitiendo modularidad, esto significa que puede adaptarse a otros sistemas de control siendo esto posible al modificar la señal de entrada al sistema (retroalimentación) o de salida, en la mayoría de los casos el convertidor analógico digital seguiría siendo de utilidad ya que la mayoría de los transductores entregan voltajes, y en el caso de la señal de salida, si un control por PWM no fuese necesario, la salida del controlador se puede modificar para adaptarse a otros actuadores como por ejemplo con un DAC.

Referencias

Ogata, Katsuhiko. 2003. Ingeniería de control moderna. Pearson. Cuarta Edición.

Enríquez González , Edgar. 2006. Sistema de control de temperatura aplicado a máquina de inyección de plástico. Universidad Autónoma de Querétaro.

Contreras López, Joaquín. 2006. Sistema de monitoreo y control de temperatura en el edificio de aulas de posgrado de la facultad de Ingeniería de la UAQ. Universidad Autónoma de Querétaro.

Burgos Flores, Martin Luis. 2004. Identificación y control en tiempo real de un horno eléctrico. Universidad Autónoma de Querétaro.

Anaya Mario, González Trejo José Leonides, Guerra Álvarez Rubén. 1998. Medición de Temperatura. Universidad Autónoma de Querétaro.

A.R. San Vicente, J.A. Acosta. 2001. Diseño y construcción de un controlador de temperatura para incubadora. Instituto Tecnológico y de Estudios Superiores de Monterrey Campus Estado de México.

JA. Hernández, S Borromeo, R. de la Prieta, D Nevado, C. Rodríguez. 2003. Una aplicación de Ingeniería biomédica basada en microcontroladores: control de temperatura en ratas de laboratorio para experimentación quirúrgica. Departamento de Ingeniería Telemática y Tecnológica Electrónica. Universidad Rey Juan Carlos. Madrid.

Mendoza Joshua, Ornelas Gerardo, Parra Joel, Guerrero Héctor, Herrera Gilberto, Castañeda Rodrigo. 2007. Instrumentación Climatológica aplicada a granjas de producción animal. Universidad Autónoma de Querétaro.

Osornio-Rios Roque Alfredo, Romero-Troncoso René de Jesús, Herrera-Ruiz Gilberto, Castañeda-Miranda Rodrigo. 2008. Implementación en FPGA de perfiles de aceleración con polinomios de orden superior para la reducción jerk pico en servomotores. Robotics *and Computer-Integrated Manufacturing, Volume 25, Issue 2, April 2009, Pages 379-392*

Avendaño José, Chavez José, Rivas Edgar. Medidor digital de corriente alterna implementado en FPGA. 2007. Universidad Autónoma de Querétaro.

Creus Sole Antonio. 2005. Instrumentación Industrial. Alfaomega. Séptima edición.

Navarro Rina. 2004. Ingeniería de Control Analógica y Digital. Mc. Graw-Hill.

www.omega.com/prodinfo/TemperatureControllers.html

Dorsey John. 2005. Sistemas de control continuos y discretos. Mc. Graw-Hill.

C. Kuo, Benjamin. 2007. Sistemas de control digital. Grupo editorial Patria.

Ogata, Katsuhiko. 1996. Sistemas de control en tiempo discreto. Pearson. Segunda edición.

Bolton, W. 2001. Ingeniería de Control. Alfaomega. Segunda edición.

Osornio Ríos, Roque Alfredo. 2007. Diseño de control CNC de alta velocidad. Universidad Autónoma de Querétaro.

Brown Stephen, Vranesic Svonko. 2006. Fundamentos de lógica digital con diseño VHDL. Mc. Graw-Hill. Segunda edición.

G. Maxinez David, Alcalá Jessica. 2007. VHDL El arte de programar sistemas digitales. CECSA.

J. Maloney, Timothy. 2006. Electrónica industrial moderna. Pearson. Quinta edición.

Bolton, W. 2003. Mechatronics, Electronic controls systems in Mechanical and Electrical Engineering. Pearson. Third Edition.

Manuel Torres, Miguel A. Torres. 2005. Diseño e Ingeniería Electrónica asistida con Protel DXP. Alfaomega.

Apéndice

A.1 Listados de módulo "PID".

A.1.1 Integración de bloques en módulo PID.

```
library IEEE;
use IEEE.std_logic_1164.all;
entity PID is
        port(
  RST: in std_logic;
  CLK: in std_logic;
  STA: in std_logic;
  sel: in std_logic;
  XK_AUX: in std_logic_vector(35 downto 0);
        YK: out std_logic_vector(71 downto 0);
  Z: out std_logic
);
end PID;
architecture PID of PID is
Component Coeficientes_PID_New
port(
sel: in std_logic;
b1,b2: out std_logic_vector(35 downto 0);
a0,a1,a2: out std_logic_vector(35 downto 0)
);
end Component;
Component mux_coeficientes_pid
port(
B1,B2: in std_logic_vector(35 downto 0);
A0,A1,A2: in std_logic_vector(35 downto 0);
IDX: in std_logic_vector(2 downto 0);
B: out std_logic_vector(35 downto 0)
);
end Component;
Component Coeficientes_muestra
port(
    RST : in  std_logic;
    CLK : in  std_logic;
    XK  : in  std_logic_vector(35 downto 0);
    YF  : in  std_logic_vector(35 downto 0);
          RDY : in  std_logic;
          IDX : in  std_logic_vector(2  downto 0);
          STA: in std_logic;
    Y   : out std_logic_vector(35 downto 0)
           );
end Component;
Component MAC_NEW
          port(
    RST : in  std_logic;
    CLK : in  std_logic;
    STA : in std_logic;
    A : in std_logic_vector(35 downto 0);
          B : in std_logic_vector(35 downto 0);
    R : out std_logic_vector(71 downto 0);
    DEC: out std_logic;
          RDY: out std_logic
          );
end Component;
Component indice
  port(
    RST : in  std_logic;
    CLK : in  std_logic;
    DEC : in  std_logic;
    IDX : out std_logic_vector(2 downto 0);
```

```
    Z  : out std_logic
    );
end Component;
Signal A_AUX,YF_AUX,B_aux,A0_AUX,A1_AUX,A2_AUX,B1_AUX,B2_AUX: std_logic_vector(35 downto 0);
Signal IDX_AUX: std_logic_vector(2 downto 0);
Signal Y_LEDS: std_logic_vector(11 downto 0);
Signal RDY_AUX,DEC_AUX: std_logic;
Signal R_AUX: std_logic_vector(71 downto 0);
Signal R_LIMITA: std_logic_vector(48 downto 0);
begin
Pid_New_01: Coeficientes_PID_New port map(sel,B1_AUX,B2_AUX,A0_AUX,A1_AUX,A2_AUX);
Pid_New_02:       mux_coeficientes_pid                                          port
map(B1_AUX,B2_AUX,A0_AUX,A1_AUX,A2_AUX,IDX_AUX,B_AUX);
Pid_New_03:       Coeficientes_muestra                                          port
map(RST,CLK,XK_AUX,YF_AUX,RDY_AUX,IDX_AUX,STA,A_AUX);
Pid_New_04:       MAC_NEW port map(RST,CLK,STA,A_AUX,B_AUX,R_AUX,DEC_AUX,RDY_AUX);
Pid_New_05:       indice port          map(RST,CLK,DEC_AUX,IDX_AUX,Z);
YF_AUX<= R_AUX(46 DOWNTO 11);
YK<=R_aux;
end PID;
```

A.1.2 Listado Bloques "MAC_NEW".

```
library IEEE;
use IEEE.std_logic_1164.all;
entity MAC_NEW is
          port(
    RST : in  std_logic;
    CLK : in  std_logic;
    STA : in std_logic;
    A : in std_logic_vector(35 downto 0);
          B : in std_logic_vector(35 downto 0);
    R : out std_logic_vector(71 downto 0);
    DEC: out std_logic;
          RDY: out std_logic
          );
end MAC_NEW;
architecture MAC_NEW of MAC_NEW is
Component acumulador_72
port(
    RST : in  std_logic;
    CLK : in  std_logic;
    S   : in  std_logic_vector(71 downto 0);
    OPA : in  std_logic;
          ACC : out std_logic_vector(71 downto 0)
    );
end Component;
Component resultado_71
          port(
    RST : in  std_logic;
    CLK : in  std_logic;
    S   : in  std_logic_vector(71 downto 0);
    OPR : in  std_logic;
    R   : out std_logic_vector(71 downto 0)
    );
end Component;
Component multiplica_36
          port(
          X: in std_logic_vector(35 downto 0);
          Y: in std_logic_vector(35 downto 0);
          M: out std_logic_vector(71 downto 0)
          );
end Component;
Component sumador_72
          port(
          A: in std_logic_vector(71 downto 0);
          P: in std_logic_vector(71 downto 0);
```

```
        S: out std_logic_vector(71 downto 0)
        );
end Component;
Component FSM_MACN
        port(
    RST : in  std_logic;
    CLK : in  std_logic;
    STA : in  std_logic;
    OPA : out std_logic;
    OPR : out std_logic;
    DEC : out std_logic;
    RDY : out std_logic
    );
end Component;
signal M_AUX,S_AUX,ACC_AUX: std_logic_vector(71 downto 0);
Signal OPA_AUX,OPR_AUX: std_logic;
 begin
MAC_NEW_01: multiplica_36 port map(A,B,M_AUX);
MAC_NEW_02:   sumador_72 port   map(M_AUX,ACC_AUX,S_AUX);
MAC_NEW_03:   acumulador_72 port map(RST,CLK,S_AUX,OPA_AUX,ACC_AUX);
MAC_NEW_04:   resultado_71 port map(RST,CLK,S_AUX,OPR_AUX,R);
MAC_NEW_05:   FSM_MACN port map(RST,CLK,STA,OPA_AUX,OPR_AUX,DEC,RDY);
end MAC_NEW;


library IEEE;
use IEEE.std_logic_1164.all;
entity acumulador_72 is
port(
    RST : in  std_logic;
    CLK : in  std_logic;
    S   : in  std_logic_vector(71 downto 0);
    OPA : in  std_logic;
          ACC : out std_logic_vector(71 downto 0)
    );
end acumulador_72;
architecture acumulador_72 of acumulador_72 is
signal Ap, An : std_logic_vector(71 downto 0); -- FSM states
begin
 Combinational: process(S,OPA,Ap)
  begin
    if (OPA='1') then
    An <= (others => '0');
                  else
      An <= S;
    end if;
    ACC <= Ap;
  end process Combinational;
  Sequential: process(RST,CLK)
  begin
    if (RST='1') then
      Ap <= (others => '0');
    elsif (CLK'event and CLK='1') then
      Ap <= An;
    end if;
  end process Sequential;
end acumulador_72;


library IEEE;
use IEEE.std_logic_1164.all;
entity resultado_71 is
        port(
    RST : in  std_logic;
    CLK : in  std_logic;
    S   : in  std_logic_vector(71 downto 0);
```

```
    OPR : in  std_logic;
    R   : out std_logic_vector(71 downto 0)
    );
end resultado_71;
architecture resultado_71 of resultado_71 is
        signal Rp, Rn : std_logic_vector(71 downto 0); -- FSM states
begin
 Combinational: process(S,OPR,Rp)
  begin
    if (OPR='0') then
      Rn <= Rp;
    else
      Rn <= S;
    end if;
    R <= Rp;
  end process Combinational;
  Sequential: process(RST,CLK)
  begin
    if (RST='1') then
      Rp <= (others => '0');
    elsif (CLK'event and CLK='1') then
      Rp <= Rn;
    end if;
  end process Sequential;

end resultado_71;

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_ARITH.all;
use IEEE.STD_LOGIC_UNSIGNED.all;
entity multiplica_36 is
        port(
        X: in std_logic_vector(35 downto 0);
        Y: in std_logic_vector(35 downto 0);
        M: out std_logic_vector(71 downto 0)
          );
end multiplica_36;
architecture multiplica_36 of multiplica_36 is
begin
        process(X,Y)
        variable R: signed (71 downto 0);
        begin
                R:=signed(X)*signed(Y);
                M<= std_logic_vector(R);
                end process;
end multiplica_36;

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.STD_LOGIC_ARITH.all;
use IEEE.STD_LOGIC_UNSIGNED.all;
entity sumador_72 is
         port(
        A: in std_logic_vector(71 downto 0);
        P: in std_logic_vector(71 downto 0);
        S: out std_logic_vector(71 downto 0)
        );
end sumador_72;
architecture sumador_72 of sumador_72 is
                begin
          process(A,P)
        variable R: unsigned (71 downto 0);
        begin
                R:= unsigned(A)+ unsigned(P);
                S<=std_logic_vector(R);
```

```
                    end process;
end sumador_72;

library IEEE;
use IEEE.std_logic_1164.all;
entity FSM_MACN is
            port(
    RST : in  std_logic;
    CLK : in  std_logic;
    STA : in  std_logic;
   -- Z   : in  std_logic;
    OPA : out std_logic;
    OPR : out std_logic;
    DEC : out std_logic;
    RDY : out std_logic
    );
end FSM_MACN;
architecture FSM_MACN of FSM_MACN is
signal Qp, Qn : std_logic_vector(2 downto 0); -- FSM states
begin
  Combinational: process(STA,Qp)
  begin
    case Qp is
      -- Estado inicial
      when "000" =>
        if (STA='0') then
          Qn <= Qp;    -- Espera para iniciar
        else
          Qn <= "001"; -- ir al Estado uno
        end if;
        OPA <= '1';    -- Resetea acumulador
        OPR <= '0';    -- Resultado en Espera
        DEC <= '0';    -- Contador de indice en espera
        RDY <= '1';    -- Resultado listo
      -- Inicio de multiplicacion , Estado uno
      when "001" =>
        Qn  <= "010";  -- Ir al estado dos
        OPA <= '0';    -- Cargar el acumulador
        OPR <= '0';    -- Resultado en espera
        DEC <= '1';    -- Contador de indice decrementa
          RDY <= '0';    -- Resultado no Listo
      -- Estado dos
      when "010" =>
        Qn  <= "011";  -- Ir al Estado tres
        OPA <= '0';    -- Cargar acumulador
        OPR <= '0';    -- Resultado en espera
        DEC <= '1';    -- Contador de indice decrementa
        RDY <= '0';    -- Resultado no listo
             when "011" =>
        Qn  <= "100";  -- Ir al Estado tres
        OPA <= '0';    -- Cargar acumulador
        OPR <= '0';    -- Resultado en espera
        DEC <= '1';    -- Contador de indice decrementa
        RDY <= '0';    -- Resultado no listo
                    when "100" =>
        Qn  <= "101";  -- Ir al Estado tres
        OPA <= '0';    -- Cargar acumulador
        OPR <= '0';    -- Resultado en espera
        DEC <= '1';    -- Contador de indice en espera
        RDY <= '0';    -- Resultado no listo
                   when "101" =>
        Qn  <= "110";  -- Ir al Estado tres
        OPA <= '1';    -- Cargar acumulador
        OPR <= '1';    -- Resultado en espera
        DEC <= '1';    -- Contador de indice en espera
        RDY <= '0';    -- Resultado no listo
```

```
                when others =>
        if (STA='1') then
          Qn <= Qp;    -- Espera para desactivar  STA
        else
          Qn <= "000"; -- Ir al estado inicial
        end if;
        OPA <= '1';    -- Resetea el acumulador
        OPR <= '0';    -- Resultado en espera
        DEC <= '0';    -- Contador de indice en espera
        RDY <= '1';    -- Resultado listo
end case;
end process Combinational;
Sequencial: process(RST,CLK)
  begin
    if (RST='1') then
      Qp <= (others => '0');
    elsif (CLK'event and CLK='1') then
      Qp <= Qn;
    end if;
  end process Sequencial;
end FSM_MACN;
```

A.1.3 Listado del bloque "Coeficientes_PID_New".

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;

entity Coeficientes_PID_New is
                port(
sel: in std_logic;
b1,b2: out std_logic_vector(35 downto 0);
a0,a1,a2: out std_logic_vector(35 downto 0)
);
end Coeficientes_PID_New;
architecture Coeficientes_PID_New of Coeficientes_PID_New is
begin
Process(Sel)
begin
        case Sel is
                --Coeficientes correctos
                when '0' =>
b1<="000000000000000000000000000000000000"; --0                --Formato 25.11
b2<="000000000000000000000000100000000000"; --1
a0<="000000000000000000000011101000101001";--7.270184
a1<="111111111111111111111001000100110111";-- -13.848417
a2<="000000000000000000000011010010100000"; --6.578191
--Coeficientes erroneos
When others =>
b1<="000000000000000000000000111111111111";--1                 000000001
b2<="000000000000000000000000111111111111";--0                 000000000
a0<="000000000000000000011000011111111111";--97.1997    000030999
a1<="111111111111111111011001011111111111";-- -159.5996     FFFF1CB34
a2<="000000000000000000001110011111111111";--57.3999        00001CB33
        end case;
        end process;
end Coeficientes_PID_New;
```

A.1.4 Listado del bloque "mux_coeficientes_pid".

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
entity mux_coeficientes_pid is
port(
B1,B2: in std_logic_vector(35 downto 0);
A0,A1,A2: in std_logic_vector(35 downto 0);
IDX: in std_logic_vector(2 downto 0);
B: out std_logic_vector(35 downto 0)
```

```
);
end mux_coeficientes_pid;
architecture mux_coeficientes_pid of mux_coeficientes_pid is
begin
Combinacional: process(IDX,A0,A1,A2,B1,B2)
  begin
case IDX is
      when "100" => B <= B2;  --B2
      when "011" => B <= B1;  --B1
      when "010" => B <= A2;  --A2
      when "001" => B <= A1;  --A1
      when "000" => B <= A0;  --A0
      when others    => B <= (others => '0');
 end case;
 end process Combinacional;
end mux_coeficientes_pid;
```

A.1.5 Listado de bloques "Coeficientes_muestra".

```
library IEEE;
use IEEE.std_logic_1164.all;
entity Coeficientes_muestra is
port(
    RST : in  std_logic;
    CLK : in  std_logic;
    XK  : in  std_logic_vector(35 downto 0);
    YF  : in  std_logic_vector(35 downto 0);
          RDY : in  std_logic;
          IDX : in  std_logic_vector(2  downto 0);
          STA: in std_logic;
    Y   : out std_logic_vector(35 downto 0)
           );
end Coeficientes_muestra;
architecture Coeficientes_muestra of Coeficientes_muestra is
  Component Registro_muestra
                port(
                rst:in std_logic;
                clk:IN STD_logic;
                Datain:IN STD_LOGIC_VECTOR(35 downto 0);
                dataout:out std_logic_vector(35 downto 0);
                en_reg:in std_logic
                );
end Component;
Component Maquina_muestra
          port(
          RST : in  std_logic;
   CLK : in  std_logic;
          RDY : in std_logic;
          STA: in std_logic;
          en_reg: out std_logic
          );
end Component;
signal enable :std_logic;
signal X1 : std_logic_vector(35 downto 0); -- X(k-1)
signal X2 : std_logic_vector(35 downto 0); -- X(k-2)
signal Y1 : std_logic_vector(35 downto 0); -- Y(k-1)
signal Y2 : std_logic_vector(35 downto 0); -- Y(k-2)
begin
          --MAQUINA DE ESTADOS
FSM_01:  Maquina_muestra  port map(RST,CLK,RDY,STA,enable);
--REGISTRO 01
REG_O1:          Registro_muestra   port map(RST,CLK,XK,X1,enable);
--REGISTRO 02
REG_O2:          Registro_muestra   port map(RST,CLK,X1,X2,enable);
--REGISTRO 03
REG_O3:          Registro_muestra   port map(RST,CLK,YF,Y1,enable);
--REGISTRO 04
```

```
REG_O4:          Registro_muestra   port map(RST,CLK,Y1,Y2,enable);
--salidas del multiplexor
Combinational: process(XK,IDX,X1,X2,Y1,Y2)
  begin
case IDX is
      when "100" => Y <= Y2;
      when "011" => Y <= Y1;
      when "010" => Y <= X2;
      when "001" => Y <= X1;
      when "000" => Y <= XK;
      when others    => Y <= (others => '0');
 end case;

 end process Combinational;
end Coeficientes_muestra;

library IEEE;
use IEEE.std_logic_1164.all;
entity Maquina_muestra is
          port(
          RST : in  std_logic;
   CLK : in  std_logic;
          RDY : in std_logic;
          STA: in std_logic;
          en_reg: out std_logic
           );
end Maquina_muestra;
architecture Maquina_muestra of Maquina_muestra is
 signal Qp, Qn : std_logic_vector(2 downto 0);
begin
          Combinational: process(RDY,STA,Qp)
          begin
          case Qp is
      -- Estado Inicial
      when "000" =>
        if (STA='0') then
          Qn <= Qp;
        else
          Qn <= "001"; -- Ir al Estado Uno
        end if;
        en_reg <= '0';     -- desabilita registro

      -- Estado Uno
      when "001" =>
                              Qn  <= "010";                -- Ir al estado dos
                              en_reg <= '0';     -- deshabilita registro
                    --Estado dos
                    when "010" =>
                    if(RDY='0') then
                              Qn <= Qp;
                              else
                              Qn  <= "011";                -- Ir al Estado tres
          end if;
                              en_reg <= '0';     -- deshabilita registro
                    --Estado tres
           when "011" =>
                              Qn  <= "100";                -- Ir al Estado cuatro
                              en_reg <= '1';     -- habilita registro

                              -- Estado cuatro
      when others  =>
        if (STA='1') then
          Qn <= Qp;     -- Espera para inicio
        else
          Qn <= "000"; -- Ir al Estado inicialGo to state one
        end if;
```

```
        en_reg <= '0';     -- deshabilita registro
    end case;
   end process Combinational;
  Sequential: process(RST,CLK)
   begin
    if (RST='1') then
      Qp <= (others => '0');
    elsif (CLK'event and CLK='1') then
      Qp <= Qn;
    end if;
  end process Sequential;
 end Maquina_muestra;

 library    IEEE;
 use IEEE.std_logic_1164.all;
 entity Registro_Muestra is
  port(
                    rst:in std_logic;
                    clk:IN STD_logic;
                    Datain:IN STD_LOGIC_VECTOR(35 downto 0);
                    dataout:out std_logic_vector(35 downto 0);
                    en_reg:in std_logic
                    );
 end Registro_Muestra;
 architecture Registro_Muestra of Registro_Muestra is
 signal ep_Reg, es_Reg:std_logic_vector(35 downto 0);
 begin
          Combinational:process(ep_Reg,en_reg, Datain)
 begin
          if(en_reg='1') then
                  es_reg <= Datain;    --Cargo DATO
          else
           es_reg <= ep_reg;  --Mantengo dato
          end if;
   dataout<= ep_reg;
 end process combinational;
          Secuencial_registro: process(rst,clk)
 begin
    if (rst='1') then  --reset asincrono
     ep_reg<= (others => '0');
    elsif (clk'event and clk='1') then
     ep_reg<=es_reg;          --Cambio de estados
          end if;
  end process Secuencial_registro;
 end Registro_Muestra;
```

A.1.6 Listado del bloque "índice".

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
entity indice is
   port(
    RST : in  std_logic;
    CLK : in  std_logic;
    DEC : in  std_logic;
    IDX : out std_logic_vector(2 downto 0);
    Z   : out std_logic
    );
end indice;
architecture indice of indice is
                  signal Cp, Cn : std_logic_vector(2 downto 0); -- FSM states
signal Zp    : std_logic;                -- Zero counter
begin
  Combinational: process(DEC,Cp,Zp)
  begin
```

```
    Zp  <= NOT( Cp(2) OR Cp(1) OR Cp(0));
    if (DEC='0') then
      Cn <= Cp;
    else
      if (Zp='0') then
        Cn <= Cp - 1;
      else
        Cn <= "100";
      end if;
    end if;
    IDX <= Cp;
    Z   <= Zp;
  end process Combinational;
  Sequential: process(RST,CLK)
  begin
    if (RST='1') then
      Cp <= (others => '0');
    elsif (CLK'event and CLK='1') then
      Cp <= Cn;
    end if;
  end process Sequential;
end indice;
```

A.2 Listados del módulo "medicion".

A.2.1 Integración de bloques en módulo "medición".

```
library ieee;
use ieee.std_logic_1164.all;
entity medicion is
            port(
            Clk: in std_logic;
            Rst: in std_logic;
            Inicio: in std_logic;
            Din: in std_logic;
            CS: out std_logic;
    DCLK: out std_logic;
            med_BCD: out std_logic_vector(11 downto 0);
            med_bin: out std_logic_vector(35 downto 0)
            );
end medicion;
architecture medicion of medicion is
signal DATA_conv_aux: std_logic_vector(11 downto 0);
signal temp_decimal_aux: std_logic_vector(9 downto 0);
signal med_bin_aux: std_logic_vector(35 downto 0);
component ADS7816ADC is
            port(
            Clk: in std_logic;
            Rst: in std_logic;
            Inicio: in std_logic;
            Din: in std_logic;
            CS,DCLK: out std_logic;
            Data_c: out std_logic_vector(11 downto 0)
            );
end component;
component factor
            port(
            Data_in: in std_logic_vector(11 downto 0);
            temp_decimal: out std_logic_vector(9 downto 0)
            );
end component;
component Bin_BCD_conv
            port(
            Bin: in std_logic_vector(9 downto 0);
            BCD: out std_logic_vector(11 downto 0)
            );
```

```
 end component;
 begin
  MED_1: ADS7816ADC  port map(Clk,Rst,Inicio,Din,CS,DCLK,DATA_conv_aux);
  MED_2: factor      port map(DATA_conv_aux,temp_decimal_aux);
  MED_3: Bin_BCD_conv port map(temp_decimal_aux,med_BCD);
  med_bin_aux(35 downto 23)<=(others=>'0');
  med_bin_aux(22 downto 11)<=DATA_conv_aux;
  med_bin_aux(10 downto 0)<=(others=>'0');
  med_bin<=med_bin_aux;
 end medicion;
```

A.2.2 Listados Bloque "ADS7816ADC".

```
library ieee;
use ieee.std_logic_1164.all;
entity ADS7816ADC is
        port(
        Clk: in std_logic;
        Rst: in std_logic;
        Inicio: in std_logic;
        Din: in std_logic;
        CS,DCLK: out std_logic;
        Data_c: out std_logic_vector(11 downto 0)
        );
end ADS7816ADC;
architecture ADS7816ADC of ADS7816ADC is
signal T_a_aux,T_r_aux,cnt_a_aux,cnt_r_aux,L_sp_aux,L_pp_aux: std_logic;
signal Data_aux: std_logic_vector(11 downto 0);
component t_25_us
        port(
        Clk: in std_logic;
        Rst: in std_logic;
        T_a: in std_logic;
        T_r: out std_logic
        );
end component;
component cnt_to12
        port(
        Clk: in std_logic;
        Rst: in std_logic;
        cnt_a: in std_logic;
        cnt_r: out std_logic
        );
end component;
component FSMADS7816
        port(
        Clk: in std_logic;
        Rst: in std_logic;
        Inicio: in std_logic;
        T_r: in std_logic;
        cnt_r: in std_logic;
        CS,DCLK: out std_logic;
        L_sp,L_pp: out std_logic;
        T_a,cnt_a: out std_logic
        );
end component;
component reg_sp_nbits
        Generic(n:integer:=12);
        port(
        Clk: in std_logic;
        Rst: in std_logic;
        D_in: in std_logic;
        L_sp: in std_logic;
        Data: out std_logic_vector(n-1 downto 0)
        );
end component;
component reg_pp_n
```

```
        Generic(n:integer:=12);
        port(
        Clk: in std_logic;
        Rst: in std_logic;
        D_in: in std_logic_vector(n-1 downto 0);
        L_pp: in std_logic;
        Data: out std_logic_vector(n-1 downto 0)
        );
end component;
begin
        stage0:   t_25_us     port map(Clk,Rst,T_a_aux,T_r_aux);
        stage1: cnt_to12    port map(Clk,Rst,cnt_a_aux,cnt_r_aux);
        stage2:
portmap(Clk,Rst,Inicio,T_r_aux,cnt_r_aux,CS,DCLK,L_sp_aux,L_pp_aux,T_a_aux,cnt_a_aux);
        stage3: reg_sp_nbits port map(Clk,Rst,Din,L_sp_aux,Data_aux);
        stage4: reg_pp_n port map(Clk,Rst,Data_aux,L_pp_aux,Data_c);
end ADS7816ADC;

library   ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity t_25_us is
        port(
        Clk: in std_logic;
        Rst: in std_logic;
        T_a: in std_logic;
        T_r: out std_logic
        );
end t_25_us;
architecture t_25_us of t_25_us is
signal pC,nC: std_logic_vector(10 downto 0);
begin
        process(pC,T_a)
        begin
                case T_a is
                        when '0' =>
                        nC<=(others=>'0');
                        when others =>
                        nC<=pC+1;
                         if(pC="10011100000") then
                                nC<=(others=>'0');
                         else
                                nC<=pC+1;
                         end if;
                end case;
                if(pC="10011100000") then
                        T_r<='1';
                else
                        T_r<='0';
                end if;
        end process;
        process(Clk,Rst)
        begin
                if(Rst='1') then
                        pc<=(others=>'0');
                elsif(Clk' event and clk='1') then
                        pc<=nc;
                end if;
        end process;
end t_25_us;

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
```

FSMADS7816

```
entity cnt_to12 is
	port(
	Clk: in std_logic;
	Rst: in std_logic;
	cnt_a: in std_logic;
	cnt_r: out std_logic
	);
end cnt_to12;
architecture cnt_to12 of cnt_to12 is
signal pC,nC: std_logic_vector(3 downto 0);
begin
	process(pC,cnt_a)
	begin
		case cnt_a is
			when '0' =>
			nC<=pC;
			when others =>
			nC<=pC+1;
			 if pC="1100" then
				nC<=(others=>'0');
			 else
				nC<=pC+1;
			 end if;
		end case;
		if pC="1100" then
			cnt_r<='1';
		else
			cnt_r<='0';
		end if;
	end process;
	process(Clk,Rst)
	begin
		if Rst='1' then
			pc<=(others=>'0');
		elsif Clk'event and Clk='1' then
			pC<=nC;
		end if;
	end process;
end cnt_to12;

library ieee;
use ieee.std_logic_1164.all;
entity FSMADS7816 is
	port(
	Clk: in std_logic;
	Rst: in std_logic;
	Inicio: in std_logic;
	T_r: in std_logic;
	cnt_r: in std_logic;
	CS,DCLK: out std_logic;
	L_sp,L_pp: out std_logic;
	T_a,cnt_a: out std_logic
	);
end FSMADS7816;
architecture FSMADS7816 of FSMADS7816 is
type estados is (s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12);
signal Ep,Es: estados;
begin
	process(Ep,Inicio,T_r,cnt_r)
	begin
		case Ep is
			when s0 => CS<='1';DCLK<='0';L_sp<='0';L_pp<='0';T_a<='0';cnt_a<='0';
			if Inicio='0' then
				Es<=s0;
			else
				Es<=s1;
```

```
end if;
when s1 => CS<='0';DCLK<='0';L_sp<='0';L_pp<='0';T_a<='1';cnt_a<='0';
if T_r='0' then
        Es<=s1;
else
        Es<=s2;
end if;
when s2 => CS<='0';DCLK<='1';L_sp<='0';L_pp<='0';T_a<='1';cnt_a<='0';
if T_r='0' then
        Es<=s2;
else
        Es<=s3;
end if;
when s3 => CS<='0';DCLK<='0';L_sp<='0';L_pp<='0';T_a<='1';cnt_a<='0';
if T_r='0' then
        Es<=s3;
else
        Es<=s4;
end if;
when s4 => CS<='0';DCLK<='1';L_sp<='0';L_pp<='0';T_a<='1';cnt_a<='0';
if T_r='0' then
        Es<=s4;
else
        Es<=s5;
end if;
when s5 => CS<='0';DCLK<='0';L_sp<='0';L_pp<='0';T_a<='1';cnt_a<='0';
if T_r='0' then
        Es<=s5;
else
        Es<=s6;
end if;
when s6 => CS<='0';DCLK<='1';L_sp<='0';L_pp<='0';T_a<='1';cnt_a<='0';
if T_r='0' then
        Es<=s6;
else
        Es<=s7;
end if;
when s7 => CS<='0';DCLK<='0';L_sp<='0';L_pp<='0';T_a<='1';cnt_a<='0';
if T_r='0' then
        Es<=s7;
else
        Es<=s8;
end if;
when s8 => CS<='0';DCLK<='1';L_sp<='1';L_pp<='0';T_a<='1';cnt_a<='1';
Es<=s9;
when s9 => CS<='0';DCLK<='1';L_sp<='0';L_pp<='0';T_a<='1';cnt_a<='0';
Es<=s10;
when s10 => CS<='0';DCLK<='1';L_sp<='0';L_pp<='0';T_a<='1';cnt_a<='0';
if(T_r='0' and cnt_r='0') then
        Es<=s10;
elsif(T_r='0' and cnt_r='1') then
        Es<=s11;
elsif(T_r='1' and cnt_r='0') then
        Es<=s7;
else
        Es<=s11;
end if;
when s11 => CS<='0';DCLK<='1';L_sp<='0';L_pp<='1';T_a<='1';cnt_a<='1';
Es<=s12;
when others => CS<='0';DCLK<='1';L_sp<='0';L_pp<='0';T_a<='1';cnt_a<='0';
if T_r='0' then
        Es<=s12;
else
        Es<=s0;
end if;
end case;
```

```
        end process;
        process(Clk,Rst)
        begin
                if Rst='1' then
                        Ep<=s0;
                elsif Clk' event and Clk='1' then
                        Ep<=Es;
                end if;
        end process;
end FSMADS7816;

library ieee;
use ieee.std_logic_1164.all;
entity reg_sp_nbits is
        Generic(n:integer:=12);
        port(
        Clk: in std_logic;
        Rst: in std_logic;
        D_in: in std_logic;
        L_sp: in std_logic;
        Data: out std_logic_vector(n-1 downto 0)
        );
end reg_sp_nbits;
architecture reg_sp_nbits of reg_sp_nbits is
signal Data_aux: std_logic_vector(n-1 downto 0);
begin
        process(Clk,Rst,D_in,L_sp)
        begin
                if Rst='1' then
                        Data_aux<=(others=>'0');
                        Data<=(others=>'0');
                elsif Clk'event and Clk='1' then
                        Data<=Data_aux;
                        if L_sp='1' then
                                Data_aux(0)<=D_in;
                                for i in 1 to n-1 loop
                                        Data_aux(i)<=Data_aux(i-1);
                                end loop;
                        end if;
                        Data<=Data_aux;
                end if;
        end process;
end reg_sp_nbits;

library ieee;
use ieee.std_logic_1164.all;
entity reg_pp_n is
        Generic(n:integer:=12);
        port(
        Clk: in std_logic;
        Rst: in std_logic;
        D_in: in std_logic_vector(n-1 downto 0);
        L_pp: in std_logic;
        Data: out std_logic_vector(n-1 downto 0)
        );
end reg_pp_n;
architecture reg_pp_n of reg_pp_n is
signal pD,sD: std_logic_vector(n-1 downto 0);
begin
        process(D_in,L_pp,pD)
        begin
                if L_pp='0' then
                        sD<=pD;
                else
                        sD<=D_in;
                end if;
```

```
                Data<=pD;
        end process;
        process(Clk,Rst)
        begin
                if Rst='1' then
                        pD<=(others=>'0');
                elsif Clk'event and Clk='1' then
                        pD<=sD;
                end if;
        end process;
end reg_pp_n;
```

A.2.3 Listado bloque "factor".

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity factor is
        port(
        Data_in: in std_logic_vector(11 downto 0);
        temp_decimal: out std_logic_vector(9 downto 0)
        );
end factor;
architecture factor of factor is
signal A: std_logic_vector(21 downto 0);
signal B: std_logic_vector(21 downto 0);
signal C: std_logic_vector(43 downto 0);
begin
        A(21 downto 10)<=Data_in;       A(9 downto 0)<="0000000000";
        B(21 downto 10)<="000000000000"; B(9 downto 0)<="0001111101";
        process(A,B)
        variable R: unsigned (43 downto 0);
        begin
                R:= unsigned(A)*unsigned(B);
                C<= std_logic_vector(R);
        end process;
        temp_decimal<=C(29 downto 20);
end factor;
```

A.2.4 Listado bloque "Bin_BCD_conv".

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
entity Bin_BCD_conv is
        port(
        Bin: in std_logic_vector(9 downto 0);
        BCD: out std_logic_vector(11 downto 0)
        );
end Bin_BCD_conv;
architecture Bin_BCD_conv of Bin_BCD_conv is
begin
process(Bin)
 variable z: STD_LOGIC_VECTOR (22 downto 0);
 begin
        for i in 0 to 21 loop
                z(i) := '0';
        end loop;
                z(12 downto 3) := Bin;

        for i in 0 to 6 loop
                if z(13 downto 10) > 4 then
                        z(13 downto 10) := z(13 downto 10) + 3;
                end if;
                if z(17 downto 14) > 4 then
                        z(17 downto 14) := z(17 downto 14) + 3;
```

```
                end if;
                if z(21 downto 18) > 4 then
                        z(21 downto 18) := z(21 downto 18) + 3;
                end if;
                z(22 downto 1) := z(21 downto 0);
        end loop;
        BCD <= z(21 downto 10);
end process;
end Bin_BCD_conv;
```

A.3 Listados de modulo "manejo_LCD".

A.3.1 Integración de bloques de módulo "manejo_LCD".

```
library ieee;
use ieee.std_logic_1164.all;
entity manejo_LCD is
        port(
        Clk: in std_logic;
        Rst: in std_logic;
        Inicio_LCD: in std_logic;
        UP,DN,Enter: in std_logic;
        BCD_med: in std_logic_vector(11 downto 0);
        RS,RW,E: out std_logic;
        Data_LCD: out std_logic_vector(7 downto 0);
        ref_bin: out std_logic_vector(35 downto 0)
        );
end manejo_LCD;
architecture manejo_LCD of manejo_LCD is
signal Inicia_i_aux,t_ri_aux,RSi_aux,RWi_aux,Ei_aux,t_ai_aux,fin_i_aux: std_logic;
signal Data_i_aux,caracter_aux,Data_e_aux: std_logic_vector(7downto 0);
signal Inicia_e_aux,t_re_aux,RSe_aux,RWe_aux,Ee_aux,t_ae_aux,fin_e_aux,sel_lcdmux_aux: std_logic;
signal sel_car_aux: std_logic_vector(6 downto 0);
signal BCD_ref_aux: std_logic_vector(11 downto 0);
signal t_8Hz_aux,UP_aux,DN_aux: std_logic;
signal cnt10_aux: std_logic_vector(9 downto 0);
signal ref_bin_aux: std_logic_vector(35 downto 0);
component FSM_LCD_inicia
        port(
        Clk: in std_logic;
        Rst: in std_logic;
        Inicia_i: in std_logic;
        t_r: in std_logic;
        RS,RW,E: out std_logic;
        Data_i: out std_logic_vector(7 downto 0);
        t_a: out std_logic;
        fin_i: out std_logic
        );
end component;
component T_2ms
        port(
        Clk: in std_logic;
        Rst: in std_logic;
        t_2ms_i: in std_logic;
        t_2ms_f: out std_logic
        );
end component;
component FSM_lcd_escritura
        port(
        Clk: in std_logic;
        Rst: in std_logic;
        Inicia_e: in std_logic;
        caracter: in std_logic_vector(7 downto 0);
        t_r: in std_logic;
        RS,RW,E: out std_logic;
        Data_e: out std_logic_vector(7 downto 0);
```

```
        t_a,fin_e: out std_logic
        );
end component;
component FSM_LCD_maestra
        port(
        Clk: in std_logic;
        Rst: in std_logic;
        Inicia_lcd: in std_logic;
        fin_i: in std_logic;
        Inicia_i: out std_logic;
        Inicia_e: out std_logic;
        sel_lcd_mux: out std_logic
        );
end component;
component sel_cnt
        port(
        Clk:    in std_logic;
        Rst:    in std_logic;
        Fin_e:  in std_logic;
        selector: out std_logic_vector(6 downto 0)
        );
end component;
component mux_caracter
        port(
        selector: in  std_logic_vector(6 downto 0);
        BCD_ref: in std_logic_vector(11 downto 0);
        BCD_med: in std_logic_vector(11 downto 0);
        caracter: out       std_logic_vector(7 downto 0)
        );
end component;
component LCD_mux
        port(
        sel_mux_lcd: in std_logic;
        RS_i,RW_i,E_i: in std_logic;
        Data_i: in std_logic_vector(7 downto 0);
        RS_e,RW_e,E_e: in std_logic;
        Data_e: in std_logic_vector(7 downto 0);
        RS,RW,E: out std_logic;
        Data_LCD: out std_logic_vector(7 downto 0)
        );
end component;
component t_8Hz
        port(
        Clk: in std_logic;
        Rst: in std_logic;
        T_out: out std_logic
        );
end component;
component AND_gate
        port(
        A: in std_logic;
        B: in std_logic;
        C: out std_logic
        );
end component;
component cont_10bits
        port(
        Clk: in std_logic;
        Rst: in std_logic;
        EN_UP: in std_logic;
        EN_DN: in std_logic;
        Cuenta: out std_logic_vector(9 downto 0)
        );
end component;
component Bin_BCD_conv
        port(
```

```
        Bin: in std_logic_vector(9 downto 0);
        BCD: out std_logic_vector(11 downto 0)
        );
end component;
component factor_ref
        port(
        ref_bin: in std_logic_vector(9 downto 0);
        ref_factor: out std_logic_vector(35 downto 0)     --25.11
        );
end component;
component reg_pp_36
        Generic(n:integer:=36);
        port(
        Clk: in std_logic;
        Rst: in std_logic;
        D_in: in std_logic_vector(n-1 downto 0);
        L_pp: in std_logic;
        Data: out std_logic_vector(n-1 downto 0)
        );
end component;
begin
        LCD0:                                   FSM_LCD_inicia                  port
map(Clk,Rst,Inicia_i_aux,t_ri_aux,RSi_aux,RWi_aux,Ei_aux,Data_i_aux,t_ai_aux,fin_i_aux);
        LCD1: T_2ms port map(Clk,Rst,t_ai_aux,t_ri_aux);
        LCD3:                                   FSM_lcd_escritura               port
map(Clk,Rst,Inicia_e_aux,caracter_aux,t_re_aux,RSe_aux,RWe_aux,Ee_aux,Data_e_aux,t_ae_aux,fin_e_aux);
        LCD4: T_2ms port map(Clk,Rst,t_ae_aux,t_re_aux);
        LCD5:                                   FSM_LCD_maestra                 port
map(Clk,Rst,Inicio_LCD,fin_i_aux,Inicia_i_aux,Inicia_e_aux,sel_lcdmux_aux);
        LCD6: sel_cnt port map(Clk,Rst,fin_e_aux,sel_car_aux);
        LCD7: mux_caracter port map(sel_car_aux,BCD_ref_aux,BCD_med,caracter_aux);
        LCD8:                                   LCD_mux                         port
map(sel_lcdmux_aux,RSi_aux,RWi_aux,Ei_aux,Data_i_aux,RSe_aux,RWe_aux,Ee_aux,Data_e_aux,RS,RW,E,Data
_LCD);
        ref0: t_8Hz port map(Clk,Rst,t_8Hz_aux);
        ref1: AND_gate port map(t_8Hz_aux,UP,UP_aux);
        ref2: AND_gate port map(t_8Hz_aux,DN,DN_aux);
        ref3: cont_10bits port map(Clk,Rst,UP_aux,DN_aux,cnt10_aux);
        ref4: Bin_BCD_conv port map(cnt10_aux,BCD_ref_aux);
        ref5: factor_ref port map(cnt10_aux,ref_bin_aux);
        ref6: reg_pp_36 port map(Clk,Rst,ref_bin_aux,Enter,ref_bin);
end manejo_LCD;
```

A.3.2 Listado bloque "FSM_LCD_inicia".

```
library ieee;
use ieee.std_logic_1164.all;
entity FSM_LCD_inicia is
        port(
        Clk: in std_logic;
        Rst: in std_logic;
        Inicia_i: in std_logic;
        t_r: in std_logic;
        RS,RW,E: out std_logic;
        Data_i: out std_logic_vector(7 downto 0);
        t_a: out std_logic;
        fin_i: out std_logic
        );
end FSM_LCD_inicia;
architecture FSM_LCD_inicia of FSM_LCD_inicia is
type estados is (s0,s1,s2,s3,s4,s5,s6,s7,s8,s9);
signal Ep,Es: estados;
begin
        process(Inicia_i,t_r,Ep)
        begin
                case Ep is
                when s0 => RS<='0';RW<='0';E<='1';Data_i<="00111000";t_a<='0';fin_i<='0';
```

```
                        if Inicia_i='0' then
                                Es<=s0;
                        else
                                Es<=s1;
                        end if;
                        when s1 => RS<='0';RW<='0';E<='0';Data_i<="00111000";t_a<='1';fin_i<='0';
                        if t_r='0' then
                                Es<=s1;
                        else
                                Es<=s2;
                        end if;
                        when s2 => RS<='0';RW<='0';E<='1';Data_i<="00111000";t_a<='1';fin_i<='0';
                        if t_r='0' then
                                Es<=s2;
                        else
                                Es<=s3;
                        end if;
                        when s3 => RS<='0';RW<='0';E<='0';Data_i<="00111000";t_a<='0';fin_i<='0';
                                Es<=s4;
                        when s4 => RS<='0';RW<='0';E<='0';Data_i<="00001100";t_a<='1';fin_i<='0';
                        if t_r='0' then
                                Es<=s4;
                        else
                                Es<=s5;
                        end if;
                        when s5 => RS<='0';RW<='0';E<='1';Data_i<="00001100";t_a<='1';fin_i<='0';
                        if t_r='0' then
                                Es<=s5;
                        else
                                Es<=s6;
                        end if;
                        when s6 => RS<='0';RW<='0';E<='0';Data_i<="00001100";t_a<='0';fin_i<='0';
                                Es<=s7;
                        when s7 => RS<='0';RW<='0';E<='0';Data_i<="00000001";t_a<='1';fin_i<='0';
                        if t_r='0' then
                                Es<=s7;
                        else
                                Es<=s8;
                        end if;
                        when s8 => RS<='0';RW<='0';E<='1';Data_i<="00000001";t_a<='1';fin_i<='0';
                        if t_r='0' then
                                Es<=s8;
                        else
                                Es<=s9;
                        end if;
                        when others => RS<='0';RW<='0';E<='0';Data_i<="00000001";t_a<='0';fin_i<='1';
                        Es<=s0;
                end case;
        end process;
        process(Clk,Rst)
        begin
                if Rst='1' then
                        Ep<=s0;
                elsif Clk'event and Clk='1' then
                        Ep<=Es;
                end if;
        end process;
end FSM_LCD_inicia;
```

A.3.3 Listado bloque "T_2ms".

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity T_2ms is
        port(
```

```
        Clk: in std_logic;
        Rst: in std_logic;
        t_2ms_i: in std_logic;
        t_2ms_f: out std_logic
        );
end T_2ms;

architecture T_2ms of T_2ms is
signal Cp,Cs: std_logic_vector(16 downto 0);
begin
        process(Cp,t_2ms_i)
        begin
                case t_2ms_i is
                        when '0' =>
                        Cs<=(others=>'0');
                        when others =>
                        Cs<=Cs+1;
                         if(Cs="11000011010011111") then
                                Cs<=(others=>'0');
                         else
                                Cs<=Cp+1;
                         end if;
                end case;
                if Cp="11000011010011111" then
                        t_2ms_f<='1';
                else
                        t_2ms_f<='0';
                end if;
        end process;
        process(Clk,Rst)
        begin
                if Rst='1' then
                        Cp<=(others=>'0');
                elsif Clk' event and Clk='1' then
                        Cp<=Cs;
                end if;
        end process;
end T_2ms;
```

A.3.4 Listado bloque "FSM_lcd_escritura".

```
library ieee;
use ieee.std_logic_1164.all;
entity FSM_lcd_escritura is
        port(
        Clk: in std_logic;
        Rst: in std_logic;
        Inicia_e: in std_logic;
        caracter: in std_logic_vector(7 downto 0);
        t_r: in std_logic;
        RS,RW,E: out std_logic;
        Data_e: out std_logic_vector(7 downto 0);
        t_a,fin_e: out std_logic
        );
end FSM_lcd_escritura;
architecture FSM_lcd_escritura of FSM_lcd_escritura is
type estados is (s0,s1,s2,s3);
signal Ep,Es: estados;
begin
        process(Ep,Inicia_e,caracter,t_r)
        begin
        case Ep is
                when s0 => RS<='1';RW<='0';E<='0';Data_e<=caracter;t_a<='0';fin_e<='0';
                if Inicia_e='0' then
                        Es<=s0;
                else
                        Es<=s1;
```

```
                end if;
                when s1 => RS<='1';RW<='0';E<='0';Data_e<=caracter;t_a<='1';fin_e<='0';
                if t_r='0' then
                        Es<=s1;
                else
                        Es<=s2;
                end if;
                when s2 => RS<='1';RW<='0';E<='1';Data_e<=caracter;t_a<='1';fin_e<='0';
                if t_r='0' then
                        Es<=s2;
                else
                        Es<=s3;
                end if;
                when s3 => RS<='1';RW<='0';E<='0';Data_e<=caracter;t_a<='0';fin_e<='1';
                Es<=s1;
        end case;
        end process;
        process(Clk,Rst)
        begin
                if Rst='1' then
                        Ep<=s0;
                elsif Clk'event and Clk='1' then
                        Ep<=Es;
                end if;
        end process;
end FSM_lcd_escritura;
```

A.3.5 Listado bloque "FSM_LCD_maestra".

```
library ieee;
use ieee.std_logic_1164.all;
entity FSM_LCD_maestra is
        port(
        Clk: in std_logic;
        Rst: in std_logic;
        Inicia_lcd: in std_logic;
        fin_i: in std_logic;
        Inicia_i: out std_logic;
        Inicia_e: out std_logic;
        sel_lcd_mux: out std_logic
        );
end FSM_LCD_maestra;
architecture FSM_LCD_maestra of FSM_LCD_maestra is
type estados is (s0,s1,s2);
signal Ep,Es: estados;
begin
        process(Ep,Inicia_lcd,fin_i)
        begin
                case Ep is
                        when s0 => Inicia_i<='0';Inicia_e<='0';sel_lcd_mux<='0';
                        if Inicia_lcd='0' then
                                Es<=s0;
                        else
                                Es<=s1;
                        end if;
                        when s1 => Inicia_i<='1';Inicia_e<='0';sel_lcd_mux<='0';
                        if fin_i='0' then
                                Es<=s1;
                        else
                                Es<=s2;
                        end if;
                        when s2 => Inicia_i<='0';Inicia_e<='1';sel_lcd_mux<='1';
                        Es<=s2;
                end case;
        end process;
        process(Clk,Rst)
        begin
```

```
                if Rst='1' then
                        Ep<=s0;
                elsif Clk'event and Clk='1' then
                        Ep<=Es;
                end if;
        end process;
end FSM_LCD_maestra;
```

A.3.6 Listado bloque "sel_cnt".

```
library   ieee;
use ieee.std_logic_1164.all;
use IEEE.std_logic_arith.all;
use IEEE.std_logic_unsigned.all;
entity sel_cnt is
        port(
        Clk:    in std_logic;
        Rst:    in std_logic;
        Fin_e:  in std_logic;
        selector: out std_logic_vector(6 downto 0)
        );
end sel_cnt;
architecture sel_cnt of sel_cnt is
signal Cp,Cs: std_logic_vector(6 downto 0);
begin
        process(Fin_e,Cp)
                begin
                if Fin_e='0' then
                        Cs<=Cp;
                else
                        Cs<=Cp+1;
                        if Cp="1001111" then
                          Cs<="0000000";
                   else
                          Cs<=Cp+1;
                end if;
                end if;
                selector<=Cp;

        end process;
process(Clk,Rst)
begin
        if Rst='1' then
                Cp<=(others=>'0');
        elsif Clk' event and Clk='1' then
                Cp<=Cs;
        end if;
end process;

end sel_cnt;
```

A.3.7 Listado bloque "mux_caracter".

```
library ieee;
use ieee.std_logic_1164.all;
entity mux_caracter is
        port(
        selector: in  std_logic_vector(6 downto 0);
        BCD_ref: in std_logic_vector(11 downto 0);
        BCD_med: in std_logic_vector(11 downto 0);
        caracter:  out        std_logic_vector(7 downto 0)
        );
end mux_caracter;
architecture mux_caracter of mux_caracter is
signal Rm,e,f,r,n,c,i,a,Tm,m,p,t,u: std_logic_vector(7 downto 0);
signal space,dos_puntos: std_logic_vector(7 downto 0);
signal uni_ref,dec_ref,cen_ref: std_logic_vector(7 downto 0);
```

```
signal uni_med,dec_med,cen_med: std_logic_vector(7 downto 0);
begin
        Rm<="01010010";e<="01100101";f<="01100110";r<="01110010";n<="01101110";c<="01100011";i<="0
1101001";a<="01100001";
        Tm<="01010100";m<="01101101";p<="01110000";t<="01110100";u<="01110101";dos_puntos<="00111
010";
        space<="00100000";
        uni_ref(7 downto 4)<="0011"; uni_ref(3 downto 0)<=BCD_ref(3 downto 0);
        dec_ref(7 downto 4)<="0011"; dec_ref(3 downto 0)<=BCD_ref(7 downto 4);
        cen_ref(7 downto 4)<="0011"; cen_ref(3 downto 0)<=BCD_ref(11 downto 8);

        uni_med(7 downto 4)<="0011"; uni_med(3 downto 0)<=BCD_med(3 downto 0);
        dec_med(7 downto 4)<="0011"; dec_med(3 downto 0)<=BCD_med(7 downto 4);
        cen_med(7 downto 4)<="0011"; cen_med(3 downto 0)<=BCD_med(11 downto 8);
        process(selector,uni_ref,dec_ref,cen_ref,uni_med,dec_med,cen_med)
        begin
                case selector is
                  when "0000000" => caracter<=Rm;   --00      DISPLAY REAL Linea 1
                  when "0000001" => caracter<=e;      --01
                  when "0000010" => caracter<=f;        --02
                  when "0000011" => caracter<=e;        --03
                  when "0000100" => caracter<=r;        --04
                  when "0000101" => caracter<=e;        --05
                  when "0000110" => caracter<=n;        --06
                  when "0000111" => caracter<=c;        --07
                  when "0001000" => caracter<=i;      --08
                  when "0001001" => caracter<=a;        --09
                  when "0001010" => caracter<=dos_puntos;   --0A
                  when "0001011" => caracter<=space;            --0B
                  when "0001100" => caracter<=cen_ref;            --0C
                  when "0001101" => caracter<=dec_ref;          --0D
                  when "0001110" => caracter<=uni_ref;          --0E
                  when "0001111" => caracter<="11011111";   --0F
                  when "0010000" => caracter<=space;--10 ---------- DISPLAY VIRTUAL Linea1
                  when "0010001" => caracter<=space;--11
                  when "0010010" => caracter<=space;            --12
                  when "0010011" => caracter<=space;--13
                  when "0010100" => caracter<=space;--14
                  when "0010101" => caracter<=space;--15
                  when "0010110" => caracter<=space;--16
                  when "0010111" => caracter<=space;  --17
                  when "0011000" => caracter<=space;--18
                  when "0011001" => caracter<=space;--19
                  when "0011010" => caracter<=space;  --1A
                  when "0011011" => caracter<=space;  --1B
                  when "0011100" => caracter<=space;  --1C
                  when "0011101" => caracter<=space;  --1D
                  when "0011110" => caracter<=space;  --1E
                  when "0011111" => caracter<=space;  --1F
                  when "0100000" => caracter<=space;--20
                  when "0100001" => caracter<=space;--21
                  when "0100010" => caracter<=space;--22
                  when "0100011" => caracter<=space;--23
                  when "0100100" => caracter<=space;--24
                when "0100101" => caracter<=space;      --25
                  when "0100110" => caracter<=space;--26
                  when "0100111" => caracter<=space;--27
                  when "0101000" => caracter<=Tm;     --40    --DISPLAY REAL Linea 2
                  when "0101001" => caracter<=e;      --41
                  when "0101010" => caracter<=m;      --42
                  when "0101011" => caracter<=p;        --43
                  when "0101100" => caracter<=e;        --44
                  when "0101101" => caracter<=r;      --45
                  when "0101110" => caracter<=a;      --46
                  when "0101111" => caracter<=t;      --47
                  when "0110000" => caracter<=u;      --48
```

```
                when "0110001" => caracter<=r;          --49
                when "0110010" => caracter<=a;          --4A
                when "0110011" => caracter<=dos_puntos;--4B
                when "0110100" => caracter<=cen_med;            --4C
                when "0110101" => caracter<=dec_med;            --4D
                when "0110110" => caracter<=uni_med;          --4F
                when "0110111" => caracter<="11011111";    --50
                when "0111000" => caracter<=space;--51     --Display virtual linea 2
                when "0111001" => caracter<=space;  --52
                when "0111010" => caracter<=space;--53
                when "0111011" => caracter<=space;--54
                when "0111100" => caracter<=space;--55
                when "0111101" => caracter<=space;--56
                when "0111110" => caracter<=space;--57
                when "0111111" => caracter<=space;  --58
                when "1000000" => caracter<=space;  --59
                when "1000001" => caracter<=space;  --5A
                when "1000010" => caracter<=space;  --5B
                when "1000011" => caracter<=space;  --5C
                when "1000100" => caracter<=space;  --5D
                when "1000101" => caracter<=space;  --5E
                when "1000110" => caracter<=space;  --5F
                when "1000111" => caracter<=space;  --60
                when "1001000" => caracter<=space;  --61
                when "1001001" => caracter<=space;  --62
                when "1001010" => caracter<=space;  --63
                when "1001011" => caracter<=space;  --64
                when "1001100" => caracter<=space;  --65
                when "1001101" => caracter<=space;  --66
                when "1001110" => caracter<=space;  --67
                when "1001111" => caracter<=space;  --68
                when others => caracter<=space;
        end case;
end process;

end mux_caracter;
```

A.3.8 Listado bloque "LCD_mux".

```
library ieee;
use ieee.std_logic_1164.all;
entity LCD_mux is
        port(
        sel_mux_lcd: in std_logic;
        RS_i,RW_i,E_i: in std_logic;
        Data_i: in std_logic_vector(7 downto 0);
        RS_e,RW_e,E_e: in std_logic;
        Data_e: in std_logic_vector(7 downto 0);
        RS,RW,E: out std_logic;
        Data_LCD: out std_logic_vector(7 downto 0)
        );
end LCD_mux;
architecture LCD_mux of LCD_mux is
begin
 process(sel_mux_lcd,RS_i,RW_i,E_i,Data_i,RS_e,RW_e,E_e,Data_e)
 begin
          if sel_mux_lcd='0' then
                RS<=RS_i;RW<=RW_i;E<=E_i;Data_lcd<=Data_i;
          else
                RS<=RS_e;RW<=RW_e;E<=E_e;Data_lcd<=Data_e;
          end if;
 end process;
end LCD_mux;
```

A.3.9 Listado bloque "t_8Hz".

```
library ieee;
```

```
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity t_8Hz is
        port(
        Clk: in std_logic;
        Rst: in std_logic;
        T_out: out std_logic
        );
end t_8Hz;
architecture t_8Hz of t_8Hz is
signal Cp,Cs: std_logic_vector(22 downto 0);
begin
        process(Cp)
        begin
                Cs<=Cp+1;
                T_out<='0';
                if Cp="10111110101111000001111" then
                        Cs<=(others=>'0');
                        T_out<='1';
                end if;
        end process;
process(Clk,Rst)
begin
        if Rst='1' then
                Cp<=(others=>'0');
        elsif Clk' event and Clk='1' then
                Cp<=Cs;
        end if;
end process;
end t_8Hz;
```

A.3.10 Listado bloque "AND_gate".

```
library ieee;
use ieee.std_logic_1164.all;
entity AND_gate is
        port(
        A: in std_logic;
        B: in std_logic;
        C: out std_logic
        );
end AND_gate;
architecture AND_gate of AND_gate is
begin
        C<=A and B;
end AND_gate;
```

A.3.11 Listado bloque "cont_10bits".

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity cont_10bits is
        port(
        Clk: in std_logic;
        Rst: in std_logic;
        EN_UP: in std_logic;
        EN_DN: in std_logic;
        Cuenta: out std_logic_vector(9 downto 0)
        );
end cont_10bits;
architecture cont_10bits of cont_10bits is
signal Cp,Cs: std_logic_vector(9 downto 0);
begin
  process(Cp,EN_UP,EN_DN)
```

```
 begin
        if (EN_UP='0' and EN_DN='0') then
                Cs<=Cp;
        elsif (EN_UP='0' and EN_DN='1') then
                if Cp="00000000" then
                        Cs<="1111100111";
                else
                        Cs<=Cp-1;
                end if;
        elsif (EN_UP='1' and EN_DN='0') then
                if Cp="1111100111" then
                   Cs<=(others=>'0');
                else
                        Cs<=Cp+1;
                end if;
        else
                Cs<=Cp;
        end if;
        Cuenta<=Cp;
 end process;
 process(Clk,Rst)
 begin
        if Rst='1' then
                Cp<=(others=>'0');
        elsif Clk'event and Clk='1' then
                Cp<=Cs;
        end if;
 end process;
end cont_10bits;
```

A.3.12 Listado bloque "factor_ref".

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity factor_ref is
        port(
        ref_bin: in std_logic_vector(9 downto 0);
        ref_factor: out std_logic_vector(35 downto 0)     --25.11
        );
end factor_ref;
architecture factor_ref of factor_ref is
signal A: std_logic_vector(20 downto 0);  --10.11
signal B: std_logic_vector(20 downto 0);  --10.11
signal C: std_logic_vector(41 downto 0);  --20.22
begin
        A(20 downto 11)<=ref_bin;     A(10 downto 0) <=(others=>'0');
        B(20 downto 15)<=(others=>'0');B(14 downto 11)<="1000";     B(10 downto 0)<="00110001001";
        process(A,B)
        variable R: unsigned (41 downto 0);
        begin
                R:= unsigned(A)*unsigned(B);
                C<= std_logic_vector(R);
        end process;
        ref_factor(35 downto 31)<=(others=>'0');  ref_factor(30 downto 11)<=C(41 downto 22);          --25
        ref_factor(10 downto  0)<=C(21 downto 11);  --.11
end factor_ref;
```

A.3.13 Listado bloque "reg_pp_36".

```
library ieee;
use ieee.std_logic_1164.all;
entity reg_pp_36 is
        Generic(n:integer:=36);
        port(
        Clk: in std_logic;
```

```
        Rst: in std_logic;
        D_in: in std_logic_vector(n-1 downto 0);
        L_pp: in std_logic;
        Data: out std_logic_vector(n-1 downto 0)
        );
end reg_pp_36;
architecture reg_pp_36 of reg_pp_36 is
signal pD,sD: std_logic_vector(n-1 downto 0);
begin
        process(D_in,L_pp,pD)
        begin
                if L_pp='0' then
                        sD<=pD;
                else
                        sD<=D_in;
                end if;
                Data<=pD;
        end process;
        process(Clk,Rst)
        begin
                if Rst='1' then
                        pD<=(others=>'0');
                elsif Clk'event and Clk='1' then
                        pD<=sD;
                end if;
        end process;
end reg_pp_36;
```

A.4 Listados de modulo "pwm".

A.4.1 Integración de bloques de módulo "pwm".

```
library ieee;
use ieee.std_logic_1164.all;
entity PWM is
        port(
        Clk: in std_logic;
        Rst: in std_logic;
        Inicio_PWM: in std_logic;
        D:  in std_logic_vector(13 downto 0);
        PWM_s: out std_logic
        );
end PWM;
architecture PWM of PWM is
signal t_a_1ms_aux,t_r_1ms_aux,t_a_100ns_aux,t_r_100ns_aux,L_pp_aux: std_logic;
signal OPC_aux: std_logic_vector(1 downto 0);
signal cnt_aux: std_logic_vector(13 downto 0);
signal D_aux: std_logic_vector(13 downto 0);
component t_1ms is
        port(
        Clk: in std_logic;
        Rst: in std_logic;
        t_a: in std_logic;
        t_r: out std_logic
        );
end component;
component t_100ns
        port(
        Clk: in std_logic;
        Rst: in std_logic;
        t_a: in std_logic;
        t_r: out std_logic
        );
end component;
component FSM_PWM is
        port(
```

```
        Clk:   in std_logic;
        Rst:   in std_logic;
        Inicio_PWM: in std_logic;
        t_r_100ns: in  std_logic;
        t_r_1ms: in  std_logic;
        L_pp: out std_logic;
        t_a100_ns: out std_logic;
        t_a_1ms: out std_logic;
        OPC: out std_logic_vector(1 downto 0)
        );
end component;
component cnt_PWM
        port(
        Clk: in std_logic;
        Rst: in std_logic;
        OPC: in std_logic_vector(1 downto 0);
        cnt: out std_logic_vector(13 downto 0)
        );
end component;
component reg_pp_14
        Generic(n:integer:=14);
        port(
        Clk: in std_logic;
        Rst: in std_logic;
        D_in: in std_logic_vector(n-1 downto 0);
        L_pp: in std_logic;
        Data: out std_logic_vector(n-1 downto 0)
        );
end component;
component comp
        port(
        D: in std_logic_vector(13 downto 0);
        cnt: in std_logic_vector(13 downto 0);
        PWM_S: out std_logic
        );
end component;
begin
        PWM0: t_1ms port map(Clk,Rst,t_a_1ms_aux,t_r_1ms_aux); --puede ser externo
        PWM1: t_100ns port map(Clk,Rst,t_a_100ns_aux,t_r_100ns_aux);
        PWM2:                                   FSM_PWM                                 port
map(Clk,Rst,Inicio_PWM,t_r_100ns_aux,t_r_1ms_aux,L_pp_aux,t_a_100ns_aux,t_a_1ms_aux,OPC_aux);
        PWM3: cnt_PWM port map(Clk,Rst,OPC_aux,cnt_aux);
        PWM4: reg_pp_14 port map(Clk,Rst,D,L_pp_aux,D_aux);
        PWM5: comp port map(D_aux,cnt_aux,PWM_s);
end PWM;
```

A.4.2 Listado bloque "t_1ms".

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity t_1ms is
        port(
        Clk: in std_logic;
        Rst: in std_logic;
        t_a: in std_logic;
        t_r: out std_logic
        );
end t_1ms;
architecture t_1ms of t_1ms is
signal Cp,Cs: std_logic_vector(15 downto 0);
begin
        process(Cp,t_a)
        begin
                case t_a is
                        when '0' =>
```

```
                    Cs<=(others=>'0');
                    when others =>
                     if(Cs="1100001101001111") then
                            Cs<="1100001101001111";
                     else
                            Cs<=Cs+1;
                     end if;
            end case;
            if Cs="1100001101001111" then
                    t_r<='1';
            else
                    t_r<='0';
            end if;
    end process;
    process(Clk,Rst)
  begin
    if Rst='1' then
            Cp<=(others=>'0');
    elsif Clk' event and Clk='1' then
            Cp<=Cs;
    end if;
end process;
end t_1ms;
```

A.4.3 Listado bloque "t_100ns".

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity t_100ns is
    port(
    Clk: in std_logic;
    Rst: in std_logic;
    t_a: in std_logic;
    t_r: out std_logic
    );
end t_100ns;
architecture t_100ns of t_100ns is
signal Cp,Cs: std_logic_vector(2 downto 0);
begin
    process(Cp,t_a)
    begin
            case t_a is
                    when '0' =>
                    Cs<=(others=>'0');
                    when others =>
                     if(Cs="100") then
                            Cs<=(others=>'0');
                     else
                            Cs<=Cs+1;
                     end if;
            end case;
            if Cp="100" then
                    t_r<='1';
            else
                    t_r<='0';
            end if;
    end process;
process(Clk,Rst)
begin
    if Rst='1' then
            Cp<=(others=>'0');
    elsif Clk' event and Clk='1' then
            Cp<=Cs;
    end if;
end process;
```

```
end t_100ns;
```

A.4.4 Listado bloque "FSM_PWM".

```
library ieee;
use ieee.std_logic_1164.all;
entity FSM_PWM is
        port(
        Clk:   in  std_logic;
        Rst:   in  std_logic;
        Inicio_PWM: in std_logic;
        t_r_100ns: in  std_logic;
        t_r_1ms: in  std_logic;
        L_pp: out std_logic;
        t_a100_ns: out std_logic;
        t_a_1ms: out std_logic;
        OPC: out std_logic_vector(1 downto 0)
        );
end FSM_PWM;
architecture FSM_PWM of FSM_PWM is
type Estados is (s0,s1,s2,s3,s4);
signal Ep,Es: Estados;
begin
        process(Ep,t_r_100ns,t_r_1ms,Inicio_PWM)
        begin
                case Ep is
                        when s0 => L_pp<='0';t_a100_ns<='0';t_a_1ms<='0';OPC<="10";
                        if Inicio_PWM='0' then
                                Es<=s0;
                        else
                                Es<=s1;

                        end if;
                        when s1 => L_pp<='1';t_a100_ns<='0';t_a_1ms<='0';OPC<="10";
                        Es<=s2;
                        when s2 => L_pp<='0';t_a100_ns<='1';t_a_1ms<='1';OPC<="00";
                        if t_r_100ns='0' then
                                Es<=s2;
                        else
                                Es<=s3;

                        end if;
                        when s3 => L_pp<='0';t_a100_ns<='1';t_a_1ms<='1';OPC<="01";
                        Es<=s4;
                        when others => L_pp<='0';t_a100_ns<='1';t_a_1ms<='1';OPC<="00";
                        if t_r_1ms='0' then
                                Es<=s2;
                        else
                                Es<=s1;

                        end if;
                end case;
  end process;
        process(Clk,Rst)
        begin
                if Rst='1' then
                        Ep<=s0;
                elsif Clk' event and Clk='1' then
                        Ep<=Es;
                end if;
        end process;

end FSM_PWM;
```

A.4.5 Listado bloque "cnt_PWM".

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;
entity cnt_PWM is
        port(
        Clk: in std_logic;
        Rst: in std_logic;
        OPC: in std_logic_vector(1 downto 0); -- 00 Hold; 01 Inc; 1x Clear
        cnt: out std_logic_vector(13 downto 0)
        );
end cnt_PWM;
architecture cnt_PWM of cnt_PWM is
signal Cp,Cs: std_logic_vector(13 downto 0);
begin
        process(OPC,Cp)
        begin
                case OPC is
                        when "00" =>
                        Cs<=Cp;
                        when "01" =>
                        --if t='1' then
                                --if Cp="10011100010000" then
                                --      Cs<=(others=>'0');
                          --else
                            Cs<=Cs+1;
                          --end if;
                        --else
                        --      Cs<=cp;
                        --end if;
                        when others =>
                        Cs<=(others=>'0');
                end case;
        cnt<=Cp;
        end process;

        process(Clk,Rst)
        begin
                if Rst='1' then
                        Cp<=(others=>'0');
                elsif Clk' event and Clk='1' then
                        Cp<=Cs;
                end if;
        end process;
end cnt_PWM;
```

A.4.6 Listado bloque "reg_pp_14".

```
library ieee;
use ieee.std_logic_1164.all;
entity reg_pp_14 is
        Generic(n:integer:=14);
        port(
        Clk: in std_logic;
        Rst: in std_logic;
        D_in: in std_logic_vector(n-1 downto 0);
        L_pp: in std_logic;
        Data: out std_logic_vector(n-1 downto 0)
        );
end reg_pp_14;
architecture reg_pp_14 of reg_pp_14 is
signal pD,sD: std_logic_vector(n-1 downto 0);
begin
        process(D_in,L_pp,pD)
        begin
```

```
                if L_pp='0' then
                        sD<=pD;
                else
                        sD<=D_in;
                end if;
                Data<=pD;
        end process;
        process(Clk,Rst)
        begin
                if Rst='1' then
                        pD<=(others=>'0');
                elsif Clk'event and Clk='1' then
                        pD<=sD;
                end if;
        end process;
end reg_pp_14;
```

A.4.7 Listado bloque "comp".

```
library ieee;
use ieee.std_logic_1164.all;
entity comp is
        port(
        D: in std_logic_vector(13 downto 0);
        cnt: in std_logic_vector(13 downto 0);
        PWM_S: out std_logic
        );
end comp;
architecture comp of comp is
begin
        process(D,cnt)
        begin
                if D>cnt then
                        PWM_s<='1';
                else
                        PWM_s<='0';
                end if;
        end process;
end comp;
```

A.5 Hoja de datos para el ADC ADS7816.

BURR-BROWN®

ADS7816

12-Bit High Speed Micro Power Sampling ANALOG-TO-DIGITAL CONVERTER

FEATURES

- 200kHz SAMPLING RATE
- MICRO POWER:
  1.9mW at 200kHz
  150µW at 12.5kHz
- POWER DOWN: 3µA Max
- 8-PIN MINI-DIP, SOIC, AND MSOP
- DIFFERENTIAL INPUT
- SERIAL INTERFACE

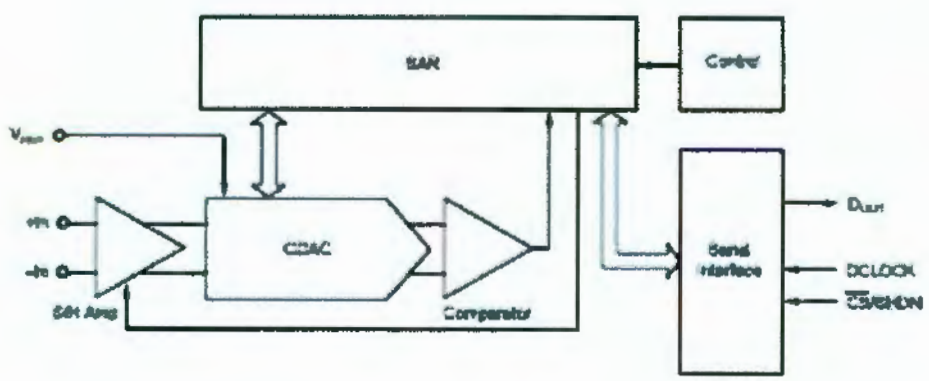
APPLICATIONS

- BATTERY OPERATED SYSTEMS
- REMOTE DATA ACQUISITION
- ISOLATED DATA ACQUISITION

DESCRIPTION

The ADS7816 is a 12-bit, 200kHz sampling analog-to-digital converter. It features low power operation with automatic power down, a synchronous serial interface, and a differential input. The reference voltage can be varied from 100mV to 5V, with a corresponding resolution from 24µV to 1.22mV.

Low power, automatic power down, and small size make the ADS7816 ideal for battery operated systems or for systems where a large number of signals must be acquired simultaneously. It is also ideal for remote and/or isolated data acquisition. The ADS7816 is available in an 8-pin plastic mini-DIP, an 8-lead SOIC, or an 8-lead MSOP package.



PDS-1195B

Printed in U.S.A., March, 1997

SPECIFICATIONS

At –40°C to +85°C, $+V_{CC}$ = +5V, V_{REF} = +5V, f_{SAMPLE} = 200kHz, f_{CLK} = 16 × f_{SAMPLE}, unless otherwise specified.

PARAMETER	CONDITIONS	ADS7816			ADS7816B			ADS7816C			UNITS
		MIN	TYP	MAX	MIN	TYP	MAX	MIN	TYP	MAX	
ANALOG INPUT											
Full-Scale Input Span	+in – (–in)	0		V_{REF}	•		•	•		•	V
Absolute Input Voltage	+in	–0.2		V_{CC} +0.2	•		•	•		•	V
	–in	–0.2		+0.2	•		•	•		•	V
Capacitance			25			•			•		pF
Leakage Current			±1			•			•		µA
SYSTEM PERFORMANCE											
Resolution			12			•			•		Bits
No Missing Codes		11			12			•			Bits
Integral Linearity Error			±0.5	±2		±0.5	±2		±0.5	±1	LSB[1]
Differential Linearity Error			±0.5	±2		±0.5	±1		±0.25	±0.75	LSB
Offset Error				±4			•			•	LSB
Gain Error				±4			•			•	LSB
Noise			33			•			•		µVrms
Power Supply Rejection			82			•			•		dB
SAMPLING DYNAMICS											
Conversion Time				12			•			•	Clk Cycles
Acquisition Time		1.5			•			•			Clk Cycles
Throughput Rate				200			•			•	kHz
DYNAMIC CHARACTERISTICS											
Total Harmonic Distortion	V_{IN} = 5.0Vp-p at 1kHz		–84			•			•		dB
	V_{IN} = 5.0Vp-p at 5kHz		–82			•			•		dB
SINAD	V_{IN} = 5.0Vp-p at 1kHz		72			•			•		dB
Spurious Free Dynamic Range	V_{IN} = 5.0Vp-p at 1kHz		85			•			•		dB
REFERENCE INPUT											
Voltage Range		0.1		V_{CC}	•		•	•		•	V
Resistance	$\overline{CS}$ = GND, f_{SAMPLE} = 0Hz		5			•			•		kΩ
	$\overline{CS}$ = V_{CC}		5			•			•		kΩ
Current Drain	At Code 710h		35	100		•	•		•	•	µA
	f_{SAMPLE} = 12.5kHz		2.4	20		•	•		•	•	µA
	$\overline{CS}$ = V_{CC}		0.001	3		•	•		•	•	µA
DIGITAL INPUT/OUTPUT											
Logic Family			CMOS			•			•		
Logic Levels											
V_{IH}	I_{IH} = +5µA	3		V_{CC} +0.3	•		•	•		•	V
V_{IL}	I_{IL} = +5µA	–0.3		0.8	•		•	•		•	V
V_{OH}	I_{OH} = –250µA	3.6			•			•			V
V_{OL}	I_{OL} = 250µA			0.4			•			•	V
Data Format			Straight Binary				•			•	
POWER SUPPLY REQUIREMENTS											
V_{CC}	Specified Performance	4.50		5.25	•		•	•		•	V
Quiescent Current			360	700		•	•		•	•	µA
	f_{SAMPLE} = 12.5kHz[2]		30			•			•		µA
	f_{SAMPLE} = 12.5kHz[3]		280	400		•			•		µA
Power Down	$\overline{CS}$ = V_{CC}, f_{SAMPLE} = 0Hz			3			•			•	µA
TEMPERATURE RANGE											
Specified Performance		–40		+85	•		•	•		•	°C

• Specifications same as grade to the left.

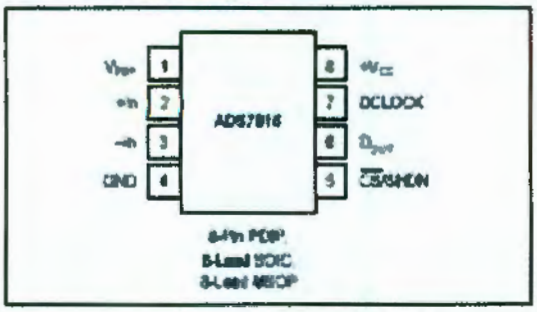
NOTE: (1) LSB means Least Significant Bit, with V_{REF} equal to +5V, one LSB is 1.22mV. (2) f_{CLK} = 3.2MHz, $\overline{CS}$ = V_{CC} for 201 clock cycles out of every 256. (3) See the Power Dissipation section for more information regarding lower sample rates.

ABSOLUTE MAXIMUM RATINGS[1]

$+V_{CC}$	+6V
Analog Input	−0.3V to ($+V_{CC}$ + 0.3V)
Logic Input	−0.3V to ($+V_{CC}$ + 0.3V)
Case Temperature	+100°C
Junction Temperature	+150°C
Storage Temperature	+125°C
External Reference Voltage	+5.5V

NOTE: (1) Stresses above these ratings may permanently damage the device.

PIN CONFIGURATION



ELECTROSTATIC DISCHARGE SENSITIVITY

Electrostatic discharge can cause damage ranging from performance degradation to complete device failure. Burr-Brown Corporation recommends that all integrated circuits be handled and stored using appropriate ESD protection methods.

ESD damage can range from subtle performance degradation to complete device failure. Precision integrated circuits may be more susceptible to damage because very small parametric changes could cause the device not to meet published specifications.

PIN ASSIGNMENTS

PIN	NAME	DESCRIPTION
1	V_{REF}	Reference Input
2	+In	Non Inverting Input
3	−In	Inverting Input. Connect to ground or to remote ground sense point
4	GND	Ground
5	$\overline{CS}$/SHDN	Chip Select when LOW. Shutdown Mode when HIGH
6	D_{OUT}	The serial output data word is comprised of 12 bits of data. In operation the data is valid on the falling edge of DCLOCK. The second clock pulse after the falling edge of $\overline{CS}$ enables the serial output. After one null bit the data is valid for the next 12 edges.
7	DCLOCK	Data Clock synchronizes the serial data transfer and determines conversion speed.
8	$+V_{CC}$	Power Supply

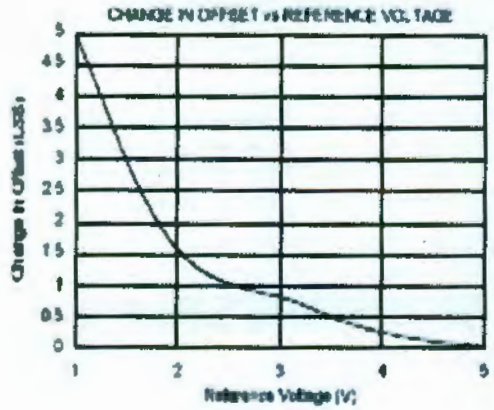
PACKAGE/ORDERING INFORMATION

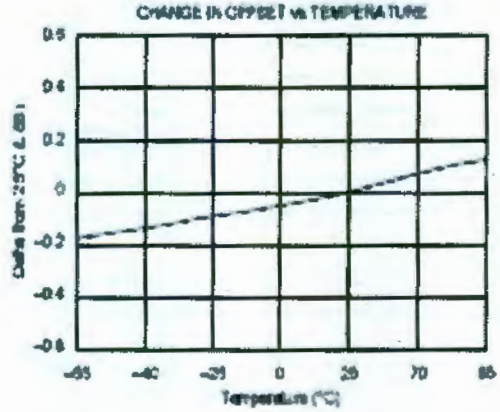
PRODUCT	MAXIMUM INTEGRAL LINEARITY ERROR (LSB)	MAXIMUM DIFFERENTIAL LINEARITY ERROR (LSB)	TEMPERATURE RANGE	PACKAGE	PACKAGE DRAWING NUMBER[1]
ADS7816P	±2	±2	−40°C to +85°C	Plastic DIP	006
ADS7816U	±2	±2	−40°C to +85°C	SOIC	182
ADS7816E	±2	±2	−40°C to +85°C	MSOP	337
ADS7816PB	±2	±1	−40°C to +85°C	Plastic DIP	006
ADS7816UB	±2	±1	−40°C to +85°C	SOIC	182
ADS7816EB	±2	±1	−40°C to +85°C	MSOP	337
ADS7816PC	±1	±0.75	−40°C to +85°C	Plastic DIP	006
ADS7816UC	±1	±0.75	−40°C to +85°C	SOIC	182
ADS7816EC	±1	±0.75	−40°C to +85°C	MSOP	337

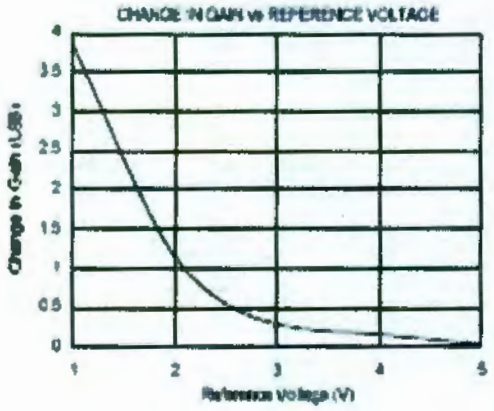
NOTE: (1) For detailed drawing and dimension table, please see end of data sheet, or Appendix C of Burr-Brown IC Data Book.

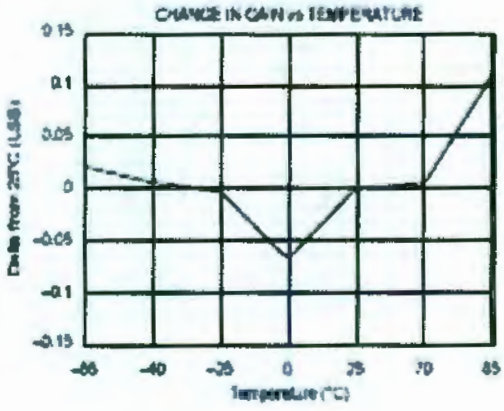
TYPICAL PERFORMANCE CURVES

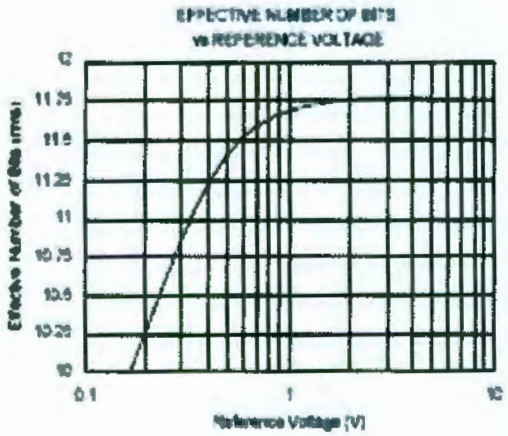
At T_A = +25°C, V_{CC} = +5V, V_{REF} = +5V, f_{SAMPLE} = 200kHz and f_{SCLK} = 18 × f_{SAMPLE}, unless otherwise specified.

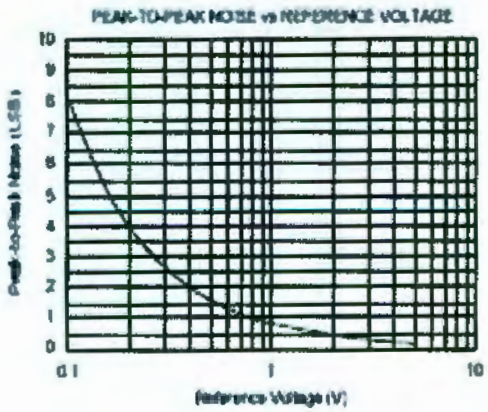






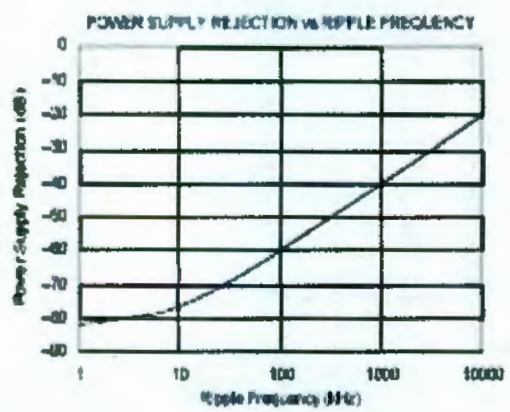


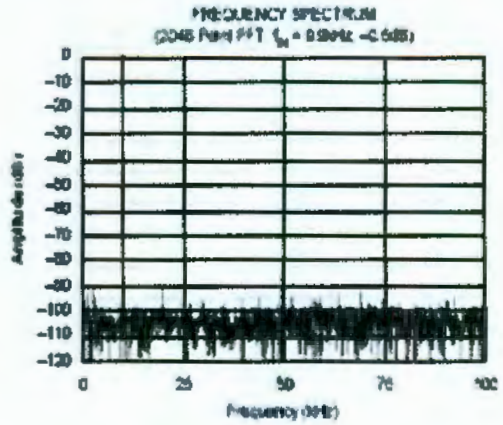


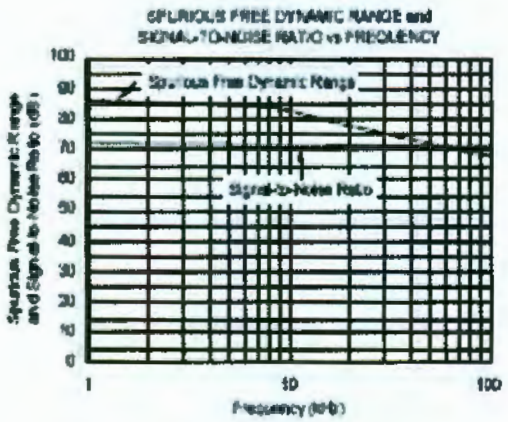


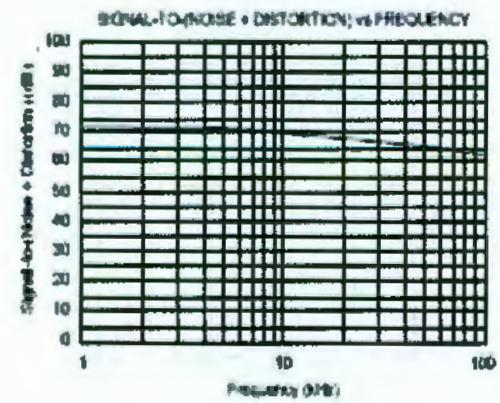
TYPICAL PERFORMANCE CURVES (CONT)

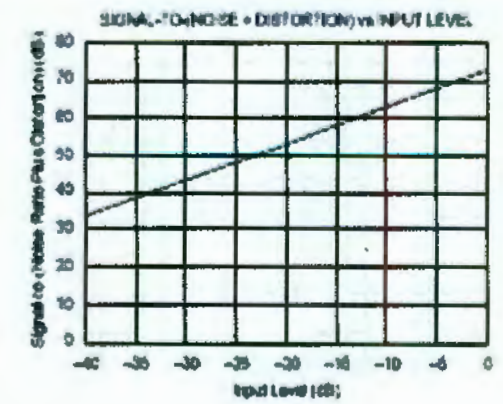
At T_A = +25°C, V_{CC} = +5V, V_{REF} = +5V, f_{SAMPLE} = 200kHz, and t_{ACQ} = 16 × t_{SAMPLE}, unless otherwise specified.





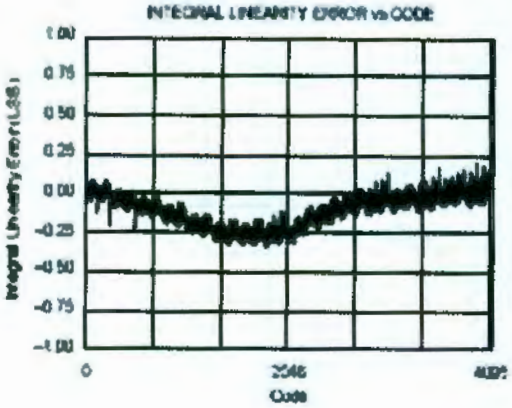


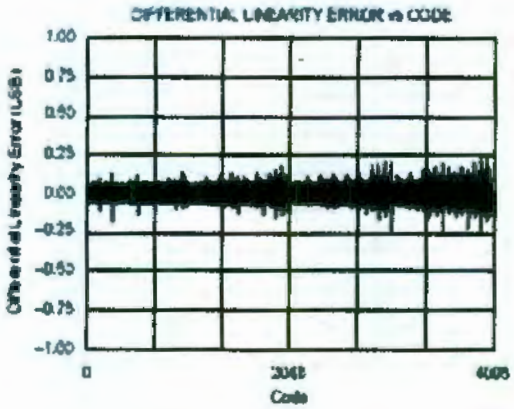


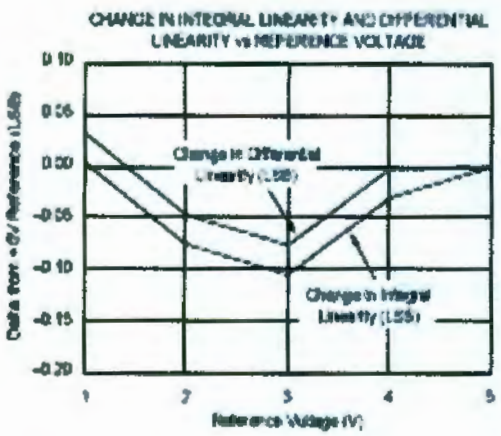


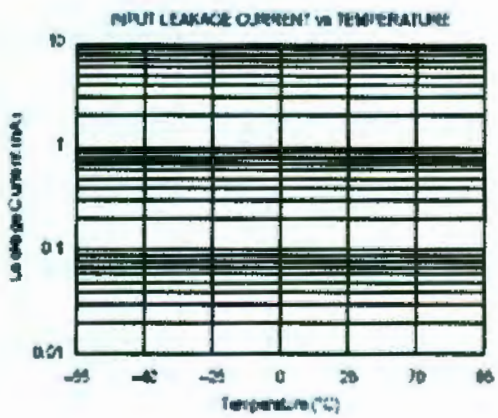
TYPICAL PERFORMANCE CURVES (CONT)

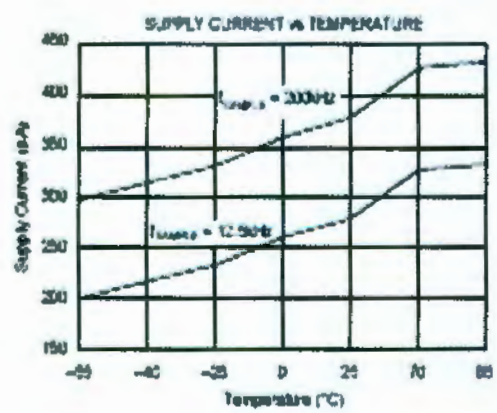
At T_A = +25°C, V_{CC} = +5V, V_{REF} = +5V, f_{SAMPLE} = 200kHz, and f_{CLK} = 16 × f_{SAMPLE}, unless otherwise specified.

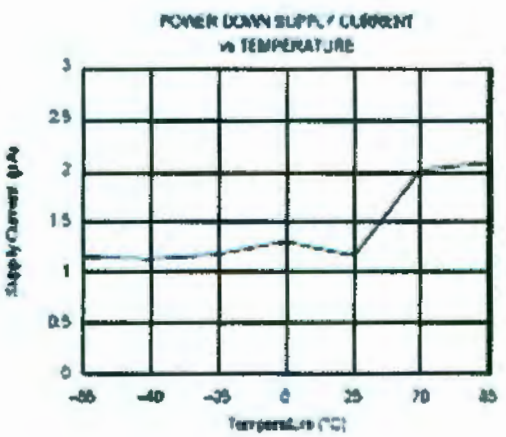






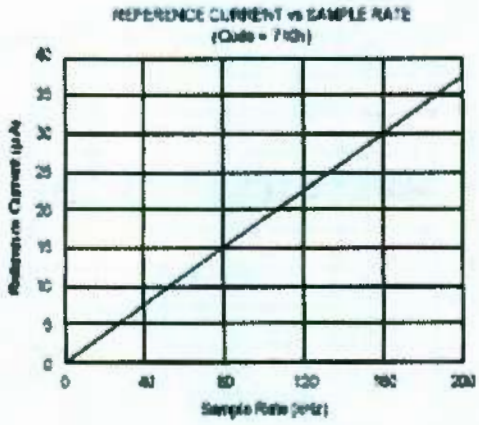


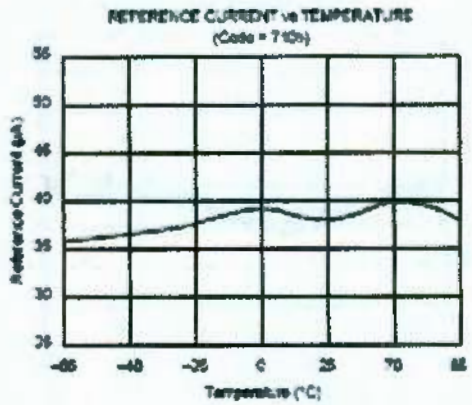


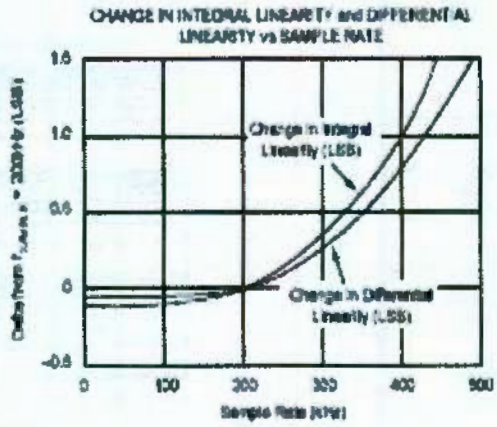


TYPICAL PERFORMANCE CURVES (CONT)

At T_A = +25°C, V_{CC} = +5V, V_{REF} = +5V, f_{SAMPLE} = 200kHz, and f_{CLK} = 16 × f_{SAMPLE}, unless otherwise specified.







THEORY OF OPERATION

The ADS7816 is a classic successive approximation register (SAR) analog-to-digital (A/D) converter. The architecture is based on capacitive redistribution which inherently includes a sample hold function. The converter is fabricated on a 0.6µ CMOS process. The architecture and process allow the ADS7816 to acquire and convert an analog signal at up to 200,000 conversions per second while consuming very little power.

The ADS7816 requires an external reference, an external clock, and a single +5V power source. The external reference can be any voltage between 100mV and V_{CC}. The value of the reference voltage directly sets the range of the analog input. The reference input current depends on the conversion rate of the ADS7816.

The external clock can vary between 10kHz (625Hz throughput) and 3.2MHz (200kHz throughput). The duty cycle of the clock is essentially unimportant as long as the minimum high and low times are at least 150ns. The minimum clock frequency is set by the leakage on the capacitors internal to the ADS7816.

The analog input is provided to two input pins: +In and –In. When a conversion is initiated, the differential input on these pins is sampled on the internal capacitor array. While a conversion is in progress, both inputs are disconnected from any internal function.

The digital result of the conversion is clocked out by the DCLOCK input and is provided serially, most significant bit first, on the D_{OUT} pin. The digital data that is provided on the D_{OUT} pin is for the conversion currently in progress—there is no pipeline delay. It is possible to continue to clock the ADS7816 after the conversion is complete and to obtain the serial data least significant bit first. See the Digital Interface section for more information.

ANALOG INPUT

The +In and –In input pins allow for a differential input signal. Unlike some converters of this type, the –In input is not re-sampled later in the conversion cycle. When the converter goes into the hold mode, the voltage difference between +In and –In is captured on the internal capacitor array.

The range of the –In input is limited to ±200mV. Because of this, the differential input can be used to reject only small signals that are common to both inputs. Thus, the –In input is best used to sense a remote signal ground that may move slightly with respect to the local ground potential.

The input current on the analog inputs depends on a number of factors: sample rate, input voltage, source impedance, and power down mode. Essentially, the current into the ADS7816 charges the internal capacitor array during the sample period. After this capacitance has been fully charged, there is no further input current. The source of the analog input voltage must be able to charge the input capacitance (25pF) to a 12-bit settling level within 1.5 clock cycles. When the converter goes into the hold mode or while it is in the power down mode, the input impedance is greater than 1GΩ.

Care must be taken regarding the absolute analog input voltage. To maintain the linearity of the converter, the –In input should not exceed GND ±200mV. The +In input should always remain within the range of GND –200mV to V_{CC} +200mV. Outside of these ranges, the converter's linearity may not meet specifications.

REFERENCE INPUT

The external reference sets the analog input range. The ADS7816 will operate with a reference in the range of 100mV to V_{CC}. There are several important implications of this.

As the reference voltage is reduced, the analog voltage weight of each digital output code is reduced. This is often referred to as the LSB (least significant bit) size and is equal to the reference voltage divided by 4096. This means that any offset or gain error inherent in the A/D converter will appear to increase, in terms of LSB size, as the reference voltage is reduced. The typical performance curves of "Change in Offset vs Reference Voltage" and "Change in Gain vs Reference Voltage" provide more information.

The noise inherent in the converter will also appear to increase with lower LSB size. With a 5V reference, the internal noise of the converter typically contributes only 0.16 LSB peak-to-peak of potential error to the output code. When the external reference is 100mV, the potential error contribution from the internal noise will be 50 times larger—8 LSBs. The errors due to the internal noise are gaussian in nature and can be reduced by averaging consecutive conversion results.

For more information regarding noise, consult the typical performance curves "Effective Number of Bits vs Reference Voltage" and "Peak-to-Peak Noise vs Reference Voltage." The effective number of bits (ENOB) figure is calculated based on the converter's signal-to-(noise + distortion) ratio with a 1kHz, 0dB input signal. SINAD is related to ENOB as follows: SINAD = 6.02 • ENOB +1.76.

With lower reference voltages, extra care should be taken to provide a clean layout including adequate bypassing, a clean power supply, a low-noise reference, and a low-noise input signal. Because the LSB size is lower, the converter will also be more sensitive to external sources of error such as nearby digital signals and electromagnetic interference.

The current that must be provided by the external reference will depend on the conversion result. The current is lowest at full-scale (FFFh) and is typically 23µA at a 200kHz conversion rate (25°C). For the same conditions, the current will increase as the input approaches zero, reaching 50µA at an output result of 000h. The current does not increase linearly, but depends, to some degree, on the bit pattern of the digital output.

The reference current diminishes directly with both conversion rate and reference voltage. As the current from the reference is drawn on each bit decision, clocking the converter more quickly during a given conversion period will not reduce the overall current drain from the reference. The reference current changes only slightly with temperature. See the curves, "Reference Current vs Sample Rate" and "Reference Current vs Temperature" in the Typical Performance Curves section for more information.

DIGITAL INTERFACE

SERIAL INTERFACE

The ADS7816 communicates with microprocessors and other digital systems via a synchronous 3-wire serial interface as shown in Figure 1 and Table I. The DCLOCK signal synchronizes the data transfer with each bit being transmitted on the falling edge of DCLOCK. Most receiving systems will capture the bitstream on the rising edge of DCLOCK. However, if the minimum hold time for D_{OUT} is acceptable, the system can use the falling edge of DCLOCK to capture each bit.

A falling $\overline{CS}$ signal initiates the conversion and data transfer. The first 1.5 to 2.0 clock periods of the conversion cycle are used to sample the input signal. After the second falling DCLOCK edge, D_{OUT} is enabled and will output a LOW value for one clock period. For the next 12 DCLOCK periods, D_{OUT} will output the conversion result, most significant bit first. After the least significant bit (B0) has been output, subsequent clocks will repeat the output data but in a least significant bit first format.

After the most significant bit (B11) has been repeated, D_{OUT} will tri-state. Subsequent clocks will have no effect on the converter. A new conversion is initiated only when $\overline{CS}$ has been taken HIGH and returned LOW.

SYMBOL	DESCRIPTION	MIN	TYP	MAX	UNITS
t_{SMPL}	Analog Input Sample Time	1.5		2.0	DCLOCK Cycles
t_{CONV}	Conversion Time		12		DCLOCK Cycles
f_{SMP}	Throughput Rate			200	kHz
t_{CSS}	$\overline{CS}$ Falling to DCLOCK LOW			0	ns
t_{DSOE}	$\overline{CS}$ Falling to DCLOCK Rising	30			ns
t_{DDO}	DCLOCK Falling to Current D_{OUT} Not Valid	15			ns
t_{DDV}	DCLOCK Falling to Next D_{OUT} Valid		85	150	ns
t_{TDO}	$\overline{CS}$ Rising to D_{OUT} Tri-State		25	50	ns
t_{DE}	DCLOCK Falling to D_{OUT} Enabled		55	100	ns
t_f	D_{OUT} Fall Time		25	100	ns
t_r	D_{OUT} Rise Time		25	100	ns

TABLE I. Timing Specifications –40°C to +85°C.

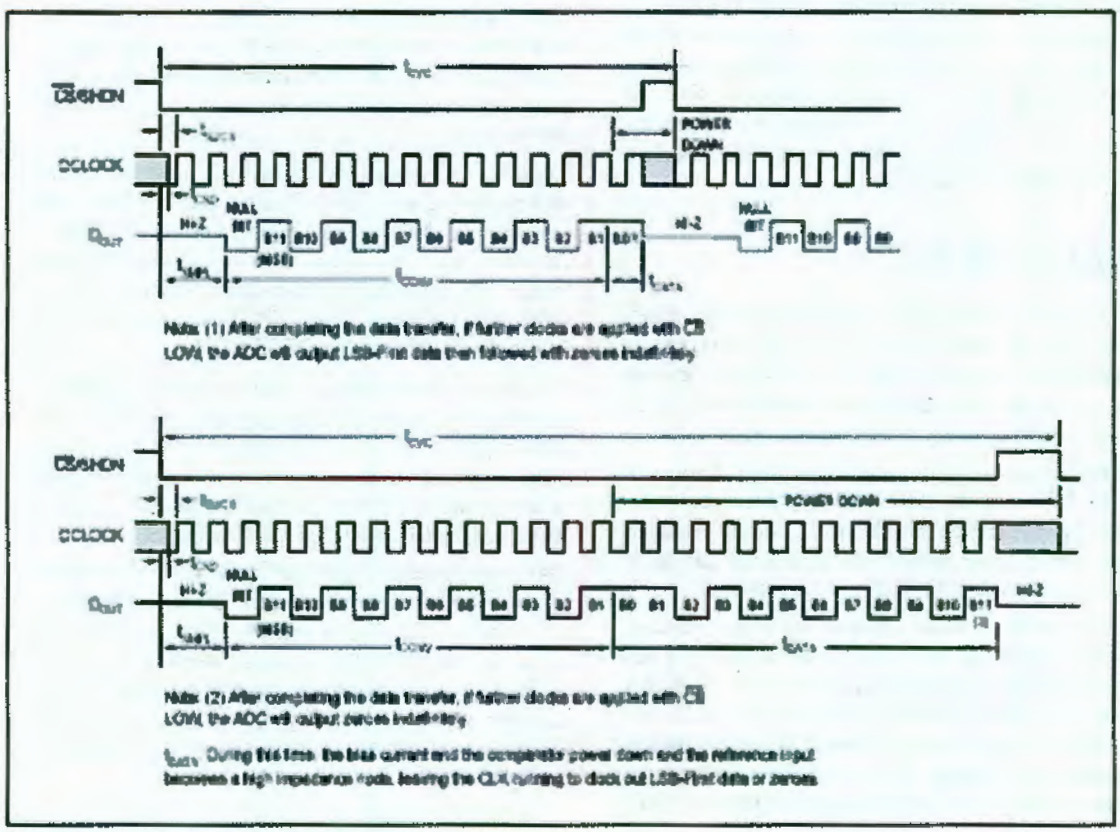


FIGURE 1. ADS7816 Basic Timing Diagrams.

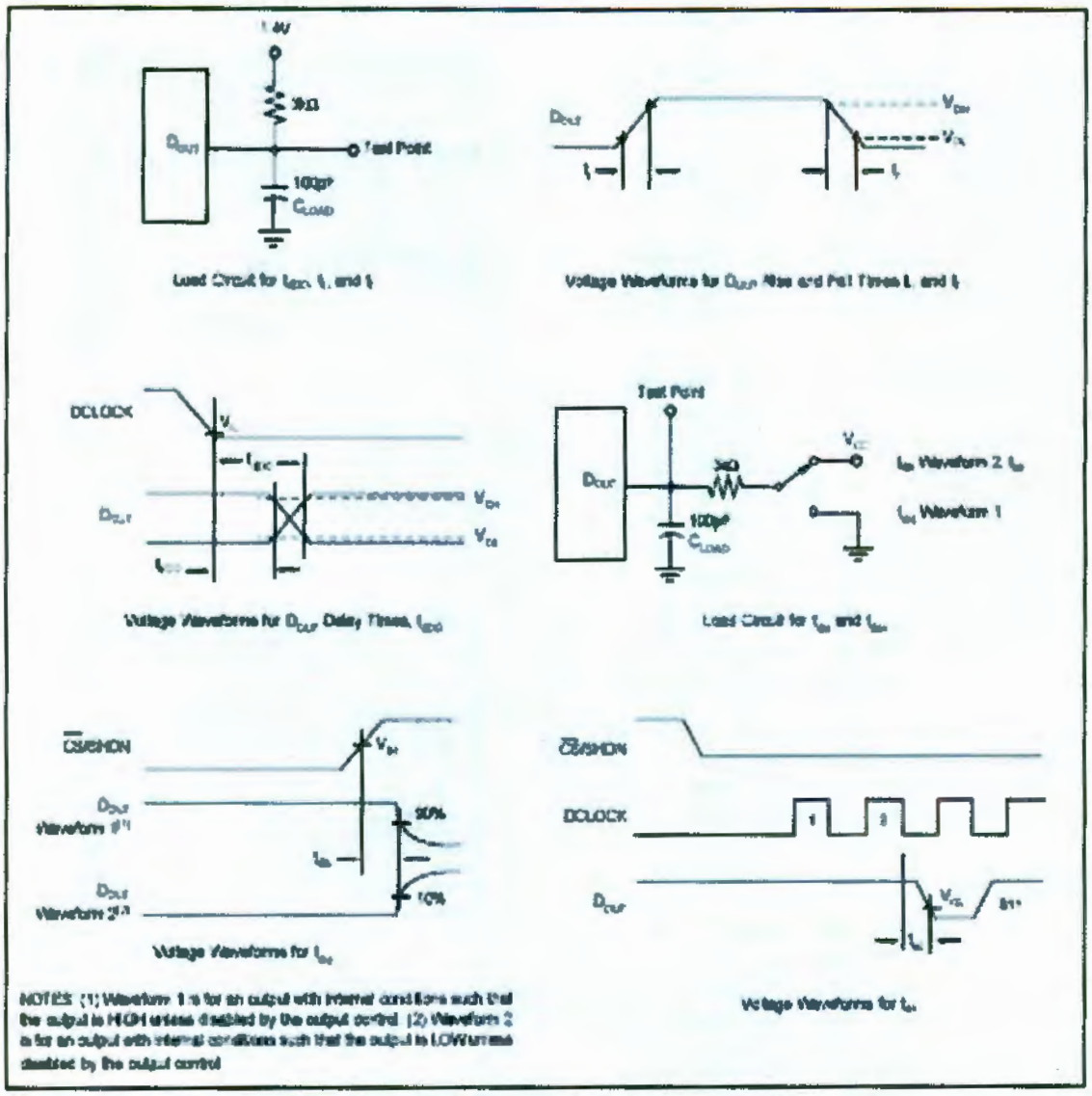


FIGURE 2. Timing Diagrams and Test Circuits for the Parameters in Table I.

DATA FORMAT

The output data from the ADS7816 is in Straight Binary format as shown in Table II. This table represents the ideal output code for the given input voltage and does not include the effects of offset, gain error, or noise.

DESCRIPTION	ANALOG VALUE	DIGITAL OUTPUT STRAIGHT BINARY	
Full Scale Range Least Significant Bit (LSB)	V_{REF} V_{REF}/4096	BINARY CODE	HEX CODE
Full Scale	V_{REF} – 1 LSB	1111 1111 1111	FFF
Midscale	V_{REF}/2	1000 0000 0000	800
Midscale – 1 LSB	V_{REF}/2 – 1 LSB	0111 1111 1111	7FF
Zero	0V	0000 0000 0000	000

Table II. Ideal Input Voltages and Output Codes.

POWER DISSIPATION

The architecture of the converter, the semiconductor fabrication process, and a careful design allow the ADS7816 to convert at up to a 200kHz rate while requiring very little power. Still, for the absolute lowest power dissipation, there are several things to keep in mind.

The power dissipation of the ADS7816 scales directly with conversion rate. The first step to achieving the lowest power dissipation is to find the lowest conversion rate that will satisfy the requirements of the system.

In addition, the ADS7816 is in power down mode under two conditions: when the conversion is complete and whenever $\overline{CS}$ is HIGH (see Figure 1). Ideally, each conversion should occur as quickly as possible, preferably, at a 3.2MHz clock rate. This way, the converter spends the longest possible time in the power down mode. This is very important as the

converter not only uses power on each DCLOCK transition (as is typical for digital CMOS components) but also uses some current for the analog circuitry, such as the comparator. The analog section dissipates power continuously, until the power down mode is entered.

Figure 3 shows the current consumption of the ADS7816 versus sample rate. For this graph, the converter is clocked at 3.2MHz regardless of the sample rate—$\overline{CS}$ is HIGH for the remaining sample period. Figure 4 also shows current consumption versus sample rate. However, in this case, the DCLOCK period is 1/16th of the sample period—$\overline{CS}$ is HIGH for one DCLOCK cycle out of every 16.

There is an important distinction between the power down mode that is entered after a conversion is complete and the full power down mode which is enabled when $\overline{CS}$ is HIGH. While both power down the analog section, the digital section is powered down only when $\overline{CS}$ is HIGH. Thus, if $\overline{CS}$ is left LOW at the end of a conversion and the converter is continually clocked, the power consumption will not be as low as when $\overline{CS}$ is HIGH. See Figure 5 for more information.

By lowering the reference voltage, the ADS7816 requires less current to completely charge its internal capacitors on both the analog input and the reference input. This reduction in power dissipation should be weighed carefully against the resulting increase in noise, offset, and gain error as outlined in the Reference section. The power dissipation of the ADS7816 is reduced roughly 10% when the reference voltage and input range are changed from 5V to 100mV.

SHORT CYCLING

Another way of saving power is to utilize the $\overline{CS}$ signal to short cycle the conversion. Because the ADS7816 places the latest data bit on the D_{OUT} line as it is generated, the converter can easily be short cycled. This term means that the conversion can be terminated at any time. For example, if only 8-bits of the conversion result are needed, then the conversion can be terminated (by pulling $\overline{CS}$ HIGH) after the 8th bit has been clocked out.

This technique can be used to lower the power dissipation (or to increase the conversion rate) in those applications where an analog signal is being monitored until some condition becomes true. For example, if the signal is outside a predetermined range, the full 12-bit conversion result may not be needed. If so, the conversion can be terminated after the first n-bits, where n might be as low as 3 or 4. This results in lower power dissipation in both the converter and the rest of the system, as they spend more time in the power down mode.

LAYOUT

For optimum performance, care should be taken with the physical layout of the ADS7816 circuitry. This is particularly true if the reference voltage is low and/or the conversion rate is high. At 200kHz conversion rate, the ADS7816 makes a bit decision every 312ns. That is, for each subsequent bit decision,

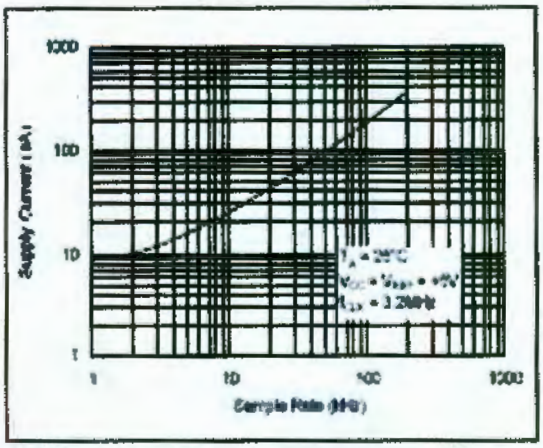


FIGURE 3. Maximizing f_{CLK} at the Highest Possible Rate Allows Supply Current to Drop Directly with Sample Rate.

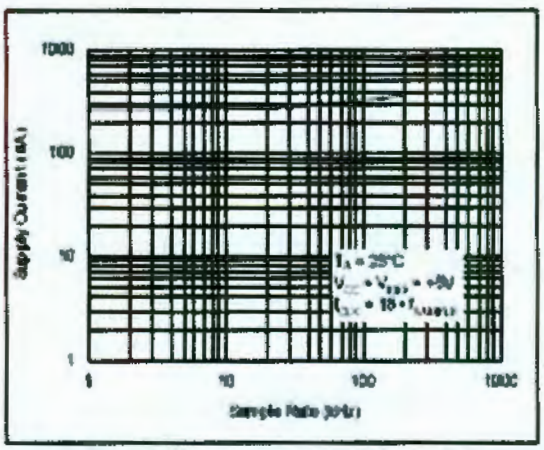


FIGURE 4. Scaling f_{CLK} Reduces Supply Current Only Slightly with Sample Rate.

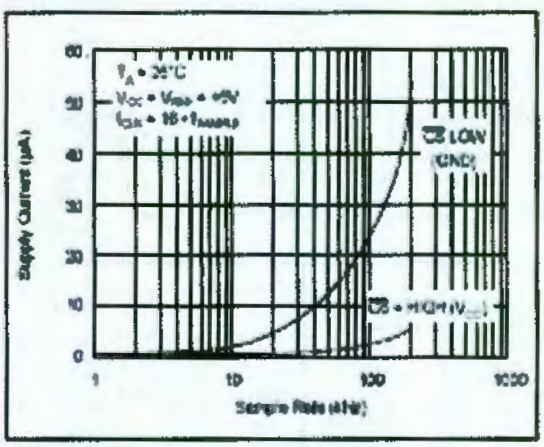


FIGURE 5. Shutdown Current is Considerably Lower with $\overline{CS}$ HIGH than when $\overline{CS}$ is LOW.

tion, the digital output must be updated with the results of the last bit decision, the capacitor array appropriately switched and charged, and the input to the comparator settled to a 12-bit level all within one clock cycle.

The basic SAR architecture is sensitive to spikes on the power supply, reference, and ground connections that occur just prior to latching the comparator output. Thus, during any single conversion for an n-bit SAR converter, there are n "windows" in which large external transient voltages can easily affect the conversion result. Such spikes might originate from switching power supplies, digital logic, and high power devices, to name a few. This particular source of error can be very difficult to track down if the glitch is almost synchronous to the converter's DCLOCK signal—as the phase difference between the two changes with time and temperature, causing sporadic misoperation.

With this in mind, power to the ADS7816 should be clean and well bypassed. A 0.1µF ceramic bypass capacitor should be placed as close to the ADS7816 package as possible. In addition, a 1 to 10µF capacitor and a 10Ω series resistor may be used to lowpass filter a noisy supply.

The reference should be similarly bypassed with a 0.1µF capacitor. Again, a series resistor and large capacitor can be used to lowpass filter the reference voltage. If the reference voltage originates from an op amp, be careful that the op-amp can drive the bypass capacitor without oscillation (the series resistor can help in this case). Keep in mind that while the ADS7816 draws very little current from the reference on average, there are higher instantaneous current demands placed on the external reference circuitry.

Also, keep in mind that the ADS7816 offers no inherent rejection of noise or voltage variation in regards to the reference input. This is of particular concern when the reference input is tied to the power supply. Any noise and ripple from the supply will appear directly in the digital results. While high frequency noise can be filtered out as described in the previous paragraph, voltage variation due to the line frequency (50Hz or 60Hz), can be difficult to remove.

The GND pin on the ADS7816 should be placed on a clean ground point. In many cases, this will be the "analog" ground. Avoid connecting the GND pin too close to the grounding point for a microprocessor, microcontroller, or digital signal processor. If needed, run a ground trace directly from the converter to the power supply connection point. The ideal layout will include an analog ground plane for the converter and associated analog circuitry.

The –In input pin should be connected directly to ground. In those cases where the ADS7816 is a large distance from the signal source and/or the circuit environment contains large EMI or RFI sources, the –In input should be connected to the ground nearest the signal source. This should be done with a signal trace that is adjacent to the +In input trace. If appropriate, coax cable or twisted-pair wire can be used.

APPLICATION CIRCUITS

Figures 6, 7, and 8 show some typical application circuits for the ADS7816. Figure 6 uses an ADS7816 and a multiplexer to provide for a flexible data acquisition circuit. A resistor string provides for various voltages at the multiplexer input. The selected voltage is buffered and driven into V_{REF}. As shown in Figure 6, the input range of the ADS7816 is programmable to 100mV, 200mV, 300mV, or 400mV. The 100mV range would be useful for sensors such as the thermocouple shown.

Figure 7 is more complex variation of Figure 6 with increased flexibility. In this circuit, a digital signal processor designed for audio applications is put to use in running three ADS7816s and a DAC56. The DAC56 provides a variable voltage for V_{REF}—enabling the input range of the ADS7816s to be programmed from 100mV to 3V.

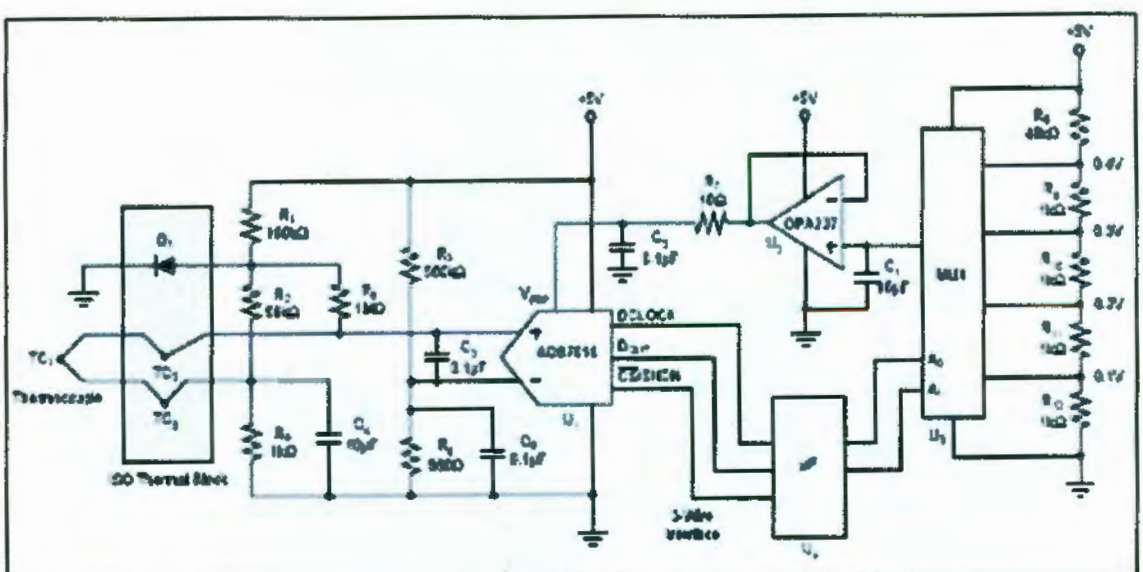


FIGURE 6. Thermocouple Application Using a MUX to Scale the Input Range of the ADS7816

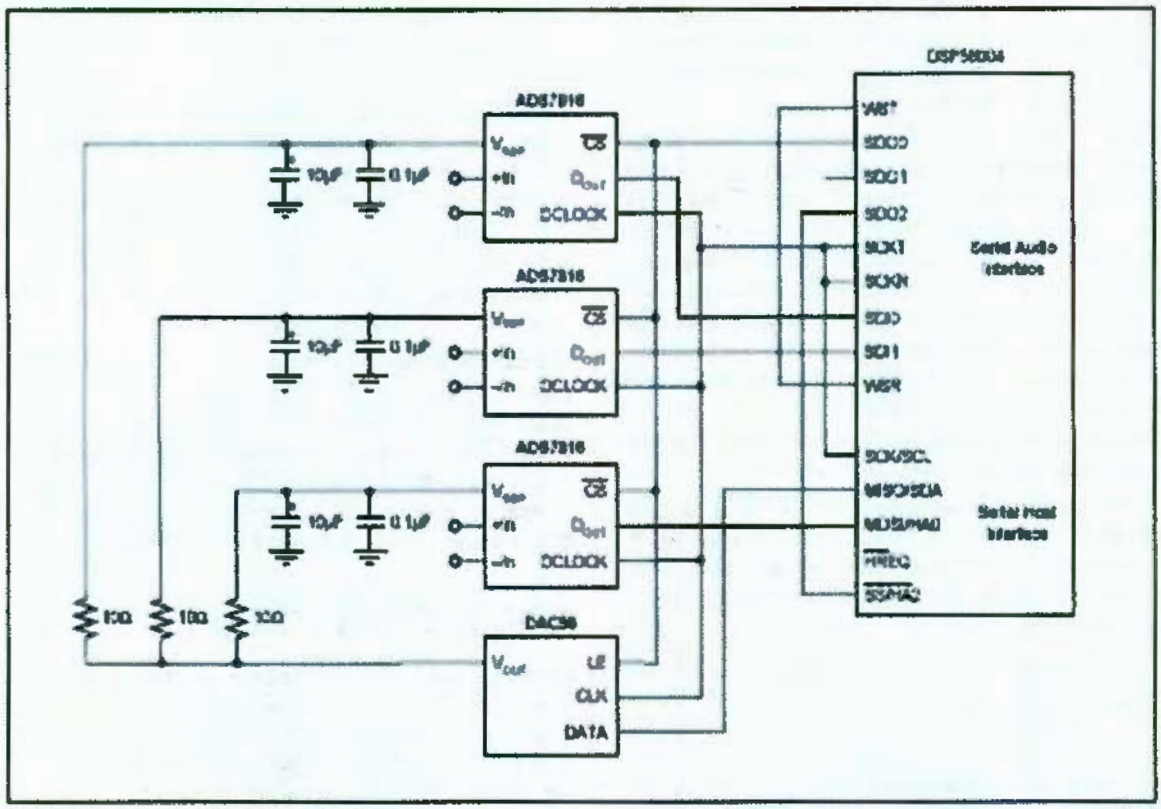


FIGURE 7. Flexible Data Acquisition System.

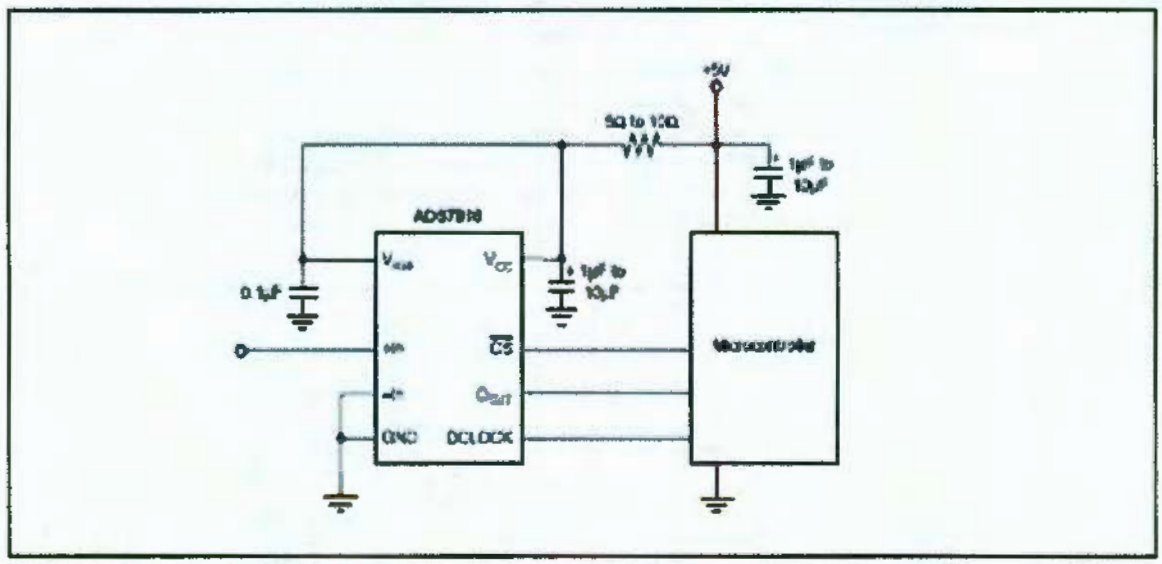


FIGURE 8. Basic Data Acquisition System.

The ADS7816s and the DSP56004 can all be placed into a power down mode. Or, the DSP56004 can run the ADS7816s at a full 3.2MHz clock rate while on-board software enables the ADS7816s as needed. With additional glue logic, the DSP56004 could be used to run multiple DAC56s or provide $\overline{CS}$ controls for each of the three ADS7816s.

Figure 8 shows a basic data acquisition system. The ADS7816 input range is 0V to 5V, as the reference input is connected directly to the +5V supply. The 5Ω to 10Ω resistor and 1µF to 10µF capacitor filter the microcontroller "noise" on the supply, as well as any high-frequency noise from the supply itself. The exact values should be picked such that the filter provides adequate rejection of the noise.

PACKAGING INFORMATION

Orderable Device	Status [1]	Package Type	Package Drawing	Pins	Package Qty	Eco Plan [2]	Lead/Ball Finish	MSL Peak Temp [3]
ADS7816E/250	ACTIVE	MSOP	DGK	8	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR
ADS7816E/250G4	ACTIVE	MSOP	DGK	8	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR
ADS7816E/2K5	ACTIVE	MSOP	DGK	8	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR
ADS7816E/2K5G4	ACTIVE	MSOP	DGK	8	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR
ADS7816EB/250	ACTIVE	MSOP	DGK	8	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR
ADS7816EB/250G4	ACTIVE	MSOP	DGK	8	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR
ADS7816EB/2K5	ACTIVE	MSOP	DGK	8	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR
ADS7816EB/2K5G4	ACTIVE	MSOP	DGK	8	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR
ADS7816EC/250	ACTIVE	MSOP	DGK	8	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR
ADS7816EC/250G4	ACTIVE	MSOP	DGK	8	250	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR
ADS7816EC/2K5	ACTIVE	MSOP	DGK	8	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR
ADS7816EC/2K5G4	ACTIVE	MSOP	DGK	8	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR
ADS7816P	ACTIVE	PDIP	P	8	50	Green (RoHS & no Sb/Br)	CU NIPDAU	N / A for Pkg Type
ADS7816PB	ACTIVE	PDIP	P	8	50	Green (RoHS & no Sb/Br)	CU NIPDAU	N / A for Pkg Type
ADS7816PBG4	ACTIVE	PDIP	P	8	50	Green (RoHS & no Sb/Br)	CU NIPDAU	N / A for Pkg Type
ADS7816PC	ACTIVE	PDIP	P	8	50	Green (RoHS & no Sb/Br)	CU NIPDAU	N / A for Pkg Type
ADS7816PCG4	ACTIVE	PDIP	P	8	50	Green (RoHS & no Sb/Br)	CU NIPDAU	N / A for Pkg Type
ADS7816PG4	ACTIVE	PDIP	P	8	50	Green (RoHS & no Sb/Br)	CU NIPDAU	N / A for Pkg Type
ADS7816U	ACTIVE	SOIC	D	8	100	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR
ADS7816U/2K5	ACTIVE	SOIC	D	8	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR
ADS7816U/2K5G4	ACTIVE	SOIC	D	8	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR
ADS7816UB	ACTIVE	SOIC	D	8	100	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR
ADS7816UB/2K5	ACTIVE	SOIC	D	8	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR
ADS7816UB/2K5G4	ACTIVE	SOIC	D	8	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR
ADS7816UBG4	ACTIVE	SOIC	D	8	100	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR

Orderable Device	Status [1]	Package Type	Package Drawing	Pins	Package Qty	Eco Plan [2]	Lead/Ball Finish	MSL Peak Temp [3]
ADS7816UC	ACTIVE	SOIC	D	8	100	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR
ADS7816UC/2K5	ACTIVE	SOIC	D	8	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR
ADS7816UC/2K5G4	ACTIVE	SOIC	D	8	2500	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR
ADS7816UCG4	ACTIVE	SOIC	D	8	100	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR
ADS7816UC/4	ACTIVE	SOIC	D	8	100	Green (RoHS & no Sb/Br)	CU NIPDAU	Level-2-260C-1 YEAR

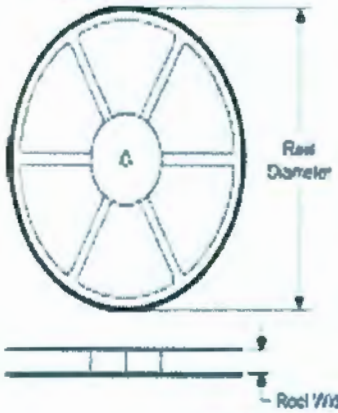
[1] The marketing status values are defined as follows:
ACTIVE: Product device recommended for new designs.
LIFEBUY: TI has announced that the device will be discontinued, and a lifetime-buy period is in effect.
NRND: Not recommended for new designs. Device is in production to support existing customers, but TI does not recommend using this part in a new design.
PREVIEW: Device has been announced but is not in production. Samples may or may not be available.
OBSOLETE: TI has discontinued the production of the device.

[2] Eco Plan - The planned eco-friendly classification: Pb-Free (RoHS), Pb-Free (RoHS Exempt), or Green (RoHS & no Sb/Br) - please check http://www.ti.com/productcontent for the latest availability information and additional product content details.
TBD: The Pb-Free/Green conversion plan has not been defined.
Pb-Free (RoHS): TI's terms "Lead-Free" or "Pb-Free" mean semiconductor products that are compatible with the current RoHS requirements for all 6 substances, including the requirement that lead not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, TI Pb-Free products are suitable for use in specified lead-free processes.
Pb-Free (RoHS Exempt): This component has a RoHS exemption for either 1) lead-based flip-chip solder bumps used between the die and package, or 2) lead-based die adhesive used between the die and leadframe. The component is otherwise considered Pb-Free (RoHS compatible) as defined above.
Green (RoHS & no Sb/Br): TI defines "Green" to mean Pb-Free (RoHS compatible), and free of Bromine (Br) and Antimony (Sb) based flame retardants (Br or Sb do not exceed 0.1% by weight in homogeneous material)

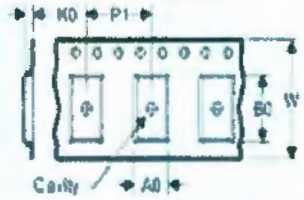
[3] MSL, Peak Temp. -- The Moisture Sensitivity Level rating according to the JEDEC industry standard classifications, and peak solder temperature.

TAPE AND REEL INFORMATION

REEL DIMENSIONS

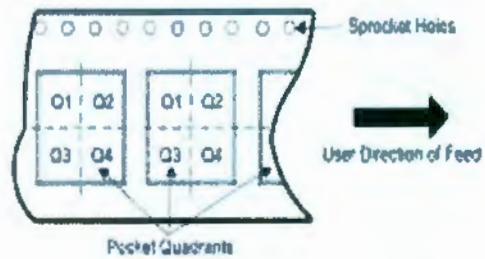


TAPE DIMENSIONS



A0	Dimension designed to accommodate the component width
B0	Dimension designed to accommodate the component length
K0	Dimension designed to accommodate the component thickness
W	Overall width of the carrier tape
P1	Pitch between successive cavity centers

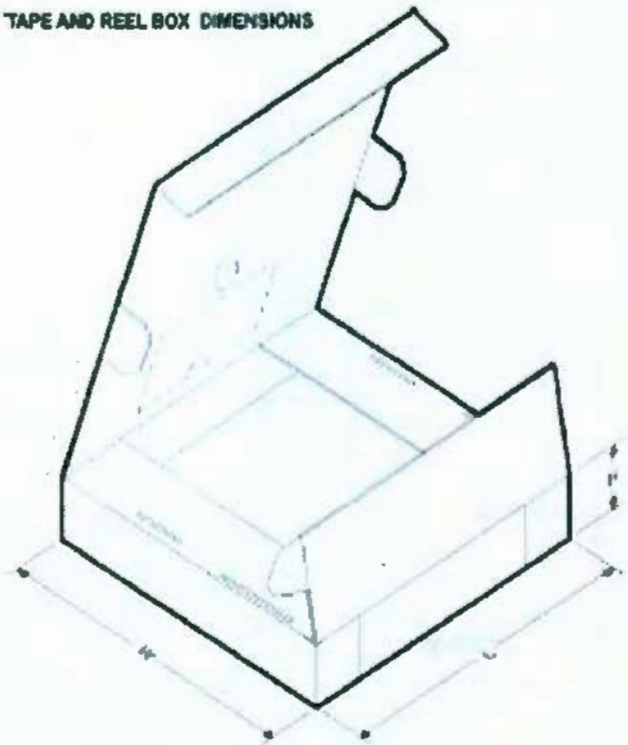
QUADRANT ASSIGNMENTS FOR PIN 1 ORIENTATION IN TAPE



*All dimensions are nominal

Device	Package Type	Package Drawing	Pins	SPQ	Reel Diameter (mm)	Reel Width W1 (mm)	A0 (mm)	B0 (mm)	K0 (mm)	P1 (mm)	W (mm)	Pin1 Quadrant
ADS7816E/250	MSOP	DGK	8	250	330.0	12.4	5.3	3.4	1.4	8.0	12.0	Q1
ADS7816E/2K5	MSOP	DGK	8	2500	330.0	12.4	5.3	3.4	1.4	8.0	12.0	Q1
ADS7816EB/250	MSOP	DGK	8	250	330.0	12.4	5.3	3.4	1.4	8.0	12.0	Q1
ADS7816EB/2K5	MSOP	DGK	8	2500	330.0	12.4	5.3	3.4	1.4	8.0	12.0	Q1
ADS7816ED/250	MSOP	DGK	8	250	330.0	12.4	5.3	3.4	1.4	8.0	12.0	Q1
ADS7816ED/2K5	MSOP	DGK	8	2500	330.0	12.4	5.3	3.4	1.4	8.0	12.0	Q1
ADS7816J/2K5	SOIC	D	8	2500	330.0	12.4	6.4	5.2	2.1	8.0	12.0	Q1
ADS7816JB/2K5	SOIC	D	8	2500	330.0	12.4	6.4	5.2	2.1	8.0	12.0	Q1
ADS7816UC/2K5	SOIC	D	8	2500	330.0	12.4	6.4	5.2	2.1	8.0	12.0	Q1

TAPE AND REEL BOX DIMENSIONS



*All dimensions are nominal

Device	Package Type	Package Drawing	Pins	SPQ	Length (mm)	Width (mm)	Height (mm)
ADS7816E/250	MSOP	DGK	8	250	346.0	346.0	29.0
ADS7816E/2K5	MSOP	DGK	8	2500	346.0	346.0	29.0
ADS7816EB/250	MSOP	DGK	8	250	346.0	346.0	29.0
ADS7816EB/2K5	MSOP	DGK	8	2500	346.0	346.0	29.0
ADS7816EC/250	MSOP	DGK	8	250	346.0	346.0	29.0
ADS7816EC/2K5	MSOP	DGK	8	2500	346.0	346.0	29.0
ADS7816U/2K5	SOIC	D	8	2500	346.0	346.0	29.0
ADS7816UB/2K5	SOIC	D	8	2500	346.0	346.0	29.0
ADS7816UC/2K5	SOIC	D	8	2500	346.0	346.0	29.0

A.6 Hoja de datos para el AD596.

ANALOG DEVICES

Thermocouple Conditioner and Setpoint Controller

AD596*/AD597*

FEATURES
Low Cost
Operates with Type J (AD596) or Type K (AD597) Thermocouples
Built-In Ice Point Compensation
Temperature Proportional Operation – 10 mV/°C
Temperature Setpoint Operation – ON/OFF
Programmable Switching Hysteresis
High Impedance Differential Input

GENERAL DESCRIPTION
The AD596/AD597 is a monolithic temperature setpoint controller that has been optimized for use at elevated temperatures such as those found in oven control applications. The device cold junction compensates and amplifies a type J or K thermocouple input to derive an internal signal proportional to temperature. The internal signal is then compared with an externally applied setpoint voltage to yield a low impedance switched output voltage. Dead-Band or switching hysteresis can be programmed using a single external resistor. Alternately, the AD596/AD597 can be configured to provide a voltage output (10 mV/°C) directly from a type J or K thermocouple signal. It can also be used as a stand-alone voltage output temperature sensor.

The AD596/AD597 can be powered with a single supply from +5 V to +36 V, or dual supplies up to a total span of 36 V. Typical quiescent supply current is 160 μA, which minimizes self-heating errors.

The AD596/AD597 H package option includes a thermocouple failure alarm that indicates an open thermocouple lead when operated in the temperature proportional measurement mode. The alarm output has a flexible format which can be used to drive relays, LEDs or TTL logic.

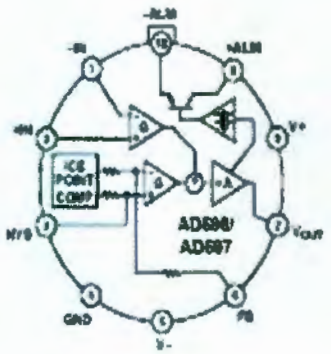
The device is packaged in a reliability qualified, cost effective 10-pin metal can or SOIC and is trimmed to operate over an ambient temperature range from +25°C to +100°C. Operation over an extended ambient temperature range is possible with slightly reduced accuracy. The AD596 will amplify thermocouple signals covering the entire -208°C to +760°C temperature range recommended for type J thermocouples while the AD597 can accommodate -200°C to +1250°C type K inputs.

The AD596/AD597 has a calibration accuracy of ±4°C at an ambient temperature of 60°C and an ambient temperature stability specification of 0.05°C/°C from +25°C to +100°C. If higher accuracy, or a lower ambient operating temperature is required, either the AD594 (J thermocouple) or AD595 (K thermocouple) should be considered.

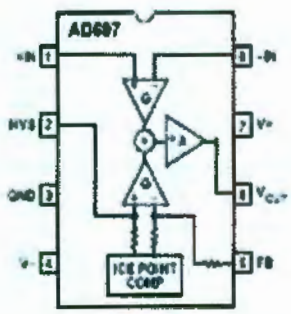
*Protected by U.S. Patent No. 4,029,974.

FUNCTIONAL BLOCK DIAGRAM

TO-100



SOIC



PRODUCT HIGHLIGHTS
1. The AD596/AD597 provides cold junction compensation and a high gain amplifier which can be used as a setpoint comparator.
2. The input stage of the AD596/AD597 is a high quality instrumentation amplifier that allows the thermocouple to float over most of the supply voltage range.
3. Linearization not required for thermocouple temperatures close to 175°C (+100°C to +540°C for AD596).
4. Cold junction compensation is optimized for ambient temperatures ranging from +25°C to +100°C.
5. In the stand-alone mode, the AD596/AD597 produces an output voltage that indicates its own temperature.

REV. B

One Technology Way, P.O. Box 9106, Norwood, MA 02062-9106, U.S.A.
Tel: 781/329-4700 World Wide Web Site: http://www.analog.com
Fax: 781/326-8703

AD596/AD597–SPECIFICATIONS (@ +60°C and V_S = 10 V, Type J (AD596), Type K (AD597) Thermocouple, unless otherwise noted)

Model	AD596AH			AD597AH			AD597AR			Units
	Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
ABSOLUTE MAXIMUM RATINGS										
$+V_S$ to $-V_S$			36			36			36	Volts
Common-Mode Input Voltage	$(-V_S - 0.15)$		$+V_S$	$(-V_S - 0.15)$		$+V_S$	$(-V_S - 0.15)$		$+V_S$	Volts
Differential Input Voltage	$-V_S$		$+V_S$	$-V_S$		$+V_S$	$-V_S$		$+V_S$	Volts
Alarm Voltages										
+ALM	$-V_S$		$(-V_S + 36)$	$-V_S$		$(-V_S + 36)$	$-V_S$		$(-V_S + 36)$	Volts
-ALM	$-V_S$		$+V_S$	$-V_S$		$+V_S$	$-V_S$		$+V_S$	Volts
Operating Temperature Range	-55		+125	-55		+125	-40		+125	°C
Output Short Circuit to Common	Indefinite			Indefinite			Indefinite			
TEMPERATURE MEASUREMENT										
(Specified Temperature Range +25°C to +100°C)										
Calibration Error[1]	-4		+4	-4		+4	-4		+4	°C
Stability vs. Temperature[2]		±0.02	±0.05		±0.02	±0.05		±0.02	±0.05	°C/°C
Gain Error	-1.5		+1.5	-1.5		+1.5	-1.5		+1.5	%
Nominal Transfer Function		10			10			10		mV/°C
AMPLIFIER CHARACTERISTICS										
Closed Loop Gain[3]		180.6			245.5			245.5		V/V
Input Offset Voltage		°C × 53.21 + 235			°C × 41.27 - 37			°C × 41.27 - 37		µV
Input Bias Current		0.1			0.1			0.1		µA
Differential Input Range	-10		+50	-10		+50	-10		+50	mV
Common-Mode Range	$(-V_S - 0.15)$		$(+V_S - 4)$	$(+V_S - 0.15)$		$(+V_S - 4)$	$(-V_S - 0.15)$		$(+V_S - 4)$	Volts
Common-Mode Sensitivity-RTO			10			10			10	mV/V
Power Supply Sensitivity-RTO		1	10		1	10		1	10	mV/V
Output Voltage Range										
Dual Supplies	$(-V_S + 2.5)$		$(+V_S - 2)$	$(-V_S + 2.5)$		$(+V_S - 2)$	$(-V_S + 2.5)$		$(+V_S - 2)$	Volts
Single Supply	0		$(+V_S - 2)$	0		$(+V_S - 2)$	0		$(+V_S - 2)$	Volts
Linear Output Current[4]	±5			±5			±5			mA
3 dB Bandwidth		15			15			15		kHz
ALARM CHARACTERISTICS[5]							Alarm Function Not Pinned Out			
$V_{OUT(ON)}$ at 2 mA		0.3			0.3					Volts
Leakage Current			±1			±1				µA
Operating Voltage at -ALM			$(+V_S - 4)$			$(+V_S - 4)$				Volts
Short Circuit Current		20			20					mA
POWER REQUIREMENTS										
Operating		$(+V_S$ to $-V_S) \le 30$			$(+V_S$ to $-V_S) \le 30$			$(+V_S$ to $-V_S) \le 30$		Volts
Quiescent Current										
$+V_S$		160	300		160	300		160	300	µA
$-V_S$		100	200		100	200		100	200	µA

NOTES

[1]This is a measure of the deviation from ideal with a measuring thermocouple junction at 175°C and a chip temperature of 60°C. The ideal transfer function is given by:
AD596: $V_{OUT} = 180.57 \times (V_{th} - V_a + (\text{ambient in °C}) \times 53.21\ \mu V/°C - 235\ \mu V)$
AD597: $V_{OUT} = 245.46 \times (V_{th} - V_a + (\text{ambient in °C}) \times 41.27\ \mu V/°C - 37\ \mu V)$
where V_{th} and V_a represent the measuring and ambient temperatures and are taken from the appropriate J or K thermocouple table. The ideal transfer function minimizes the error over the ambient temperature range of 25°C to 100°C with a thermocouple temperature of approximately 175°C.
[2]Defined as the slope of the line connecting the AD596/AD597 C)C errors measured at 25°C and 100°C ambient temperature.
[3]Pin 5 shorted to Pin 7.
[4]Current Sink Capability in single supply configuration is limited to current drawn to ground through a 50 kΩ resistor or output voltage below 2.5 V.
[5]Alarm function available on H package option only.

Specifications subject to change without notice.
Specifications shown in boldface are tested on all production units at final electrical test. Results from those tests are used to calculate outgoing quality levels. All min and max specifications are guaranteed, although only those shown in boldface are tested on all production units.

ORDERING GUIDE

Model	Package Description	Package Options
AD596AH	TO-100	H-10A
AD597AH	TO-100	H-10A
AD597AR*	Plastic SOIC	SO-8

*Consult factory for availability.

Table I. Output Voltage vs. Thermocouple Temperature (Ambient +69°C, V_S = –5 V, +15 V)

Thermocouple Temperature °C	Type J Voltage mV	AD596 Output mV	Type K Voltage mV	AD597 Output mV
-200	-7.890	-1370	-5.891	-1446
-180	-7.402	-1282	-5.550	-1362
-160	-6.821	-1177	-5.141	-1262
-140	-6.159	-1058	-4.669	-1148
-120	-5.426	-925	-4.138	-1016
-100	-4.632	-782	-3.553	-872
-80	-3.785	-629	-2.920	-717
-60	-2.892	-468	-2.243	-551
-40	-1.960	-299	-1.527	-375
-20	-.995	-125	-.777	-191
-10	-.501	-36	-.392	-96
0	0	54	0	0
10	.507	146	.397	97
20	1.019	238	.798	198
25	1.277	285	1.000	245
30	1.536	332	1.203	296
40	2.058	426	1.611	396
50	2.585	521	2.022	496
60	3.115	617	2.436	594
80	4.186	810	3.266	802
100	5.268	1006	4.095	1005
120	6.359	1203	4.919	1207
140	7.457	1401	5.733	1407
160	8.562	1600	6.539	1605
180	9.667	1900	7.338	1801
200	10.777	2000	8.137	1997
220	11.887	2201	8.938	2194
240	12.998	2401	9.745	2392
260	14.108	2602	10.560	2592
280	15.217	2902	11.381	2794
300	16.325	3002	12.209	2996
320	17.432	3202	13.040	3201
340	18.537	3402	13.874	3406
360	19.640	3601	14.712	3611
380	20.743	3800	15.552	3817
400	21.846	3999	16.395	4024
420	22.949	4198	17.241	4232
440	24.054	4398	18.088	4440
460	25.161	4598	18.938	4649
480	26.272	4798	19.788	4857

Thermocouple Temperature °C	Type J Voltage mV	AD596 Output mV	Type K Voltage mV	AD597 Output mV
500	27.388	5000	20.640	5068
520	28.511	5203	21.492	5276
540	29.642	5407	22.346	5485
560	30.782	5613	23.198	5694
580	31.933	5821	24.050	5903
600	33.096	6031	24.902	6112
620	34.273	6243	25.751	6321
640	35.464	6458	26.599	6529
660	36.671	6676	27.445	6737
680	37.893	6897	28.289	6944
700	39.130	7120	29.128	7150
720	40.382	7346	29.965	7355
740	41.647	7575	30.799	7560
750	42.293	7689	31.214	7662
760	-	-	31.629	7764
780	-	-	32.455	7968
800	-	-	33.277	8168
820	-	-	34.096	8369
840	-	-	34.909	8569
860	-	-	35.718	8767
880	-	-	36.524	8965
900	-	-	37.325	9162
920	-	-	38.122	9357
940	-	-	38.915	9552
960	-	-	39.703	9745
980	-	-	40.488	9938
1000	-	-	41.269	10130
1020	-	-	42.045	10321
1040	-	-	42.817	10510
1060	-	-	43.585	10699
1080	-	-	44.436	10886
1100	-	-	45.108	11072
1120	-	-	45.863	11258
1140	-	-	46.612	11441
1160	-	-	47.356	11624
1180	-	-	48.096	11805
1200	-	-	48.828	11985
1220	-	-	49.556	12164
1240	-	-	50.276	12341
1250	-	-	50.633	12428

TEMPERATURE PROPORTIONAL OUTPUT MODE

The AD596/AD597 can be used to generate a temperature proportional output of 10 mV/°C when operated with J and K type thermocouples as shown in Figure 1. Thermocouples produce low level output voltages which are a function of both the temperature being measured and the reference or cold junction temperature. The AD596/AD597 compensates for the cold junction temperature and amplifies the thermocouple signal to produce a high level 10 mV/°C voltage output which is a function only of the temperature being measured. The temperature stability of the part indicates the sensitivity of the output voltage to changes in ambient or device temperatures. This is typically 0.02°C/°C over the +25°C to +100°C recommended ambient temperature range. The parts will operate over the extended ambient temperature ranges from -55°C to +125°C, but thermocouple nonlinearity at the reference junction will degrade the temperature stability over this extended range. Table I is a list of ideal AD596/AD597 output voltages as a function of Celsius temperature for type J and K ANSI standard thermocouples with package and reference junction at 0°C. As is normally the case, these outputs are subject to calibration and temperature sensitivity errors. These tables are derived using the ideal transfer functions:

AD596 output = (Type J voltage ÷ 301.5 μV) × 180.57
AD597 output = (Type K voltage) × 245.46

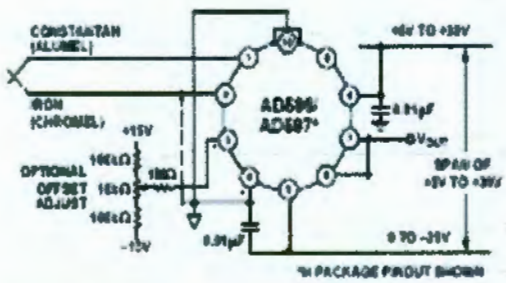


Figure 1. Temperature Proportional Output Connection

The offsets and gains of these devices have been laser trimmed to closely approximate thermocouple characteristics over measurement temperature ranges centered around 175°C with the AD596/AD597 at an ambient temperature between 25°C and 100°C. This eliminates the need for additional gain or offset adjustments to make the output voltage read:

V_{OUT} = 10 mV/°C × (thermocouple temperature in °C) (within specified tolerances).

Excluding calibration errors, the above transfer function is accurate to within 1°C from +80°C to +550°C for the AD596 and -20°C to +350°C for the AD597. The different temperature ranges are due to the differences in J and K type thermocouple curves.

European DIN FE-CuNi thermocouple vary slightly from ANSI type J thermocouples. Table I does not apply when these types of thermocouples are used. The transfer functions given previously and a thermocouple table should be used instead.

Figure 1 also shows an optional trimming network which can be used to change the device's offset voltage. Injecting or sinking 200 nA from Pin 3 will offset the output approximately 10 mV (1°C).

The AD596/AD597 can operate from a single supply from 5 V to 36 V or from split supplies totalling 36 V or less as shown. Since the output can only swing to within 2 V of the positive supply, the usable measurement temperature range will be restricted when positive supplies less than 15 V for the AD597 and 10 V for the AD596 are used. If the AD596/AD597 is to be used to indicate negative Celsius temperatures, then a negative supply is required.

Common-mode voltages on the thermocouple inputs must remain within the common-mode voltage range of the AD596/AD597, with a return path provided for the bias currents. If the thermocouple is not remotely grounded, then the dotted line connection shown in Figure 1 must be made to one of the thermocouple inputs. If there is no return path for the bias currents, the input stage will saturate, causing erroneous output voltages.

In this configuration, the AD596/AD597 H package option has circuitry which detects the presence of an open thermocouple. If the thermocouple loop becomes open, one or both of the inputs to the device will be deprived of bias current causing the output to saturate. It is this saturation which is detected internally and used to activate the alarm circuitry. The output of this feature has a flexible format which can be used to source or sink up to 20 mA of current. The collector (+ALM) should not be allowed to become more positive than ($-V_S$ + 36 V), however, it may be permitted to be more positive than $+V_S$. The emitter voltage (-ALM) should be constrained such that it does not become more positive than 4 V below $+V_S$. If the alarm feature is not used, this pin should be connected to Pins 4 or 5 as shown in Figure 1. The alarm function is unavailable on the AR package option.

SETPOINT CONTROL MODE

The AD596/AD597 can be connected as a setpoint controller as shown in Figure 2. The thermocouple voltage is cold junction compensated, amplified, and compared to an external setpoint voltage. The relationship between setpoint voltage and temperature is given in Table I. If the temperature to be controlled is within the operating range (-55°C to +125°C) of the device, it can monitor its own temperature by shorting the inputs to ground. The setpoint voltage with the thermocouple inputs grounded is given by the expressions:

AD596 Setpoint Voltage = °C × 9.6 mV/°C + 42 mV
AD597 Setpoint Voltage = °C × 10.1 mV/°C - 9.1 mV

The input impedance of the setpoint pin of the AD596/AD597 is approximately 50 kΩ. The temperature coefficient of this resistance is ±15 ppm/°C. Therefore, the 100 ppm/°C 5 kΩ pot shown in Figure 2 will only introduce an additional ±1°C degradation of temperature stability over the +25°C to +100°C ambient temperature range.

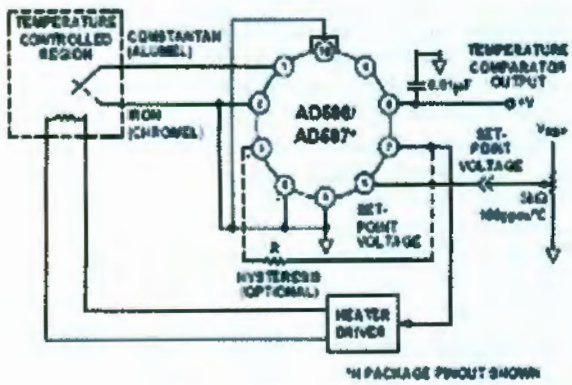


Figure 2. Setpoint Control Mode

Switching hysteresis is often used in setpoint systems of this type to provide noise immunity and increase system reliability. By reducing the frequency of on-off cycling, mechanical component wear is reduced leading to enhanced system reliability. This can easily be implemented with a single external resistor between Pins 7 and 3 of the AD596/AD597. Each 200 nA of current injected into Pin 3 when the output switches will cause about 1°C of hysteresis; that is:

$$R_{HYST}\ (\Omega) = \frac{V_{OUT}}{200\,nA} \times \frac{1}{^{\circ}C_{HYST}}$$

In the setpoint configuration, the AD596/AD597 output is saturated at all times, so the alarm transistor will be ON regardless of whether there is an open circuit or not. However, -ALM must be tied to a voltage below ($+V_S$ - 4 V) for proper operation of the rest of the circuit.

STAND-ALONE TEMPERATURE TRANSDUCER

The AD596/AD597 may be configured as a stand-alone Celsius thermometer as shown in Figure 3.

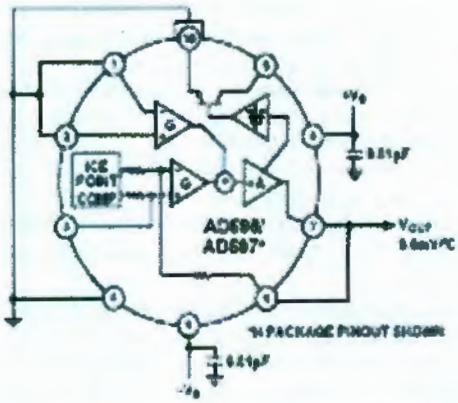


Figure 3. Stand-Alone Temperature Transducer Temperature Proportional Output Connection

Simply omit the thermocouple and connect the inputs (Pins 1 and 2) to common. The output will now reflect the compensation voltage and hence will indicate the AD596/AD597 temperature. In this three terminal voltage output, temperature sensing mode, the AD596/AD597 will operate over the full extended -55°C to +125°C temperature range. The output scaling will be 9.6 mV per °C with the AD596 and 10.1 mV per °C with the AD597. Additionally there will be a 42 mV offset with the AD596 causing it to read slightly high when used in this mode.

THERMOCOUPLE CONNECTIONS

The connection of the thermocouple wire and the normal wire or printed circuit board traces going to the AD596/AD597 forms an effective reference junction as shown in Figure 4. This junction must be kept at the same temperature as the AD596/AD597 for the internal cold junction compensation to work properly. Unless the AD596/AD597 is in a thermally stable enclosure, the thermocouple leads should be brought in directly to Pins 1 and 2.

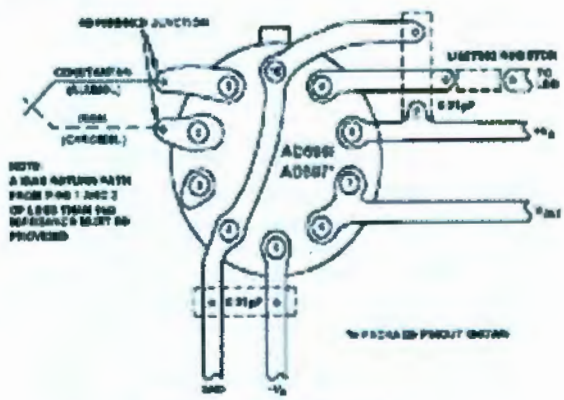


Figure 4. PCB Connections

To ensure secure bonding, the thermocouple wire should be cleaned to remove oxidization prior to soldering. Noncorrosive resin flux is effective with iron, constantan, chromel, and alumel, and the following solders: 95% tin-5% silver, or 90% tin-10% lead.

SINGLE AND DUAL SUPPLY CONNECTIONS

In the single supply configuration as used in the setpoint controller of Figure 2, any convenient voltage from +5 V to +36 V may be used, with self-heating errors being minimized at lower supply levels. In this configuration, the $-V_S$ connection at Pin 5 is tied to ground. Temperatures below zero can be accommodated in the single supply setpoint mode, but not in the single supply temperature measuring mode (Figure 1 reconnected for single supply). Temperatures below zero can only be indicated by a negative output voltage, which is impossible in the single supply mode.

Common-mode voltages on the thermocouple inputs must remain below the positive supply, and not more than 0.15 V more negative than the minus supply. In addition, a return path for the input bias currents must be provided. If the thermocouple is not remotely grounded, then the dotted line connections in Figures 1 and 2 are mandatory.

STABILITY OVER TEMPERATURE

The AD596/AD597 is specified for a maximum error of ±4°C at an ambient temperature of 60°C and a measuring junction temperature at 175°C. The ambient temperature stability is specified to be a maximum of 0.05°C/°C. In other words, for every degree change in the ambient temperature, the output will change no more than 0.05 degrees. So, at 25°C the maximum deviation from the temperature-voltage characteristic of Table I is ±5.75°C, and at 100°C it is ±6°C maximum (see Figure 5). If the offset error of ±4°C is removed with a single offset adjustment, these errors will be reduced to ±1.75°C and ±2°C max. The optional trim circuit shown in Figure 1 demonstrates how the ambient offset error can be adjusted to zero.

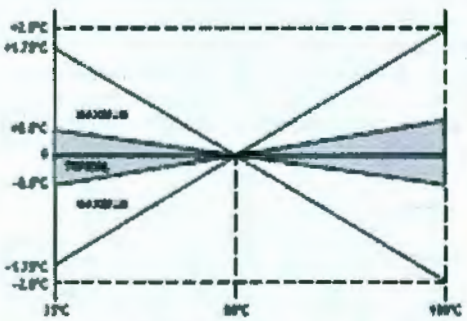


Figure 5. Drift Error vs. Temperature

THERMAL ENVIRONMENTAL EFFECTS

The inherent low power dissipation of the AD596/AD597 keeps self-heating errors to a minimum. However, device output is capable of delivering ±5 mA to an external load and the alarm circuitry can supply up to 20 mA. Since the typical junction to ambient thermal resistance in free air is 150°C/W, significant temperature difference between the package pins (where the reference junction is located) and the chip (where the cold junction temperature is measured and then compensated) can exist when the device is operated in a high dissipation mode. These temperature differences will result in a direct error at the output. In the temperature proportional mode, the alarm feature will only activate in the event of an open thermocouple or system transient which causes the device output to saturate. Self-Heating errors will not effect the operation of the alarm but two cases do need to be considered. First, after a fault is corrected and the alarm is reset, the AD596/AD597 must be allowed to cool before readings can again be accurate. This can take 5 minutes or more depending upon the thermal environment seen by the device. Second, the junction temperature of the part should not be allowed to exceed 150°C. If the alarm circuit of the AD596/AD597 is made to source or sink 20 mA with 30 V across it, the junction temperature will be 90°C above ambient causing the die temperature to exceed 150°C when ambient is above 60°C. In this case, either the load must be reduced, or a heat sink used to lower the thermal resistance.

TEMPERATURE READOUT AND CONTROL

Figure 6 shows a complete temperature indication and control system based on the AD596/AD597. Here the AD596/AD597 is being used as a closed-loop thermocouple signal conditioner and an external op amp is used to implement setpoint. This has two important advantages. It provides a high level (10 mV/°C) output for the A/D panel meter and also preserves the alarm function for open thermocouples.

The A/D panel meter can easily be offset and scaled as shown to read directly in degrees Fahrenheit. If a two temperature calibration scheme is used, the dominant residual errors will arise from two sources; the ambient temperature rejection (typically ±2°C over a 25°C to 100°C range) and thermocouple nonlinearity typical +1°C from 80°C to 550°C for type J and +1°C from -20°C to 350°C for type K.

An external voltage reference is used both to increase the stability of the A/D converter and supply a stable reference for the setpoint voltage.

A traditional requirement for the design of setpoint control thermocouple systems has been to configure the system such that the appropriate action is taken in the event of an open thermocouple. The open thermocouple alarm pin with its flexible current-limited output format supports this function when the part operates in the temperature proportional mode. In addition, if the thermocouple is not remotely grounded, it is possible to program the device for either a positive or negative full-scale output in the event of an open thermocouple. This is done by connecting the bias return resistor directly to Pin 1 if a high output voltage is desired to indicate a fault condition. Alternately, if the bias return is provided on the thermocouple lead connected to Pin 2, an open circuit will result in an output low reading. Figure 6 shows the ground return connected to Pin 1 so that if the thermocouple fails, the heater will remain off. At the same time, the alarm circuit lights the LED signalling the need to service the thermocouple. Grounding Pin 2 would lead to low output voltage saturation, and in this circuit would result in a potentially dangerous thermal runaway under fault conditions.

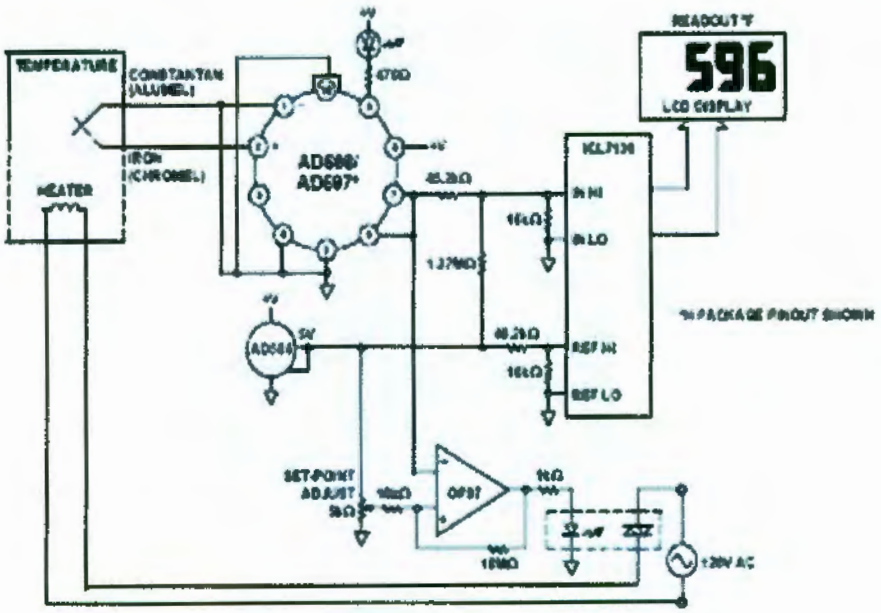
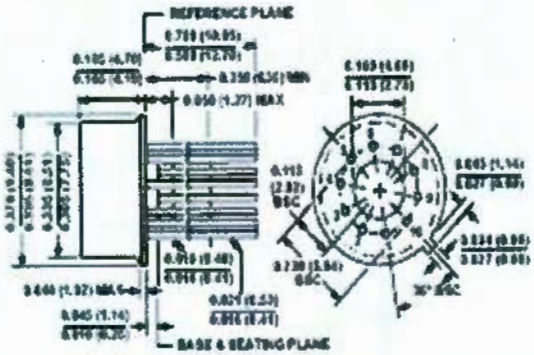


Figure 6. Temperature Measurement and Control

OUTLINE DIMENSIONS

Dimensions shown in inches and (mm).

10-Pin Metal Can
(TO-100)



8-Lead Small Outline (SOIC)
(SO-8)