UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO

FACULTAD DE INGENIERÍA ELÉCTRICA

TESIS

"MEDIDOR DE TEMPERATURA Y PRESIÓN ATMOSFÉRICA BASADO EN UN MICROCONTROLADOR DE BAJO CONSUMO DE POTENCIA"

QUE PRESENTA

Christian Pérez Negrón Pérez

PARA OBTENER EL TÍTULO DE:

INGENIERO ELECTRICISTA

ASESOR DE TESIS

Dr. Norberto García Barriga

Enero del 2008

Resumen

En el presente proyecto de tesis se presenta la implementación de un medidor de temperatura y presión atmosférica basado en un microcontrolador de bajo consumo de potencia, el cual es el núcleo principal de una estación meteorológica. Se presenta el diseño de *hardware* y *software* desarrollado en este proyecto para la aplicación del microcontrolador MSP430F4270. Se propone un sistema mínimo que incluye las conexiones básicas para operar el microcontrolador, un botón para el usuario, una pantalla de cristal líquido, la etapa de acondicionamiento de señales para los sensores y el sistema de baterías. Mientras que el sensor de presión permite conectarse a la terminal del microcontrolador, el sensor de temperatura requiere la utilización de una etapa de acondicionamiento que ajuste los niveles de voltaje a los niveles permitidos. Se proponen dos versiones de programa desarrollados en código C, los cuales realizan el proceso de medición y almacenaje de las variables meteorológicas de forma automática. Dichos programas fueron desarrollados en un estilo modular, con la finalidad de que los tesistas futuros que continúen trabajando en este proyecto puedan adherir los nuevos transductores de variables meteorológicas (propuestos en el capítulo 6), para la culminación de una estación meteorológica completa. Además, se ilustra de forma detallada el procedimiento para programar el microcontrolador con el programa de desarrollo *Workbench* utilizado en este proyecto. La etapa de la programación del microcontrolador es la de mayor importancia de este proyecto de tesis, debido a que este trabajo servirá como base para proyectos futuros en los que se requiera la utilización del microcontrolador MSP430F4270.Los resultados obtenidos con este medidor son validados con una estación meteorológica comercial.

Contenido.

Capítulo 5. Pruebas y Resultados

Capítulo 6. Conclusiones y Trabajos Futuros

Lista de Figuras

Lista de Tablas

Lista de Símbolos y Abreviaturas

Ω	ohms
CD	corriente directa
°C	grados centígrados
mB	mili bares
mmHg	milímetros de mercurio
k	kilo
Hz	hertz
V	volts
mV	mili-volts
µV	micro-volts
A	amperes
mA	mili-amperes
µA	micro-amperes
MB	mega-bytes
RAM	memoria de acceso aleatorio
CPU	unidad central de procesamiento
WDT	temporizador perro guardián
FLL	bucle de frecuencia cerrada
FET	herramienta de emulación *Flash*
ALU	unidad de aritmética lógica
PC	contador del programa
SP	apuntador
SR	registro especial
CG1	generador de constantes 1
CG2	generador de constantes 2
GIE	interrupción general habilitada
SRF	registros de función especial
GPA	amplificador de ganancia programable
ISR	rutina de servicio de interrupción

SMCLK	reloj maestro del subsistema
MCLK	reloj maestro
ACLK	reloj auxiliar
DCO	oscilador digitalmente controlable
MAB	bus de memoria de direccionamiento
MDB	bus de memoria de información
MI	interrupción enmascarable
NMI	interrupción no enmascarable
A/D	convertidor analógico a digital
D/A	convertidor digital a analógico
I/O	entrada ó salida

Capítulo 1

Introducción

En este capítulo se presenta una descripción general de los antecedentes relacionados con el trabajo desarrollado en esta tesis. Se proporciona una descripción del problema a solucionar, los objetivos y la justificación de este proyecto. Además, se presenta la metodología utilizada en este trabajo, resaltando las herramientas utilizadas en el desarrollo de este proyecto. Por último se presenta una descripción breve del contenido por capítulos de este trabajo.

1.1 Antecedentes

La meteorología es el estudio de los fenómenos atmosféricos y de los mecanismos que determinan el estado del tiempo. Los fenómenos atmosféricos ó meteoros pueden ser del tipo aéreo como el viento, acuoso como la lluvia, la nieve y el granizo, luminoso como la aurora polar o el arcoíris y eléctrico como el rayo. La presión, la temperatura y la humedad son los factores climáticos fundamentales en el estudio y predicción del estado del tiempo. La temperatura, sometida a numerosas oscilaciones, se halla condicionada por la latitud y por la altura sobre el nivel del mar [*La Guía Metas* 2005].

El estudio de la atmósfera depende de lo que se conoce como elemento meteorológico, el cual se define como aquella variable o fenómeno atmosférico (temperatura del aire, presión, viento, humedad, tormentas, nieblas, ciclones o anticiclones) que caracteriza el estado del tiempo en un lugar específico y en un tiempo dado. Una estación meteorológica es una instalación destinada a medir y registrar regularmente diversas variables meteorológicas. Los datos obtenidos se utilizan tanto para la elaboración de predicciones meteorológicas a partir de modelos numéricos como para estudios climáticos del lugar en donde son instaladas dichas estaciones. Estas estaciones meteorológicas constan de diferentes instrumentos de medición tales como:

- Termómetro (para la medición de la temperatura).
- Barómetro (para la medición de la presión atmosférica).
- Pluviómetro (para la medición de la cantidad de precipitación).

- Psicrómetro (para la medición de la humedad relativa del aire y la temperatura del punto de rocío).
- Piranómetro (para la medición de la radiación solar).
- Heliógrafo (para la medición de las horas de sol).
- Anemómetro (para la medición de la velocidad del viento).
- Veleta (para la medición de la dirección del viento).
- Ceilómetro (para la medición de la altura de las nubes).

La presión atmosférica depende directamente de la altitud, la cual es medida por un altímetro. Este instrumento permite medir la altura basado en la variación vertical de la presión de la atmósfera. La calibración de su escala está hecha bajo condiciones de atmósfera a nivel del mar y, en consecuencia, sólo indicará valores reales cuando se den a estas condiciones [Herrera 2001].

La temperatura de la Tierra es regulada por la atmósfera, la cual está formada por diferentes capas. Estas capas cambian su densidad y presión a lo largo del día y, por ende, la temperatura de la Tierra está sujeta a dichos cambios [Gil 2006]. La estimación de la temperatura ambiental es de gran importancia debido a su amplio campo de aplicación como estudios de enfermedades, monitoreo medioambiental, pronósticos de epidemias, estimación de las precipitaciones, pronósticos climáticos, usos veterinarios, cambios climáticos y modelación de rendimientos de los cultivos. La temperatura ambiental es comúnmente medida a partir de observaciones sinópticas en estaciones meteorológicas, lo cual indica una dependencia de la distribución espacial e infraestructura regional [*Ciencias de la Tierra y el Espacio* 2007].

Existen diferentes tipos de estaciones meteorológicas, las cuales se clasifican dependiendo del tipo de funcionamiento que tengan. Por ejemplo, existen proyectos de estaciones meteorológicas desarrollados por medio de instrumentación virtual [Lázaro et al. 2001], el cual consiste en un sistema de monitoreo de las variables meteorológicas que mide variables físicas a través de una tarjeta de adquisición de datos para posteriormente ser analizadas y visualizadas en una computadora personal. En el caso de este sistema en particular el lenguaje empleado es LabVIEW, el cual cuenta con una interface gráfica amigable que permite mostrar en tiempo real la evolución de las variables monitoreadas, así

como la implementación de un registro de las mediciones obtenidas y la selección de un periodo de muestreo de los datos a adquirir.

Actualmente existen otras opciones para el diseño de estaciones meteorológicas, cuyo diseño se conoce como "sistema remoto de adquisición de datos" [Hiriart y Colorado 2006]. Este tipo de sistemas realizan la adquisición, procesamiento, envío y/o almacenamiento de los datos capturados. Además, este sistema funciona de manera autónoma, ya que son energizados por medio de bancos de baterías, los cuales son recargados regularmente por medio de paneles solares. El envío de los datos obtenidos es por medio de señales de radio, esto es, son sistemas inalámbricos de gran utilidad para el estudio y predicciones de factores meteorológicos. La visualización de los datos obtenidos se puede realizar por medio de alguna pantalla en tiempo real [Navarro 2003], mediante el estudio de los datos almacenados en una memoria externa ó realizando consultas vía internet [Cruz-Abeyro et al. 2005].

Una estación meteorológica no tiene que ser 100% autónoma, es decir, existen observatorios meteorológicos sinópticos que cuentan con personal que se encargan de tomar datos específicos que no son posibles obtener mediante instrumentos. Por ejemplo, los observadores meteorológicos toman nota de la visibilidad en diferentes áreas, así como características de las nubes como su tipo, altura, la cantidad y la ubicación en tiempos anteriores y actuales. A la recolección de estos datos se les llama observación sinóptica [*RAM* 2005].

Estos sistemas por lo general se encuentran instalados y en operación en la superficie terrestre. Sin embargo, en la actualidad al ser humano le interesa trabajar no sólo en zonas terrestres si no también en zonas marítimas, por lo cual se requiere poder tener predicciones del clima tanto en los mares como en los océanos. Es por ello que también existen estaciones meteorológicas diseñadas para operar en este tipo de condiciones. Este tipo de estaciones son llamadas Estaciones Marinas, las cuales registran las mismas variables que las de la superficie, además de las temperaturas a diferentes profundidades del océano y corrientes oceánicas. Se encuentran instaladas en barcos, boyas y plataformas marinas. Los datos obtenidos se transmiten continuamente hacia los satélites, y de ahí, son retransmitidos hacia la tierra [Celis y Forni 2007].

Una estación meteorológica, es de mucha utilidad en diferentes actividades humanas, como ejemplo se puede mencionar, el uso de una estación meteorológica con fines agrícolas dentro de los invernaderos. La utilidad de una estación meteorológica en esta área, ayuda a tener una buena producción de alimentos en las estaciones del año, que no es común su producción. Es decir, con una estación meteorológica se puede pronosticar las condiciones ambientales y así, por medio de equipos, obtener la temperatura, humedad e iluminación requerida para el cultivo de diferentes alimentos [Facultad de Ingeniería Eléctrica y Electrónica – UNI 2007].

Por otro lado, una estación meteorológica, es un instrumento de vital importancia para proyectos más sofisticados. Por ejemplo, el telescopio solar más grande del mundo, llamado el Telescopio Solar de Tecnología Avanzada (ATST) instalado en la Isla de la Palma (España), incorpora dentro de sus componentes una estación meteorológica para obtener información complementaria a los estudios y pruebas realizadas [Collados y Torres 2002].

Por su parte, el monitoreo de variables meteorológicas es uno de los pasos claves que involucran el desarrollo de proyectos relacionados con energías renovables tales como la energía eólica. En este sentido, el objetivo de un sistema de monitoreo es identificar áreas con viento potencialmente explotables para fines de producción. La creación de bases de datos de variables ambientales tales como velocidad del viento, dirección del viento, temperatura, radiación solar, velocidad del viento vertical y presión barométrica representa una fuente de información muy valiosa para evaluar la factibilidad de futuros proyectos.

El presente proyecto presenta la implementación de un medidor de dos variables ambientales, presión y temperatura utilizando el microcontrolador MSP430F4270. Este microcontrolador cuenta con la característica de seleccionar modos de operación de bajo consumo de potencia. Además, gracias a su capacidad para multiplexar hasta cinco entradas al convertidor A/D de 16 *bits*, es posible monitorear cinco variables meteorológicas.

Los programas que controlan el medidor están implementados en C. Se validan los resultados con una estación meteorológica comercial de la compañía *Davis Instruments* llamada *Vantage Pro2* [*Davis Instruments* 2004].

1.2 Objetivos

El objetivo general de éste proyecto es implementar un medidor de presión y temperatura ambiental que cuente con un microcontrolador, sensores para las diferentes variables atmosféricas a analizar, una etapa de acondicionamiento de señales. Además el medidor debe de contar con un consumo de potencia mínimo con la finalidad de que pueda operar en zonas remotas de forma autónoma. Por lo que los objetivos particulares de este trabajo son:

- Monitorear cada uno de los sensores conectados a las terminales analógicas del convertidor A/D a intervalos de tiempo predeterminados.
- Disminuir al mínimo el consumo de potencia del medidor.
- Determinar la presión barométrica local.
- Determinar la temperatura ambiental local.
- Implementar el código del programa en lenguaje C usando un enfoque modular que permita agregar más sensores.
- Almacenar en la memoria *Flash* del microcontrolador los valores muestreados y asociarles el dato de la hora en que se realizó la captura de los datos.

1.3 Justificación

La implementación de un sistema de monitoreo de variables meteorológicas es hoy en día una necesidad relevante no únicamente para organismos dedicados a predecir e informar el estado del tiempo, si no también es de gran importancia para los ingenieros e investigadores que desarrollan proyectos relacionados con la explotación de recursos renovables. Hay que recordar que para llevar a cabo cualquier proyecto asociado con recursos renovables es necesario contar con la información histórica del comportamiento del recurso renovable. Es decir, es necesario contar con las bases de datos confiables y accesibles que permitan determinar la ubicación y magnitud del recurso.

La utilización de estaciones meteorológicas permite monitorear el comportamiento de las variables de interés. Sin embargo, en la actualidad dichos equipos tienen un costo elevado, lo cual hace imposible que una institución de educación superior pública pudiera establecer una red de monitoreo básica.

Es por ello que en este trabajo se propone desarrollar un sistema automatizado de medición y almacenaje de dos variables meteorológicas: temperatura y presión atmosférica. Este medidor será de bajo costo y sentará las bases para el desarrollo de una estación meteorológica completa que monitoreará tres variables adicionales a las mencionadas las cuales son: velocidad y dirección del viento, radiación solar y precipitación pluvial.

1.4 Metodología

La implementación del sistema de medición involucra la utilización de dos sensores, los cuales convertirán en señales eléctricas los parámetros físicos tales como temperatura ambiental y presión atmosférica. Las señales provenientes de los sensores se ajustan a niveles de voltaje y corriente adecuados por medio de una etapa de acondicionamiento, la cual incluye reguladores de voltaje y amplificadores operacionales del tipo LF356. Se elige para este proyecto un microcontrolador que cuenta con un multiplexor de cinco entradas analógicas, lo cual permite agregar más sensores a las entradas restantes.

El microcontrolador usado en este proyecto es del tipo MSP430F4270 de *Texas Instruments,* el cual tiene tres cualidades que lo hacen ideal para este proyecto:
a) Muy bajo consumo de potencia del orden de 250μA en estado activo, por lo cual es posible operarlo con un banco de baterías recargables, b) cuenta con un convertidor A/D de 16 *bits*, lo cual indica que tiene una muy buena resolución, además de que puede atender hasta 5 entradas analógicas y c) cuenta con memoria *Flash* lo cual es gran ayuda debido a que no es necesario utilizar dispositivos adicionales para realizar el almacenamiento de datos capturados.

Este microcontrolador fue programado en lenguaje C, en un *software* muy versátil llamado *Workbench*, el cual permite generar un archivo .C, que posteriormente es adherido a una espacio de trabajo (*work space*) para ser descargado en la memoria del microcontrolador una vez que este ha sido compilado. En este *software* se pueden visualizar los resultados con una pantalla LCD o bien se pueden monitorear las variables dentro de una ventana que se llama *watch*, en donde se puede observar el comportamiento de cada una de las variables.

El programa del microcontrolador se desarrolla bajo un esquema de interrupciones. Las tareas que atiende el sistema de interrupciones son la conversión A/D de los datos y el

botón del usuario. Se implementa una etapa de acondicionamiento que tiene la función de adecuar los niveles de voltaje y corriente de los sensores que se conectan al microcontrolador. Además, esta etapa tiene la finalidad de adecuar las señales de los sensores a los niveles requeridos por el convertidor A/D. Esta etapa de acondicionamiento se diseña utilizando dos amplificadores operacionales LF356, los cuales cuentan con terminales para realizar la corrección del voltaje de desviación (offset).

1.5 Contenido de la Tesis

En el primer capítulo, se presentan los antecedentes de este proyecto, los diferentes trabajos que se han realizado sobre estaciones meteorológicas y se explica su funcionamiento básico. Se presenta el objetivo de este proyecto así como la descripción del problema a solucionar y en la justificación se mencionan las razones por las cuales se desarrolla este proyecto. En la metodología se marcan las etapas que se desarrollaron así como su finalidad de cada una de estas.

En el segundo capítulo, se describen las partes que constituyen el sistema de medición. Es decir, se describen los dispositivos utilizados en este proyecto así como la finalidad de cada uno de ellos.

En el tercer capítulo, se describen las partes principales del microcontrolador que son utilizadas, así como los modos de operación de ahorro de energía con los que cuenta. Además se propone un sistema mínimo para el desarrollo de este prototipo.

En el capítulo cuatro se presenta el desarrollo del *software* para el microcontrolador. Se mencionan los dos programas que fueron desarrollados para la adquisición y visualización de datos. Además, se da la explicación paso a paso de la utilización del *software* usado (*Workbench*) para la implementación de los códigos, la programación del microcontrolador así como su monitoreo de variables.

En el capítulo cinco, se presentan las pruebas y resultados obtenidos del proyecto realizado. Estas pruebas fueron realizadas tanto al *software* como al *hardware* del sistema.

Por último, en el capítulo seis se presentan las conclusiones de este trabajo de tesis, así como los trabajos futuros que servirán para complementar el desarrollo de este proyecto.

Capítulo 2

Configuración del Sistema

En este capítulo se presenta la estructura completa del sistema de medición de variables meteorológicas, haciendo énfasis en la arquitectura de cada uno de los elementos que lo integran. Además se menciona brevemente el tipo de tecnología seleccionada para este proyecto.

2.1 Estructura del Sistema de Medición

En la Figura 2.1 se muestra el diagrama de bloques de la estructura general del sistema de medición implementado, en el cual se puede observar que intervienen diferentes herramientas y dispositivos. Las partes principales de este sistema de medición son una computadora personal, un microcontrolador, sensores de variables ambientales, una etapa de acondicionamiento de señales, un botón de control y una pantalla LCD.

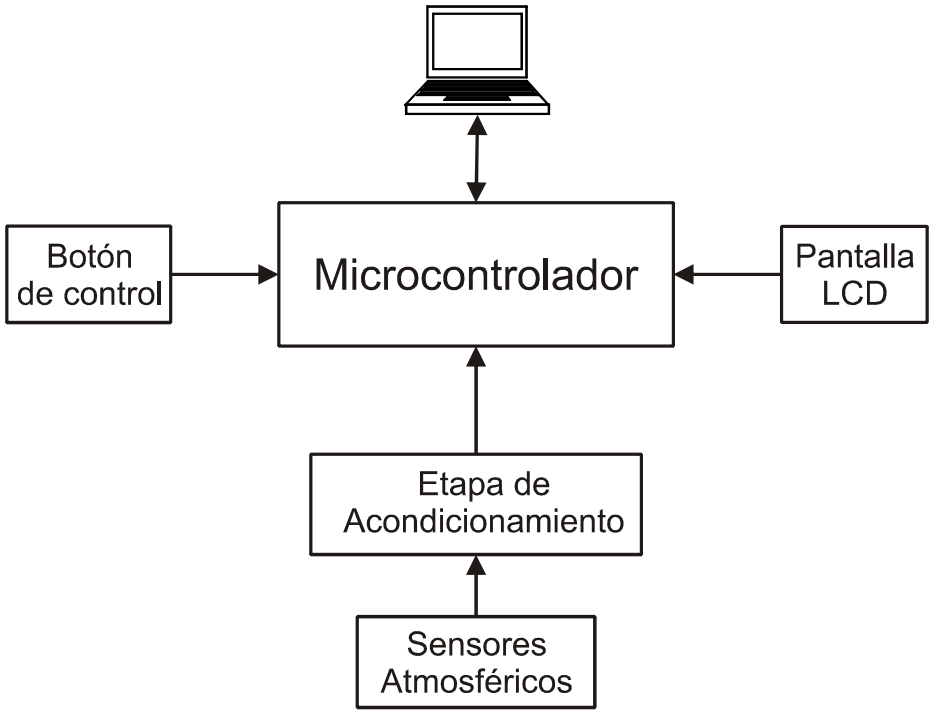


Figura 2.1 Diagrama de Bloques de la Estructura del Sistema de Medición.

La columna vertebral de este sistema de medición está basada en el uso de un microcontrolador de la familia MSP430 de *Texas Instruments*, el cual tiene la función de concentrar y procesar toda la información del sistema. Este microcontrolador se encarga de realizar diferentes actividades, dentro de las cuales resaltan las siguientes:

- Realizar la conversión de analógico a digital de las señales eléctricas generadas por los sensores.
- Monitorear el estado de un botón, el cual se utiliza para tener el control de las variables a medir. Se utiliza el botón para realizar un recorrido a través de las variables a medir en el momento que el usuario lo desee.
- Mostrar en una pantalla de cristal líquido las mediciones realizadas.
- Almacenar los datos en la memoria *Flash* del microcontrolador.
- Crear un reloj que permita llevar el conteo del tiempo y asociar este tiempo con el instante en el cual se realizan las mediciones.

El estado del microcontrolador puede ser consultado en cualquier momento durante su operación ya que el usuario puede estar monitoreando el estado del programa que se ejecuta en el microcontrolador y con esto analizar el funcionamiento de las variables de medición. Además, se pueden visualizar las localidades de memoria utilizadas para el almacenamiento de datos por medio de un programa en la computadora personal. Este microcontrolador es programado a través del puerto USB, por medio de una tarjeta básica de desarrollo que cuenta con un circuito, MSP-FET430U48, un dispositivo que es la interface de conexión de la tarjeta con la computadora por medio de un cable de entrada para el puerto USB de la computadora y con un cable para la interfaz JTAG, con el cual, el microcontrolador es programado [Manual_Texas_6 2005].

En este proyecto se utilizaron dos sensores, uno para medir la presión atmosférica y otro para medir la temperatura ambiental. En el caso del sensor de presión, este es energizado por medio de una fuente de 3V de CD, la cual es proporcionada por el microcontrolador. Para el acoplamiento de este sensor no fue necesario realizar una etapa de acondicionamiento de la señal, debido a que los voltajes de salida del sensor se encuentran dentro de los límites requeridos. Mediante la utilización de la ganancia programable que proporciona el convertidor de analógico a digital se obtiene la medición

de la presión atmosférica en milibares, los cuales son convertidos a mmHg para su comparación con la medición obtenida con una estación meteorológica comercial que proporciona la medición de presión en estas unidades. La altitud en metros se determina un vez que se cuenta con la presión. Sin embargo, para la conexión del sensor de temperatura sí fue necesario realizar una etapa de acondicionamiento de la señal, utilizando para ello el diseño de un amplificador de instrumentación. En este caso, el amplificador de instrumentación se diseña para cumplir la función de atenuar el voltaje de salida del sensor debido a que el rango de voltaje de entrada al convertidor no debe exceder el rango de ±0.6345V. Los amplificadores operacionales utilizados en la implementación de este amplificador de instrumentación son los LF356, los cuales cuentan con terminales para la corrección del voltaje de desviación. Por otra parte, tanto el sensor de temperatura como los amplificadores operacionales son energizados por medio de un banco de baterías recargables, las cuales tienen un arreglo para proporcionar voltajes de ± 8.4v.

Este proyecto cuenta con dos alternativas para la visualización de las mediciones. Una de ellas es por medio de una pantalla LCD, el cual muestra las diferentes mediciones obtenidas cada vez que el botón es presionado. La otra opción es monitoreando las variables por medio de la computadora con la ventada *watch* que proporciona el *software Workbench* que acompaña al microcontrolador. Además, es posible realizar las mediciones de forma automática y almacenar los valores en la memoria *Flash* del microcontrolador en intervalos de tiempo determinados.

2.2 Uso del Microcontrolador como Núcleo Principal de Medición

El microcontrolador es el dispositivo que se encarga de controlar todas las actividades relacionadas con el sistema de medición. Este microcontrolador pertenece a la familia MSP430 de *Texas Instruments,* la cual es una familia de microcontroladores de bajo consumo de potencia [Manual_Texas_8 2006]. Las aplicaciones típicas donde se utilizan este tipo de dispositivos son en sistemas de sensores analógicos y digitales, control digital de motores, control remoto, termostatos, temporizadores digitales y altímetros portátiles.

Una de las principales características para la elección de este microcontrolador es que cuenta con cinco modos de operación para bajo consumo de potencia, en donde estos modos de operación son seleccionados mediante *software*. Cada uno de estos modos tienen

diferentes características, es decir, dependiendo del modo de operación en el que se quiera trabajar, se deshabilitan diferentes módulos periféricos mientras el dispositivo se encuentre en operación, permitiendo que se mantengan encendidos solamente los módulos necesarios. Esta característica garantiza una larga duración para las baterías utilizadas para aquellos casos de medidores portátiles. Este microcontrolador presenta un buen funcionamiento dentro de un amplio rango de temperaturas, desde -40°C a 85°C. Lo anterior favorece la medición de factores atmosféricos en diferentes lugares geográficos a diferentes temperaturas.

En este proyecto el microcontrolador utilizado cuenta con un multiplexor de cinco entradas analógicas para el convertidor analógico digital (A/D) de 16 *bits* de resolución, relojes, un convertidor de digital a analógico de 12 *bits* de resolución (D/A), también cuenta con un controlador de pantalla LCD, 256 MB de RAM y 32KB de memoria *Flash*.

2.3 Pantalla LCD de Interface con el Usuario

Un componente importante del sistema de medición es la pantalla LCD, la cual es utilizada para mostrar las variables medidas. La LCD utilizada, es la SBLCDA4 de la marca *SoftBaugh* [*SoftBaugh* 2003], la cual es compatible con el controlador de LCD con el que cuentan los microcontroladores de la familia MSP430. Esta pantalla opera en un rango de voltaje de 2.7V a 3.6V, por lo que se conecta directamente al controlador de LCD de los microcontroladores MSP430. Esto significa que no requiere de circuitería adicional para la conexión del LCD con el microcontrolador. Además, es necesario utilizar un capacitor para habilitar la operación del módulo LCD_A el cual funge como controlador de la LCD en el microcontrolador. La pantalla cuenta con cuatro multiplexores de operación, tiene siete dígitos de siete segmentos los cuales son controlados mediante *software*, posee simbología como el signo menos, el símbolo de error, antena, batería, memoria, flechas en los cuatro puntos cardinales y cinco símbolos de funciones. Entre otras características de esta pantalla destacan el bajo consumo de potencia, lo cual es un requisito básico para este proyecto. Además se puede ajustar su contraste según las necesidades del usuario mediante *software*. Por último, esta LCD puede operar en un rango amplio de temperaturas ambiente de -20°C a 50°C.

2.4 Sensores de Presión y Temperatura Ambiente

El sensor de presión atmosférica utilizado es el MPXM2102 [*Freescale Semiconductor* 2005], el cual cuenta con una compensación de la temperatura para reducir el efecto de ésta en las mediciones realizadas con tal dispositivo. Este sensor de silicio proporciona mediciones con una gran exactitud, debido a que el voltaje de salida proporcionado por el sensor es directamente proporcional a la presión aplicada. Las características principales con las que cuenta este sensor de presión son:

- Compensación de temperatura para el rango de 0°C a 85°C.
- Voltaje de alimentación relativamente pequeños desde 2V hasta 16V de CD.
- Salida de voltaje diferencial.
- Voltaje máximo de salida de 40mV con un voltaje de 10V de entrada generando 100kPa.
- Buen ajuste a diferentes voltajes de entrada para la obtención de mediciones proporcionales.
- Entrega una señal diferencial a la salida.

De acuerdo a estas características este sensor puede ser utilizado en diferentes aplicaciones tales como: controladores de motores, robótica, indicadores de nivel, diagnósticos médicos, barómetros (medidores de presión) y altímetros (medidores de altitud). En este proyecto el sensor es energizado con una fuente de 3V de CD, obteniendo un voltaje de salida máximo de 12mV. La elección de este sensor de presión se justifica en base a la investigación bibliográfica realizada para este proyecto, en donde se encontraron aplicaciones del microcontrolador MSP430F4270 utilizando este sensor [Manual_Texas_8 2005]. Por lo que este transductor presenta características eléctricas (voltaje y corriente) y físicas adecuadas para la interacción con el microcontrolador para este proyecto de tesis además de es económico.

Para el caso de la medición de la temperatura, el sensor utilizado es el LM35 de la marca *National Semicondctor* [Manual_National_A 2000]. Éste sensor proporciona un voltaje de salida directamente proporcional a la temperatura en grados centígrados. Existen diferentes sensores LM35 que se diferencian entre sí por los rangos de medición que manejan. Por ejemplo, todos los sensores LM35 pueden medir dentro de un rango de 0°C a

150°C, utilizando sólamente una fuente de alimentación de CD positiva. Pero en el caso de este proyecto, no se puede limitar a la condición anterior, debido que este sistema va a ser instalado en lugares con diferentes condiciones climáticas. En este caso se utilizó sensor LM35 de encapsulado TO-46, el cual tiene un rango de medición de -55°C a 150°C. Sin embargo, para poder utilizar todo el rango es necesario proveer una fuente negativa. Algunas de las características principales que distinguen a este sensor son:

- Calibrado directamente en grados Centígrados °C.
- Salida lineal de +10mV / °C.
- Bajo consumo de potencia.
- Menos de 60µA de corriente de drenaje.
- Baja impedancia de salida de .1Ω por 1 Ampere de carga.

Para le selección de este sensor de temperatura se contaba con dos opciones: el LM35 de *National Semiconductor* y el TMP36 de *Analog Devices*, en donde estos dos sensores tienen características muy similares y el costo de ambos es muy similar ya que son económicos, sin embargo, el TMP36 no requiere de una fuente negativa para generación de temperaturas bajo cero por lo que lo hace mejor opción para este proyecto. Pero este transductor tiene la desventaja de no ser muy comercial en el país a diferencia con el LM35 que sí lo es. Esta es la principal razón de por qué se selecciono este transductor para este proyecto.

2.5 Baterías de Alimentación

Las baterías de alimentación utilizadas en este proyecto son baterías de la marca GP, que proporcionan 8.4V y 170mA por hora según los requerimientos del sistema. Estas baterías proporcionan suficiente potencia para energizar los diferentes componentes debido a que estos consumen poca potencia. Este proyecto requiere de dos baterías de este tipo para formar el banco de baterías que alimente los sensores ambientales, tales como el sensor de temperatura y la etapa de acondicionamiento del mismo. Por otra parte, debido a que cuando las baterías son recargadas proporcionan un voltaje mayor a los 8.4V, es necesario regular el voltaje en algunas etapas específicas del sistema que así lo requieren. La elección de este tipo de baterías fue debido a que tienen buenas características eléctricas para el proyecto y que son económicas en comparación a otras baterías del mismo tipo.

Capítulo 3

Diseño del Hardware del Sistema de Medición

En este capítulo se presenta la descripción del microcontrolador MSP430F4270 como elemento principal del diseño del sistema de medición. Se describen los módulos principales del microcontrolador y se explica el procedimiento que se debe de realizar para hacer que estos operen de acuerdo a las características requeridas por este proyecto.

3.1 Descripción del Microcontrolador MSP430F4270

La familia MSP430 de *Texas Instruments* de microcontroladores de bajo consumo de potencia ofrece un conjunto de tarjetas de desarrollo para diferentes aplicaciones. La arquitectura combinada con cinco modos de operación de bajo consumo de potencia optimiza el funcionamiento de los microcontroladores para alcanzar una mayor vida útil de las baterías de alimentación en aplicaciones de mediciones portátiles.

El microcontrolador MSP430F4270 tiene una configuración atractiva para ser utilizado en este proyecto de tesis debido a que cuenta con un convertidor A/D (analógico a digital) de 16-*bits* de alto rendimiento, un convertidor D/A (digital a analógico) de 12-*bits*, 32 terminales que pueden ser utilizados como entrada o salida (I/O), 32kB de memoria *Flash* para código y 256B de memoria *Flash* para almacenamiento de datos. Las aplicaciones típicas en donde se utiliza este dispositivo son en sistemas de sensores analógicos y digitales, en sistemas de control digital de motores, controles remotos, termostatos, temporizadores digitales y altímetros [Manual_Texas_6 2005].

Una característica muy importante de este microcontrolador es que tiene un buen funcionamiento en un rango muy amplio de temperaturas de -40°C a 85°C. Este microcontrolador es de bajo consumo de potencia debido a que es energizado con un voltaje de 2.2V de CD, demandando una corriente en un rango de 2.5μA a 3.5 μA a una temperatura ambiente de 25°C. Estas características eléctricas de demanda de potencia se obtienen con los modos de operación de bajo consumo de potencia.

En la Figura 3.1 se muestra el diagrama de bloques funcional del microcontrolador MSP430F4270, en donde se ilustra la estructura de los módulos que conforman al microcontrolador. Los módulos utilizados en este proyecto son: la CPU, el reloj BASIC TIMER 1, el convertidor SD16_A, el módulo de emulación JTAG, el WDT, el oscilador de cristal FLL, la memoria *Flash*, la memoria RAM, el controlador de la pantalla LCD_A y los módulos para los diferentes puertos de entrada-salida con los que cuenta este dispositivo.

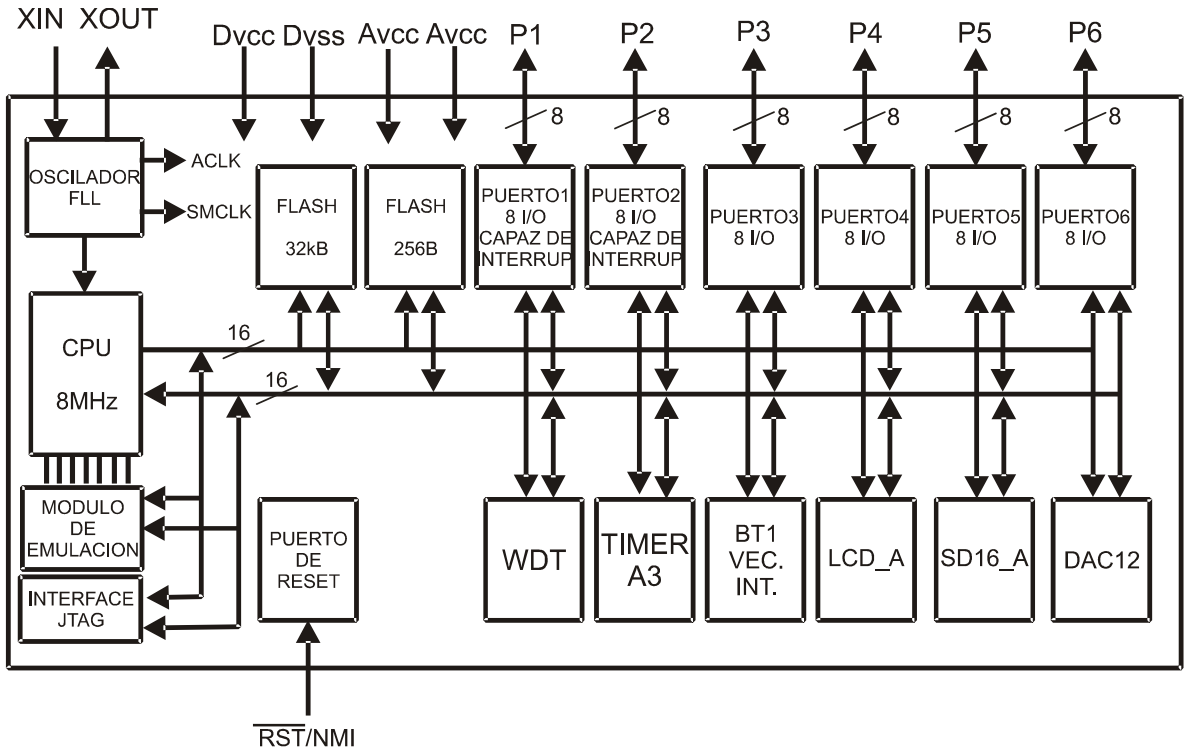


Figura 3.1 Diagrama de bloques funcional del MSP430F4270.

3.1.1 Unidad Central de Procesamiento (CPU)

La CPU de los microcontroladores MSP430 incorpora características específicamente diseñadas para técnicas modernas de programación como la bifurcación calculada, el procesamiento de tablas así como el uso de lenguajes de programación como el C, en cual fueron desarrollados los códigos para este proyecto. La CPU de este

microcontrolador puede direccionar un gran rango de *bytes* sin compaginar los datos. Esto se debe a que la CPU está conectada a su memoria por medio de dos buses de 16-*bits*, uno para direccionamiento (MAB) y el otro para información (MDB).

Todas las instrucciones del CPU son realizadas mediante operaciones de registro en unión con siete modos de direccionamiento por medio del operando origen, además de cuatro modos de direccionamiento por medio del operando destino. El CPU está integrado por una fuente de instrucción de tres fases, instrucción de decodificación, cuatro registros de uso específico, una unidad de aritmética lógica de 16-*bits* (ALU) y doce registros de trabajo, también llamados registros de propósito general. Los 16 registros con los que cuenta la CPU proporcionan un tiempo de ejecución reducido de operación del microcontrolador. Las características principales de la CPU del microcontrolador MSP43F4270 son:

- Arquitectura RISC de 16-*bits*.
- Acceso completo al registro, incluyendo al contador del programa, registros de estado así como a la pila del puntero.
- Operaciones de registro en ciclo simple.
- El gran número de registros reduce la salida de datos de la memoria.
- El bus de direcciones de 20-*bits* permite el acceso directo y la bifurcación en todo el rango de la memoria sin la necesidad de compaginar.
- El bus de datos de 16-*bits* permite una manipulación directa de argumentos de palabra amplia.
- Transferencias directas de memoria a memoria sin que intervenga un registro de retención.
- Proporciona un generador de constantes para reducir el tamaño del código.

En la Figura 3.2 se muestra la estructura de la CPU, en la cual se observa cómo está conectado a los buses de memoria y a la unidad de aritmética lógica (ALU), así como el orden de sus registros.

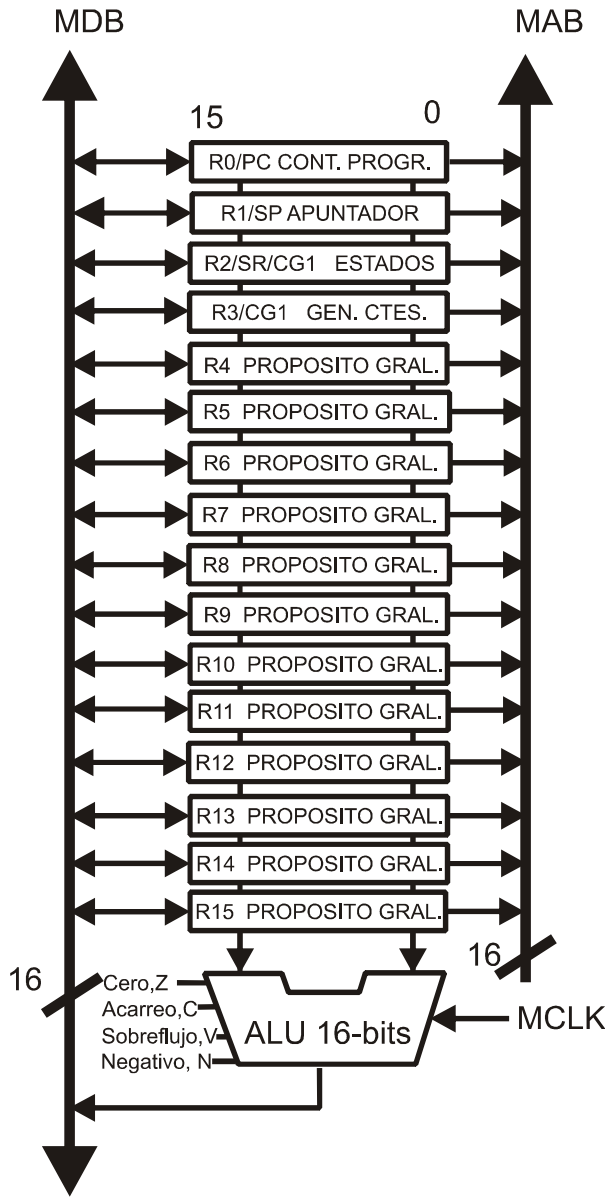


Figura 3.2 Estructura de la CPU.

Unidad de Aritmética Lógica (ALU)

El procesador de los MSP40 incluye una unidad de aritmética lógica (ALU), la cual realiza sumas, restas, comparaciones y operaciones lógicas (AND, OR y XOR). Las operaciones de la ALU pueden afectar las banderas de sobreflujo, de cero, del signo números negativos y el acarreo de banderas [Nagy2003].

Registros

La CPU cuenta con dieciséis registros los cuales tienen diferentes propósitos, cuatro de estos son registros específicos, del R0 al R3, los cuales son utilizados como contadores, apuntadores, registros de estado y como generador de contantes respectivamente. Los otros doce son los llamados registros de trabajo de 16-bits, del R4 al R15, como se observa en la Figura 3.2.

Contador del Programa (PC)

El contador del programa (PC) de 16-*bits* está localizado en R0, el cual es manipulado directamente por código y apunta a la siguiente instrucción que va a ser ejecutada. Cada instrucción utiliza un número constante de *bytes* (dos, cuatro o seis) por lo que en consecuencia el PC es incrementado en ese número de bytes.

El Apuntador (SP)

El apuntador (SP) está implementado en R1, es inicializado dentro de la RAM por parte del usuario y es alineado a cualquiera de las direcciones. El SP es utilizado por la CPU para almacenar las direcciones de regreso de las llamadas a subrutinas e interrupciones. El apuntador utiliza un esquema de pre-decremento y pos-decremento de las localidades de memoria utilizadas. Además, el SP puede ser utilizado o manipulado mediante *software* con todas las instrucciones y modos de direccionamiento.

El Registro de Estado (SR)

El registro de estado (SR) está implementado en R2 y es utilizado como registro fuente o registro destino. Éste puede ser utilizado en el modo de registro solamente direccionándolo con instrucciones de palabra. Las combinaciones restantes de los modos de direccionamiento, son utilizadas para cambiar el modo de este registro, por el modo de generador de constantes. Además, el SR consta de varias banderas de sistema. Las banderas

son accesadas directamente a través de código, excepto tres de ellas, las cuales son cambiadas directamente por el propio procesador.

Generadores de Constantes CG1 y CG2

Una propiedad de los registros R2 y R3 es que se pueden hacer operar como generadores de constantes CG1 y CG2, respectivamente, sin el requerimiento adicional de una palabra de 16-*bits* o un código de programa. Éstos generan seis constantes que son comúnmente las más utilizadas. Las ventajas del generador de constantes son las siguientes:

- No requiere de instrucciones especiales.
- No requiere código para las seis constantes.
- No se requiere accesar a la memoria para recuperar las constantes.

Registros de Propósito General

Los doce registros de propósito general o de trabajo están implementados de R4 a R15, los cuales son usados como operaciones de modo de registro. Todos estos registros pueden ser utilizados como registros de información, apuntadores de direcciones de memoria o bien como valores iníciales. Estos pueden ser accesados y modificados por medio de instrucciones *byte* ó palabra. El uso de estos registros, hace que las operaciones sean más eficientes, debido a que estos no tienen acceso a la memoria, es decir, son independientes.

3.1.2 Organización de la Memoria

La memoria del microcontrolador MSP430F4270, está dividida en siete segmentos específicos, los cuales corresponden a los siguientes módulos: registros de función especial (SRF), registros periféricos, memoria RAM, memoria de inicialización tipo *Flash*, memoria de información tipo *Flash*, memoria de código y vectores de interrupción. La estructura o mapa de la memoria del microcontrolador MSP430F4270 se puede presenta en la Tabla 3.1, en donde se indica su tamaño y su tipo.

Tabla 3.1 Estructura de la Memoria del microcontrolador MSP430F4270

Memoria	Tamaño	32KB
Principal: vector de interrupción	*Flash*	0FFFFh-0FFE0h
Principal: memoria de código	*Flash*	0FFFFh-08000h
Memoria de información	Tamaño	256 Byte
	Flash	010FFh − 01000h
Memoria de arranque	Tamaño	1KB
	ROM	0FFFh − 0C00h
Memoria RAM	Tamaño	256 Byte
		02FFh − 0200h
Periféricos	16-*bits*	01FFh − 0100h
	8-*bits*	0FFh − 010h
	8-*bits*SRF	0Fh − 00h

Una característica importante por la cual se escogió éste microcontrolador para desarrollar el presente proyecto es que cuenta con memoria *Flash*, tanto para código como para almacenamiento de datos. Esto es una gran ventaja en comparación de otros microcontroladores debido a que ya no es necesario utilizar *hardware* adicional para guardar permanentemente el código elaborado ó para el almacenamiento de información.

Como se puede observar en la Tabla 3.1, el segundo segmento de memoria corresponde al segmento de memoria de código (32kB), en donde se almacenan los códigos desarrollados para programar el microcontrolador. La versión del programa controlado por un botón tiene un tamaño de 26.2kB, mientras que la versión automática tiene un tamaño de 17.7kB.

Registros de Función Especial

Nominalmente son 16 registros con localidades de memoria que van de 0000h a 000Fh. Las localidades 0000h y 0001h contienen interrupciones habilitadas, las localidades 0002h y 0003h contienen banderas de interrupción, la localidades 0004h y 0005h un

módulo de banderas habilitadas. Además, estos registros pueden ser activados solamente mediante *software*.

Registros Periféricos

Todos los registros periféricos están ubicados dentro de la memoria, inmediatamente después de los registros de función especial. Existen dos tipos de registros periféricos:

- Los de *byte* direccionable, los cuales se encuentran ubicados en las localidades de memoria a partir de la localidad 010h a la 0FFh.

- Los de palabra direccionable, los cuales se encuentran ubicados en las localidades de memoria a partir de la localidad 0100h a la 01FFh.

Memoria RAM

La memoria RAM tiene una dimensión de 32KB, la cual va de la localidad de memoria 0200h hasta la 02FFh. La memoria RAM puede ser utilizada tanto para código como para información.

Memoria de Arranque

La memoria de inicialización está implementada en los dispositivos *Flash*, como es el caso del microcontrolador MSP430F4270, abarcando el rango 0C00h-0FFFh de localidades de memoria. Éste es el único espacio ROM en los dispositivos *Flash*.

Memoria de Información

Los dispositivos *Flash* de la Familia de microcontroladores de MSP430 tienen una característica adicional de gran importancia, la cual es una memoria *Flash* de información. Esta memoria *Flash* es programada por medio del puerto JTAG, por el cargador de secuencia de arranque o bien, dentro del sistema por medio de la CPU. En este último caso, la CPU puede realizar escrituras de un solo *byte* o de una sola palabra a la memoria *Flash*. La memoria *Flash* se encuentra dividida en dos segmentos de 128 *bytes* cada uno.

El primer segmento abarca el rango de localidades de memoria que va desde la localidad 01000h hasta la localidad 0107Fh, a la cual se le conoce como segmento B de la memoria *Flash*. El segundo segmento abarca el rango de localidades de memoria a partir de la localidad 01080h hasta la localidad 0FFFFh, este segmento es llamado el segmento A de la memoria Flash. Otra característica de este tipo de memoria, es que cada uno de sus

segmentos, pude ser escrito o borrado de forma independiente. En la Figura 3.3 se muestra la segmentación de la memoria *Flash* así como las dimensiones de cada uno de los segmentos.

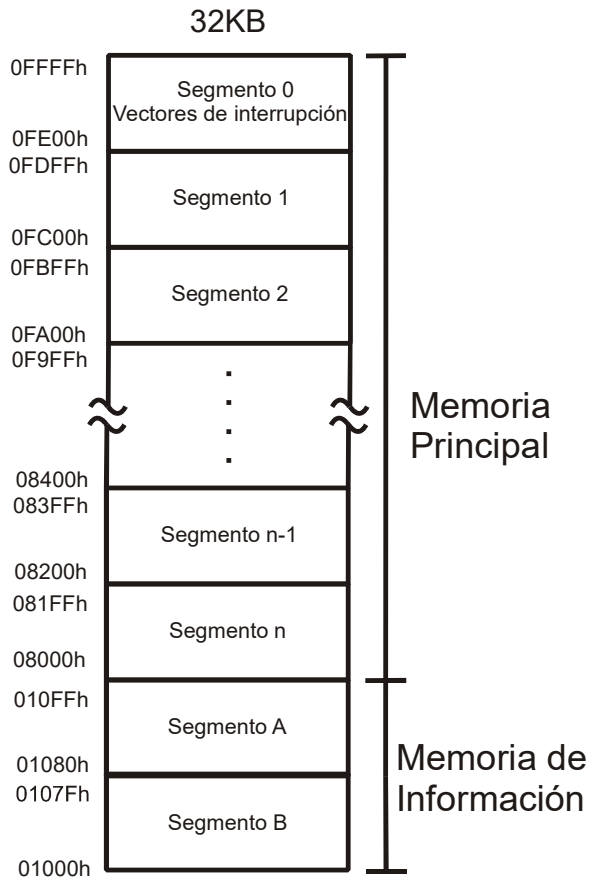


Figura 3.3 Segmentación de la Memoria Flash.

Memoria de Código

En la memoria de código se almacena el programa realizado para este proyecto. Este segmento de memoria abarca un rango de localidades de memoria que va desde 08000h hasta 0FFFFh. En este segmento de memoria también se almacenan las tablas y constantes utilizadas para el código. El código de este proyecto fue desarrollado en el lenguaje de programación C.

Vectores de Interrupción

Los vectores de interrupción se encuentran localizados en el último espacio de la memoria, abarcando las localidades de memoria a partir de la localidad 0FFE0h hasta la localidad 0FFFFh.

3.1.3 Módulo Convertidor A/D

El microcontrolador MSP430F4270 tiene integrado el módulo SD16_A el cual es un convertidor analógico a digital de 16-*bits* de resolución que cuenta con una alta impedancia en el buffer de entrada y un voltaje de referencia interno. Éste módulo cuenta con un multiplexor de ocho pares de entrada además de que contiene integrado un sensor de temperatura interno. Las características principales del SD16_A se muestran a continuación:

- Arquitectura de 16-*bits* sigma-delta.
- Un multiplexor con ocho pares de entradas para señales analógicas.
- Voltaje de referencia interno con un valor de 1.2V, el cual es activado mediante software.
- Se puede utilizar un voltaje de referencia externo o bien el interno, seleccionado cualquiera de los dos mediante *software*.
- Cuenta con un sensor de temperatura integrado.
- Contiene un modulador para frecuencias de entrada que este por arriba de 1.1MHz.
- Tiene un buffer de entrada de alta impedancia.
- Se pude seleccionar algún modo de conversión de bajo consumo de potencia.

En la Figura 3.4 se muestra el diagrama de bloques del módulo del SD16_A, en la cual se puede observar cada uno de los elementos que conforman al convertidor A/D con cada una de las características mencionadas con anterioridad.

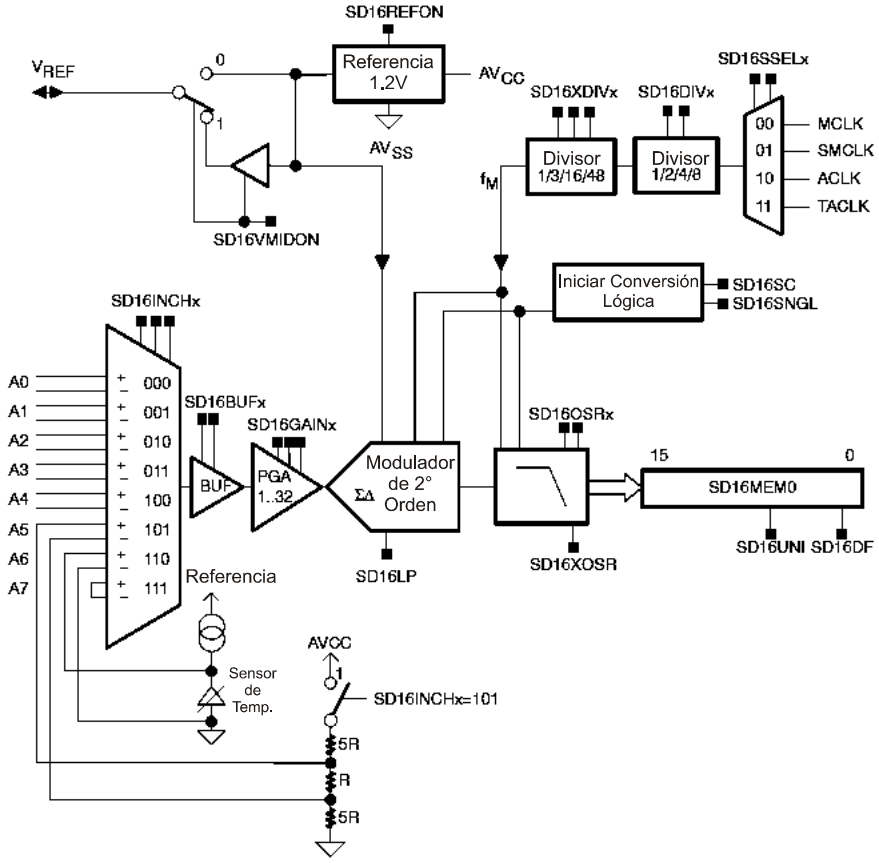


Figura 3.4 Diagrama de bloques del convertidor A/D SD16_A.

Para la activación del módulo correspondiente al convertidor A/D se utiliza un voltaje de referencia externo el cual se obtiene de un circuito divisor de tensión. El cálculo de sus elementos se realiza suponiendo un valor de una resistencia y utilizando el valor de la fuente de alimentación de Vcc = 3V. En la siguiente ecuación se presenta el cálculo de referencia externa:

$$V_{REF} = V_{CC} * \frac{Ra}{Ra+Rb} = 3V * \frac{11k}{11k+15k} = 1.269V \qquad (3.1)$$

En la Figura 3.5 se muestra la implementación del divisor de tensión que proporciona el voltaje de referencia externo utilizado para activar al convertidor A/D SD16_A. Los capacitores utilizados en este divisor de tensión son utilizados para filtrar los

voltajes de CD. El voltaje Vcc es tomado del microcontrolador a partir de una conexión entre las terminales 19 y 20, las cuales son configuradas como salidas mediante *software*.

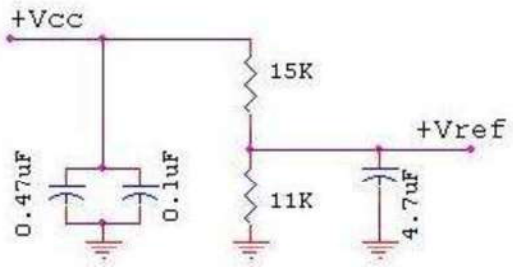


Figura 3.5 Generación del voltaje de referencia.

3.1.3.1 Relación entre el Rango de Voltaje para la Entrada Analógica y el Bloque del Amplificador de Ganancia Programable (GPA)

El rango para el voltaje de entrada a plena escala para cada uno de los pares de entradas analógicas depende del valor de la ganancia seleccionada en el bloque del amplificador de ganancia programable PGA.

El máximo rango de voltaje de entrada para cada una de las señales analógicas es de $\pm V_{FSR}$, en donde V_{FSR} está definido por:

$$V_{FSR} = \frac{V_{REF}/2}{GAIN_{PGA}} \tag{3.2}$$

V_{REF} es el voltaje de referencia y $GAIN_{PGA}$ es la ganancia seleccionada. Para obtener el máximo rango para el voltaje de entrada es necesario utilizar una ganancia unitaria. Cabe resaltar que el voltaje de referencia puede ser el interno que proporciona el SD16_A con un valor de 1.2V o bien un voltaje de referencia externo que sea aplicado a la terminal V_{REF}.

3.1.3.2 Generador del Voltaje de Referencia

El módulo SD16_A proporciona un voltaje de referencia de 1.2V el cual se obtiene habilitando el bit SD16REFON. Como recomendación para reducir el ruido en este voltaje de referencia interno, es necesario conectar un capacitor de 100nF entre las terminales del microcontrolador V_{REF} y AV_{SS}. Además, este voltaje de referencia interno puede ser

utilizado para energizar elementos externos cuando el bit SD16VMIDON=1, tomando en cuenta que esta fuente puede proporcionar una corriente de 1mA. Si se requiere utilizar este voltaje de manera externa es necesario conectar un capacitor de 470nF entre las terminales V_{REF} y AV_{SS}. Los *bits* mencionados para la habilitación del voltaje de referencia se encuentran dentro del registro SD16CTL mostrado en la Figura 3.6.

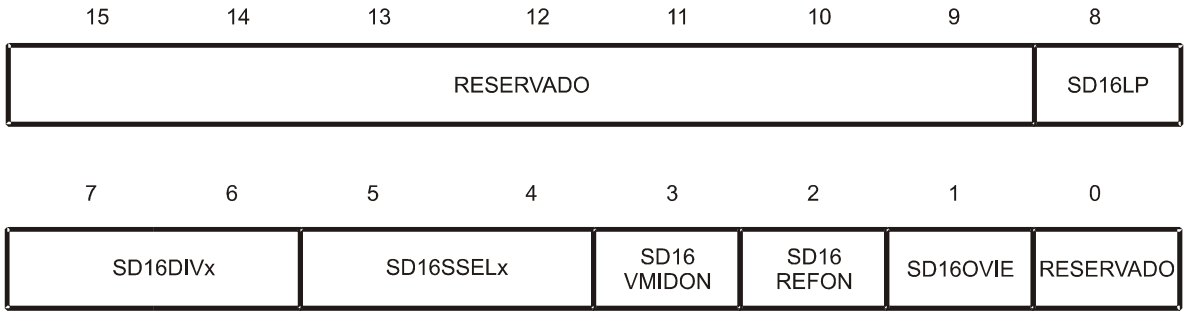


Figura 3.6 Registro SD16CTL.

3.1.3.3 Selección de Canal

El módulo SD16_A puede convertir hasta ocho pares de entradas diferenciales por medio del multiplexor con el que cuenta como se muestra en la Figura 3.4. Las señales de salida del multiplexor pasan por un bloque llamado amplificador de ganancia programable (PGA), en el cual la señal que entra a éste es amplificada en potencias de base dos (1, 2, 4, 8, 16 y 32 veces el valor de la señal de entrada). El SD16_A cuenta con cinco pares de entradas analógicas externas las cuales son A0, A1, A2, A3 y A4 (son los primeros cinco pares de los ocho con los que cuenta el multiplexor), de las cuales son seleccionadas y programadas mediante *software*. Una de las características de este módulo, como ya fue mencionado, es que el SD16_A tiene integrado un sensor de temperatura interno, el cual está disponible en el par de entradas A6. Este sensor de temperatura es utilizado en conjunto con los sensores externos para medir la temperatura del microcontrolador. Al igual que los demás pares de entradas, este par es seleccionado mediante *software*. Sin embargo, el sensor de temperatura interno es energizado con el voltaje de referencia proporcionado por el SD16_A, mientras que los sensores externos son alimentados por voltajes externos. El último par de entradas corresponde al A7 en el cual sus entradas diferenciales se encuentran conectadas entre sí, lo cual sirve para hacer la corrección de voltaje de desplazamiento (*offset*).

3.1.3.4 Activación de las Entradas Analógicas

Cada una de las entradas analógicas son configuradas por medio del bloque SD16INCTL0 (control de entrada del SD16_A), mostrado en la Figura 3.7, en cual se selecciona la entrada a utilizar así como el valor de la ganancia proporcionada por el PGA. Por medio del bloque SD16INCHx (canal de entrada), se realiza la selección de par de entradas a utilizar poniéndole un valor a la x, este valor es dado en decimal dependiendo del número de la entrada a utilizar. Al igual que con el bloque anterior el bloque SD16GAINx (ganancia programable), el valor de la ganancia es seleccionado con un número decimal que representa una de las potencias de dos. Dicho número representa la cantidad en la que se quiere amplificar la señal de entrada.

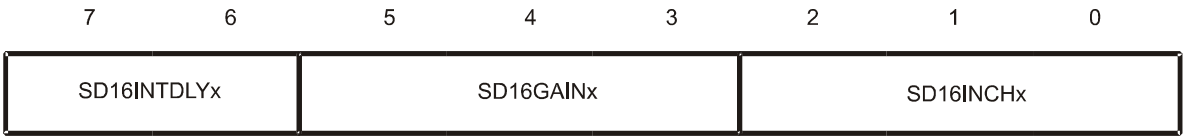


Figura 3.7 Registro SD16INCTL0.

3.1.3.5 Conversión de los Registros de Memoria: SD16MEM0

A cada uno de los canales del SD16_A se le asocia un registro SD16MEM0, en el cual se vacía la información resultante de la conversión. Una vez que son realizadas las conversiones de las señales analógicas conectadas a cada una de las entradas, los resultados de dichas conversiones son enviados al registro SD16MEM0 correspondiente a cada canal. Esto se realiza una vez que la señal pasa por el filtro digital de señales.

Formato de Salida de la Información

El formato de salida de la información puede ser configurado de tres formas: en complemento a dos (BIPOLAR), en compensación binaria (BIPOLAR) o en UNIPOLAR. El formato de la información es seleccionado por medio de los *bits* SD16DF y SD16UNI, los cuales se encuentran en el registro SD16CCTL0 mostrado en la Figura 3.8.

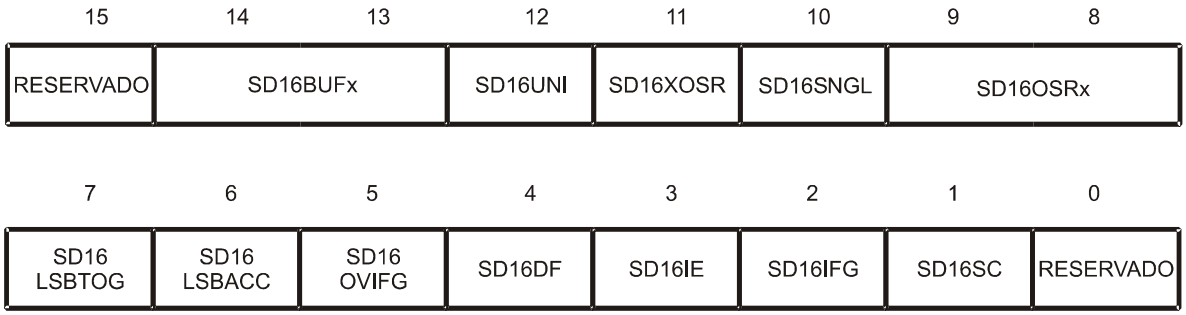


Figura 3.8 Registro SD16CCTL0.

En la Tabla 3.2 se muestra el comportamiento de las tres configuraciones del formato de la información de salida, dependiendo de las combinaciones de los *bits* SD16DF y SD16UNI.

Tabla 3.2 Configuración del formato de la información de salida.

SD16UNI	**SD16DF**	**Formato**	**Entrada analógica**	**SD16MEMx**	**Salida del filtro digital (OSR=256)**
0	0	BIPOLAR:	+FSR	FFFF	FFFFFF
		Compensación binaria	Cero	8000	800000
			-FSR	0000	000000
0	1	BIPOLAR:	+FSR	7FFF	7FFFFF
		Complemento a dos	Cero	0000	000000
			-FSR	8000	800000
1	0	UNIPOLAR	+FSR	FFFF	FFFFFF
			Cero	0000	800000
			-FSR	0000	000000

En la Figura 3.9 se muestra la relación entre el rango máximo del voltaje de entrada $\pm V_{FSR}$ y los resultados obtenidos de la conversión especificando alguno de los tres formatos de salida.

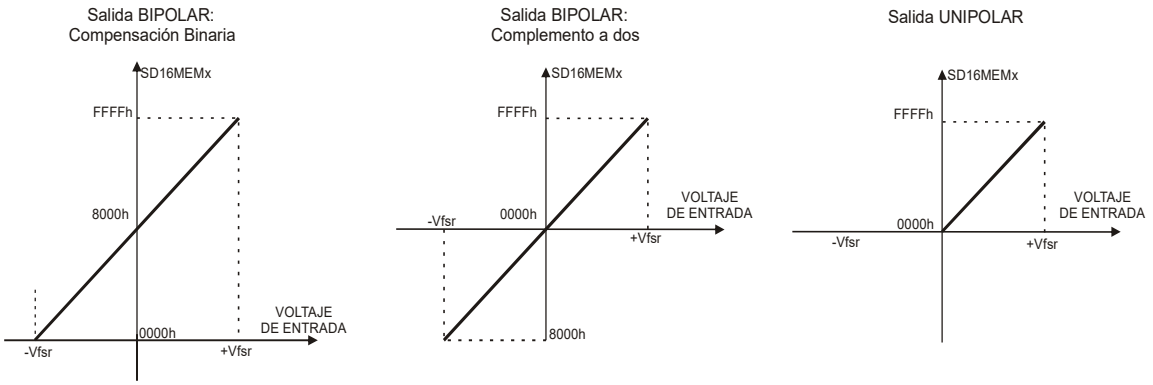


Figura 3.9 Relación entre $\pm V_{FSR}$ y los resultados obtenidos de la conversión.

3.1.3.6 Modos de Conversión

El módulo SD16_A tiene la propiedad de poder ser configurado en dos modos operación los cuales se muestran en la Tabla 3.3. En dicha tabla se puede observar que el *bit* SD16SNGL selecciona uno de estos dos modos de conversión: conversión simple y conversión continua. Este *bit* se encuentra en el registro SD16CCTL0 mostrado en la Figura 3.8.

Tabla 3.3 Resumen de los Modos de Conversión.

SD16SNGL	MODO	OPERACIÓN
1	Conversión Simple	La señal del canal es convertida una vez
0	Conversión Continua	La señal del canal es convertida continuamente

3.1.3.6.1 Conversión Simple

Cuando el bit SD16SNGL = 1 se configura el modo de conversión simple, se activa el *bit* SD16SC (*bit* para la conversión simple) y se inicia con la conversión de la señal entrante al convertidor. El bit SD16SC va a ser restablecido a su valor después de que termine la conversión de la señal analógica a digital.

Inmediatamente después de que la conversión a finalizado y que el *bit* SD16SC ha sido restablecido a su valor original, la conversión del canal se detiene, el canal es desenergizado y el filtro digital correspondiente del canal se apaga. El valor que se

encuentra en SD16MEM0 puede cambiar una vez que el *bit* SD16SC se activa, por lo que es recomendable que la información obtenida de la conversión sea leída o almacenada antes de que el *bit* SD16SC se desactive y genere un dato que no corresponde a la conversión.

3.1.3.6.2 Conversión Continua

Cuando se establece que SD16SNGL = 0 se configura el modo de conversión continua y el *bit* SD16SC es activado con lo que se inician las conversiones en el convertidor, una en seguida de otra, hasta que el *bit* SD16SC es desactivado mediante *software*. Este modo de operación se puede controlar mediante el establecimiento de un determinado tiempo que se desee para las conversiones.

Una vez que se ha cumplido con el tiempo establecido para las conversiones, o bien, una vez que el *bit* SD16SC ha sido desactivado, las conversiones del canal seleccionado son detenidas y el canal es apagado junto con el filtro digital correspondiente a dicho canal. Al igual que en el modo de conversión simple, el valor que se encuentra en SD16MEM0 puede sufrir variaciones una vez que el *bit* SD16SC es desactivado. Por lo que es recomendable que el resultado obtenido de la conversión se guarde antes de perder la información. En la Figura 3.10 se resume el funcionamiento de los dos modos de conversión en donde se puede observar el comportamiento tanto de la conversión simple como de la conversión continua.

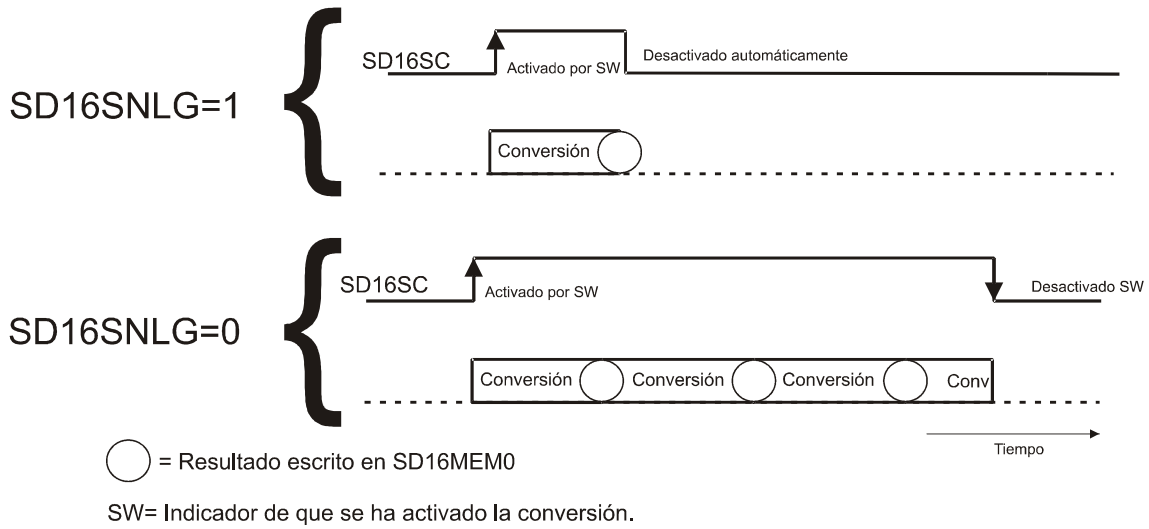


Figura 3.10 Modos de conversión Simple y Continua.

3.1.4 Interrupciones

El microcontrolador MSP430F4270 cuenta con 16 diferentes fuentes de interrupciones, como se muestra en la Figura 3.11, pero en este proyecto solamente se utilizan tres de estas fuentes de interrupciones: WDT (*wath dog timer*), PORT_1 (puerto 1) y la del módulo SD16_A.

El uso de interrupciones dentro de un proyecto basado en la utilización de un microcontrolador, es la mejor opción para tener el control de flujo del programa que está basado en eventos. Pero se debe mencionar que así como el uso de las interrupciones dentro de un programa nos permite tener el control del flujo del programa, también puede hacer que este control se pierda debido a una mala programación de las interrupciones. Esta mala programación se refiere a que puede ocurrir que después de que una interrupción es ejecutada ésta no se deshabilite lo que trae como consecuencia que el flujo del programa siempre se encuentre dentro de una interrupción.

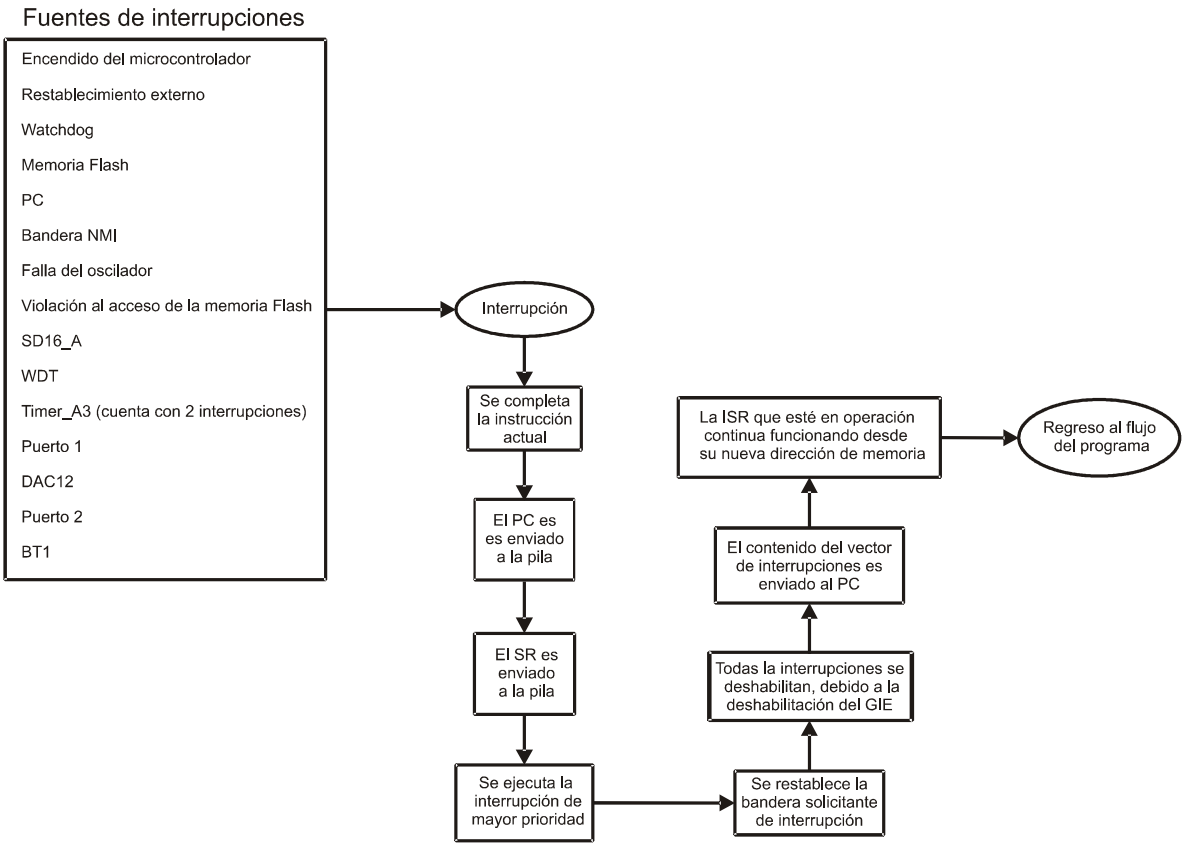


Figura 3.11 Fuentes de interrupciones y sus actividades.

En este tipo de microcontroladores existen tres tipos de interrupciones: interrupciones enmascarables, no enmascarables y las del sistema de restablecimiento. En la Figura 3.11 se muestran las diferentes fuentes de interrupciones con las que cuenta el MSP430F4270, así como las actividades que realizan todas las interrupciones cada vez que son activadas cada una de ellas. Las interrupciones tienen una jerarquía en el momento que se estén activando dos o más de estas fuentes al mismo tiempo. Dicha jerarquía se muestra en orden descendente dentro de la Figura 3.11.

3.1.4.1 Interrupciones Enmascarables (MI)

Las interrupciones enmascarables son ocasionadas por los periféricos que cuentan con la capacidad de generar una interrupción. Dentro de este tipo de periféricos se incluyen todos los mostrados en la lista de fuentes de interrupciones de la Figura 3.11, con la excepción de dos: la del oscilador y la de la memoria *Flash*, debido a que estas corresponden a las no enmascarables (NMI). Cada una de las fuentes de interrupciones enmascarables pueden ser deshabilitadas individualmente por medio de un *bit* de habilitación. Otro modo de deshabilitar las interrupciones es por medio del *bit* de Interrupción General Habilitada (GIE) que se encuentra dentro del registro de estado (SR).

3.1.4.2 Interrupciones No Enmascarables (NMI)

Las interrupciones no enmascarables son habilitadas por medio de diferentes *bits* específicos. Este tipo de interrupciones no son enmascaradas por el *bit* GIE. Cuando ocurre una NMI, todos los *bits* que se encuentran activos, son desactivados automáticamente. Para poder activar nuevamente los *bits* que fueron desactivados mediante una interrupción, es necesario que se de esta instrucción mediante *software*. Este tipo de interrupciones son generadas por tres tipos de fuentes:

- La terminal $\overline{RST}/NMI$, cuando está configurada como NMI.
- Cuando ocurre una falla con el oscilador del sistema se genera una interrupción automáticamente.
- Cuando hay una violación al acceso de la memoria *Flash*.

3.1.4.3 Procedimiento de una Interrupción

Cuando ocurre una interrupción de cualquier tipo de las ya mencionadas, las actividades que realiza el microcontrolador son las siguientes:

1. Cualquier instrucción que se encuentre en ejecución, es completada.
2. El PC, el cual apunta a la siguiente instrucción, es enviado a la pila.
3. El SR es enviado a la pila.
4. Si ocurren diferentes interrupciones al mismo tiempo, se ejecuta la que tenga mayor prioridad y las demás se quedan pendientes.
5. La bandera solicitante de interrupción se restablece automáticamente para banderas con un solo origen o fuente.
6. Debido a que el GIE es deshabilitado, cualquier interrupción adicional se deshabilita.
7. El contenido que se encuentre dentro del vector de interrupciones es cargado al PC: el programa continúa con la rutina de servicio de interrupción en esa dirección.

3.1.5 Modos de Operación de Ahorro de Energía

El MSP4304270 tiene un modo activo de operación y cinco modos de operación con bajo consumo de potencia, los cuales se seleccionan por medio del *software*. Un evento de interrupción puede despertar al dispositivo de cualquiera de los cinco modos mencionados, realizar el requerimiento solicitado y restablecerse en el modo de operación de bajo consumo de potencia que tenía antes de la interrupción en el programa.

Los siguientes seis modos de operación pueden ser configurados por medio de *software*:

1. Modo activo AM:

- Todos los relojes son activados.

2. Modo de bajo consumo de potencia 0 (LPM0):

- El CPU es deshabilitado.
- El ACLK y SMCLK permanecen activos, MCLK está disponible para los módulos.
- El control FLL+Loop permanece activo.

3. Modo de bajo consumo de potencia 1 (LPM1):

- El CPU es deshabilitado.

- El ACLK y SMCLK permanecen activos, MCLK está disponible para los módulos.
- El control FLL+Loop es desactivado.

4. **Modo de bajo consumo de potencia 2 (LPM2):**
   - El CPU es deshabilitado.
   - El MCKL, el control FLL+Loop y el CDOCLK son deshabilitados.
   - El generador de CD del oscilador digitalmente controlado (DCO) permanece habilitado.
   - El ACLK permanece activo.
5. **Modo de bajo consumo de potencia 3 (LPM3):**
   - El CPU es deshabilitado.
   - El MCLK, el control FLL+Loop y el DCOCLK son deshabilitados.
   - El generador de CD del DCO permanece es deshabilitado.
   - El ACLK permanece activo.
6. **Modo de bajo consumo de potencia 4 (LPM4):**
   - El CPU es deshabilitado.
   - El ACLK es deshabilitado.
   - El MCLK, el control FLL+Loop y el DCOCLK son deshabilitados.
   - El oscilador de cristal es detenido.

en donde ACLK es el reloj auxiliar, SMCLK es el reloj maestro del subsistema, FLL+Loop es el bucle de frecuencia cerrada, MCLK es el reloj maestro y DCOCLK es el reloj del oscilador digital controlado.

En este proyecto el modo de operación de bajo consumo de potencia utilizado es el LPM3 debido a que cuenta con las características requeridas en este sistema.

3.2 Aplicación del Microcontrolador MSP430F4270

En la Figura 3.12 se muestran los elementos que componen el sistema de medición desarrollado en este proyecto. Además, de muestran los diferentes periféricos del microcontrolador y su interacción que tienen con los elementos externos que componen la circuitería desarrollada.

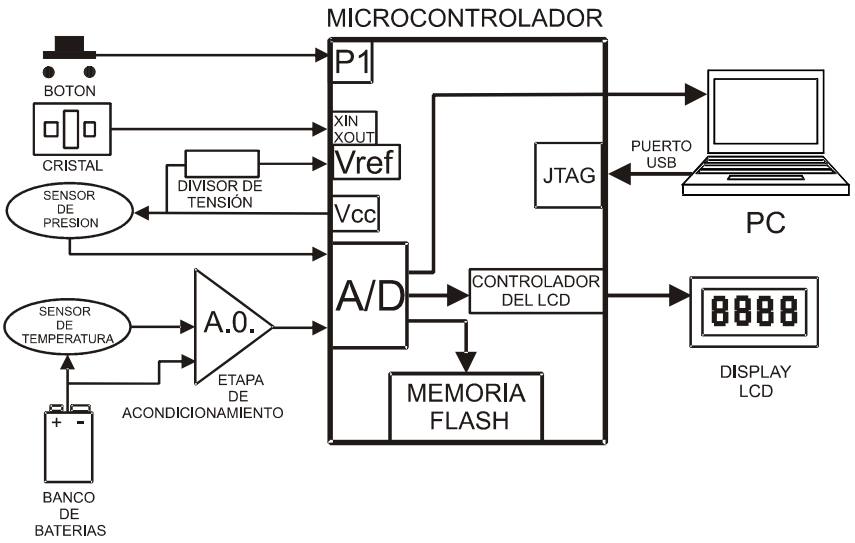


Figura 3.12 Elementos del sistema de medición.

El microcontrolador MSP430F4270 cuenta con diferentes periféricos, de los cuales se utilizaron los mostrados en la Figura 3.12 de la siguiente manera:

- El Puerto 1 (P1) cuenta con una serie de terminales las cuales pueden ser configuradas como entradas o salidas, en donde la terminal 27 etiquetada como P1.1 es configurada como entrada para conectarle un botón el cual proporciona un pulso que activa la medición de las variables ambientales.
- Las terminales XIN y XOUT, son las correspondientes al oscilador de cristal el cual genera la frecuencia necesaria para la operación del microcontrolador.
- La terminal Vref es la entrada proporcionada por el microcontrolador para cuando se requiere utilizar un voltaje de referencia externo para la activación del convertidor A/D. Este voltaje de referencia es obtenido mediante un circuito divisor de tensión, el cual es alimentado por una fuente Vcc proporcionada por el microcontrolador.
- El convertidor A/D es el módulo en donde se conectan los sensores utilizados. Estos sensores pueden ser conectados directamente al microcontrolador en caso de que las señales generadas por los mismos cumplan con las características de entrada al convertidor (como en el caso del sensor de presión conectado al par de entradas A0). De no ser así, dichas señales tienen que ser adecuadas a las características de entrada el convertidor A/D por medio de una etapa de acondicionamiento (como en

el caso del sensor de temperatura conectado al par de entradas A1). Este módulo es activado mediante un voltaje de referencia, el cual puede ser un voltaje externo conectado a la terminal Vref o bien, dentro del módulo del convertidor A/D se encuentra módulo que proporciona un voltaje con valor de 1.2V, al cual se le llama voltaje de referencia interno. Éste módulo se llama Referencia y es habilitado mediante la activación del *bit* SD16REFON el cual se encuentra en el registro SD16CTL. Este voltaje de referencia además de activar el convertidor A/D también activa al sensor de temperatura interno del microcontrolador. Es necesario comentar que cuando se utiliza un voltaje de referencia externo para la activación del convertidor A/D es necesario que los *bits* SD16REFON y SD16VMIDON del registro SD16CTL se encuentren desactivados. Una vez que las señales analógicas de entrada son convertidas a señales digitales, éstas tienen un formato de salida determinado. En el caso de este proyecto la configuración del formato de la salida de la información corresponde a la Bipolar con complemento a dos, en donde el comportamiento de este se muestra en la Figura 3.9. En este trabajo se optó por utilizar esta configuración tanto en las mediciones realizadas con sensores externos como en las mediciones realizadas con el sensor de temperatura interno. Con esta configuración fue necesario acondicionar la señal del sensor de temperatura externo LM35 al rango de valores que abarca dicha configuración en el primer cuadrante.

- El periférico correspondiente a la memoria *Flash* es el registro de memoria en donde se almacenan las mediciones realizadas en un tiempo determinado. Este procedimiento se realiza en la versión del código que realiza las mediciones de forma automática. Una vez que la memoria *Flash* se llena, ésta es borrada y nuevamente se va llenando con las nuevas mediciones realizadas.
- El periférico del controlador del LCD sirve como interface para la conexión de una pantalla LCD con el microcontrolador.
- La conexión de la computadora con el microcontrolador se realiza por medio del periférico JTAG. Por este periférico es por donde se realiza la programación del microcontrolador.
- La computadora es utilizada para la programación del microcontrolador y para el monitoreo de la memoria *Flash* del mismo.

Como se puede observar en la Figura 3.12, los sensores son alimentados de forma independiente, es decir, el sensor de presión es energizado por medio de una fuente proporcionada por el microcontrolador mientras que el sensor de temperatura es energizado por un banco de baterías que al mismo tiempo energiza a la etapa de acondicionamiento diseñada para este sensor.

3.3 Diseño del Sistema de Medición Mínimo basado en el Microcontrolador MSP430F4270

A lo largo del desarrollo de este proyecto de tesis se utilizó una tarjeta de desarrollo de propósito general para el microcontrolador MSP430F4270. Las diferentes partes del sistema de medición desarrolladas se probaron precisamente en esta tarjeta de desarrollo. Es por ello que en esta sección se detalla la construcción del sistema mínimo basado en el microcontrolador MSP430F4270. Este sistema mínimo consiste de diferentes dispositivos como son: un microcontrolador MSP430F4270 de bajo consumo de potencia, una pantalla LCD para la visualización de las mediciones en tiempo real, sensores de presión atmosférica y de temperatura ambiente y una etapa de condicionamiento de señales.

3.3.1 Conexiones Básicas

El circuito mínimo para la operación del microcontrolador MSP430F4270 se compone de una fuente de poder 3V, la cual se conecta a través de una resistencia de 69kΩ, a la terminal número 5 del microcontrolador. Además, requiere un oscilador de cristal con una frecuencia de 32.768kHz, el cual es conectado a las terminales 8 y 9 del microcontrolador. El oscilador requiere de un capacitor de 12pF en cada una de sus terminales para su funcionamiento. Por su parte para operar el convertidor A/D, se requiere un voltaje de referencia externo con un valor aproximado a 1.2V, el cual es aplicado a la terminal 12 del microcontrolador. Este voltaje de referencia se puede obtener por medio de la implementación de un circuito divisor de tensión tomando como voltaje de entrada el voltaje proporcionado por la fuente de poder. Con estos elementos el microcontrolador se enciende y está listo para poder realizar conversiones de señales de entrada analógicas a digitales.

3.3.2 Pantalla LCD SBLCDA4

La pantalla de cristal líquido SBLCDA4 es compatible con los microcontroladores de la familia MSP430, por lo que la conexión de la LCD se realiza directamente sin el requerimiento de circuitería adicional para al acoplamiento. Esta LCD es una pantalla de cristal líquido la cual opera en un rango de voltaje de 2.7V a 3.6V. Una de las características principales por la cual se utiliza esa pantalla es que consume muy poca potencia cuando se encuentra en operación, lo cual es de vital importancia para este proyecto. Esta LCD puede operar en un rango amplio de temperaturas ambiente de -20°C a 50°C. Así mismo, el contraste de la pantalla puede ser ajustado mediante software según sean las necesidades del usuario. Esta pantalla cuenta con cuatro multiplexores de operación, contiene siete dígitos de siete segmentos los cuales son controlados mediante *software* dentro del código que es cargado al microcontrolador. Además, posee diferente simbología como el signo menos, el símbolo de error, antena, batería, memoria, indicadores de los cuatro puntos cardinales y cinco símbolos que representan funciones. El único elemento requerido para el buen funcionamiento de esta pantalla consiste en un capacitor de 4.7µF, el cual habilita la operación del módulo LCD_A.

3.3.3 Botón del Usuario

El sistema mínimo incluye un botón de control del flujo del programa, el cual puede ser utilizado por el usuario del medidor para desplegar en la pantalla LCD el estado de las variables meteorológicas. Estas mediciones son visualizadas por medio de la pantalla LCD en tiempo real. Este botón se encuentra conectado al puerto 1 del microcontrolador y sirve como controlador de las interrupciones realizadas por el dicho puerto debido a que las interrupciones se activan o desactivan cada vez que el botón es presionado por un tiempo determinado.

3.3.4 Sensores

En este proyecto se utilizan los sensores MPXM2102 y LM35 para medir las variables asociadas a la presión atmosférica y temperatura ambiente, respectivamente.

Sensor de Presión Atmosférica MPXM2102

El sensor de presión atmosférica utilizado en este proyecto es el MPXM2102 de la marca *Freescale Semiconductor*, el cual cuenta con una compensación de la temperatura ambiente para reducir el efecto de esta en las mediciones realizadas con este sensor. Las características principales de este sensor de presión son:

- Compensación de la temperatura ambiente para un rango de 0°C a 85°C.
- Voltaje de alimentación de 2V a 16V de CD.
- Salida de voltaje diferencial.
- Voltaje de salida máximo de 40mV con 10V de alimentación, generando una medición máxima de 100kPa.

En este proyecto el sensor de presión es energizado con una fuente de 3V de CD como se muestra en la Figura 3.13, mismo que es utilizado para generar voltaje de referencia para la activación del convertidor A/D a través de un divisor de tensión.

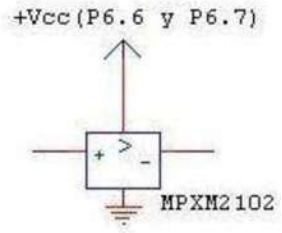


Figura 3.13 Conexión del sensor de presión.

Ahora bien, las salidas diferenciales del sensor de presión son directamente conectadas al microcontrolador sin la necesidad de pasar por una etapa de acondicionamiento, debido a que los voltajes de salida se encuentran en un rango adecuado para la entrada del convertidor A/D.

Sensor de Temperatura Ambiente LM35

El sensor utilizado para la medición de la temperatura es el LM35 de la marca *National Semiconductor*. Este sensor proporciona un voltaje de salida no diferencial directamente proporcional a la temperatura ambiente en grados centígrados. El sensor

utilizado es el del encapsulado metálico TO-46 el cual puede medir temperaturas ambiente en un rango de -55C a 150°C.

Para poder obtener el rango de mediciones mencionado el sensor de temperatura tiene que ser conectado a una fuente de ±Vcc, debido a que requiere de una fuente negativa para la generación de las mediciones negativas. El sensor tiene que ser conectado a la fuente negativa mediante una resistencia la cual se calcula una vez que se conoce el valor de la fuente que se va a utilizar y la información del fabricante respecto a la corriente en la terminal de salida. En este caso el voltaje a utilizar es Vcc = ±8.4V y la corriente requerida es de 50µA. La resistencia requerida se calcula mediante la expresión mostrada en la Ecuación (3.3).

$$R = \frac{8.4V}{50\mu A} = 168k\Omega$$

(3.3)

En la Figura 3.14 se muestra la implementación de este arreglo para hacer que el sensor pueda medir desde -55°C hasta 150°C.

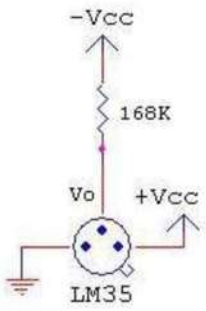


Figura 3.14 Arreglo del sensor de LM35 para mediciones negativas.

El voltaje de salida del transductor de temperatura corresponde a 10mV/°C, por lo que si se mide la temperatura máxima de 150°C entonces se tiene un voltaje de salida de 1.5V. Este voltaje sobrepasa el rango de entrada al convertidor A/D de ± 634.5mV, por lo cual se debe de acondicionar la señal para poder ser ingresada al convertidor. Para el cálculo del rango de voltaje de entrada al convertidor se utiliza el voltaje de referencia externo de 1.269V. Además la ganancia utilizada para la señal del sensor de temperatura es unitaria. El cálculo del rango del voltaje de entrada se obtiene por medio de la expresión (3.4).

$$V_{FSR} = \frac{V_{REF}/2}{GANANCIA_{PGA}} = \frac{1.269V/2}{1} = 634.5mV \quad (3.4)$$

La señal de salida del sensor de temperatura es ajustada de acuerdo a la gráfica mostrada en la Figura 3.15, en donde se muestran los rangos de valores que pueden ingresar al convertidor Vo respecto a a los voltajes del sensor Vi. Estos valores corresponden al voltaje de salida del sensor de temperatura Vo una vez que la señal ha sido acondicionada.

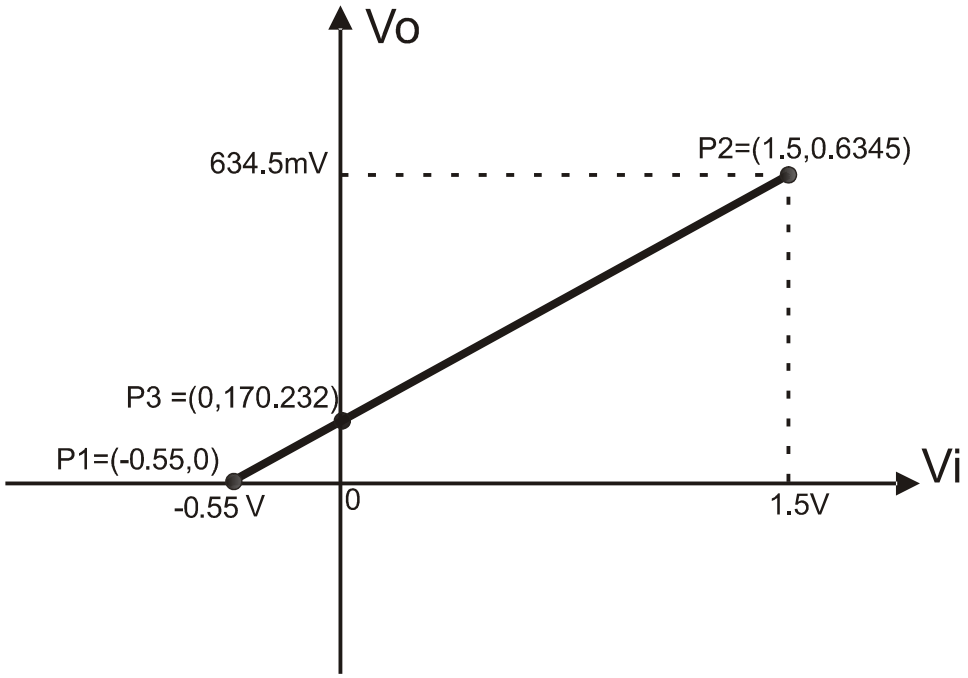


Figura 3.15 Voltaje de salida del sensor acondicionado.

Una vez ya que se tiene el rango de voltajes deseado se prosigue hacer el cálculo de los elementos que forman la etapa de acondicionamiento de la señal. Partiendo de la ecuación de la recta,

$$Vi = \frac{Vo - Vb}{m} \quad (3.5)$$

se calcula la pendiente, tomando en cuenta que esta es la ganancia Av. Se utilizan los rangos de voltaje tanto de la entrada al convertidor como el rango del voltaje de salida del transductor de la siguiente manera:

$$m = \frac{(0.6345-(0))V}{(1.5-(-0.55))V} = 0.309512 \quad (3.6)$$

Como se puede observar por el comportamiento de la pendiente, la ganancia es menor que 1, por lo que esta etapa de acondicionamiento debe realizar una atenuación del voltaje.

Despejando Vo de la ecuación y sustituyendo el valor de la ganancia se tiene,

$$Vo = m * Vi + Vb \quad (3.7)$$

$$Vo = (0.309512) * Vi + Vb \quad (3.8)$$

Para calcular Vb se utiliza el punto P1 de la Figura 3.15 tomando los valores de en las coordenadas correspondientes Vi=-0.55 y Vo=0 se tiene:

$$Vb = 0 - (0.309512) * (-0.55) = 0.170232 \quad (3.9)$$

Con lo cual la ecuación para el voltaje de salida queda:

$$Vo = (0.309512) * Vi + 0.170232 \quad (3.10)$$

Para comprobar los resultados se le dan valores a al voltaje de entrada máximo Vi con lo que se tiene como resultado:

$$Vo = (0.309512) * (1.5) + 0.170232 = 0.6345 \quad (3.11)$$

de donde se observa que corresponde al voltaje máximo de entrada para el convertidor. Ahora se comprueba para el voltaje mínimo de entrada Vi=-.55V:

$$Vo = (0.309512) * (-0.55) + 0.170232 = 0.0000004$$

con lo que se puede observar que es prácticamente cero y también corresponde al voltaje de entrada mínimo para el convertidor A/D.

Una vez que se comprobó la ecuación obtenida el siguiente paso está asociado al cálculo de las componentes para la implementación con un amplificador de instrumentación, como se muestra a continuación en el siguiente esquema de la Figura 3.16.

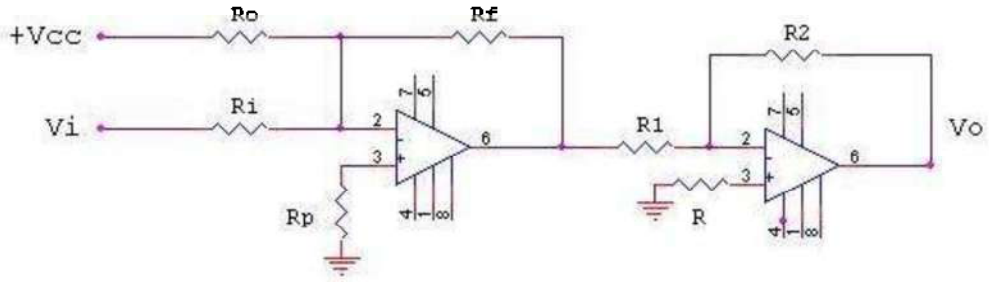


Figura 3.16 Amplificador de Instrumentación.

La fórmula para la ganancia de amplificación en la configuración del sumador inversor es:

$$A_v = \frac{R_f}{R_i}$$

Para el diseño de Rf se seleccionó un valor relativamente grande, tal que Ri no cargue al transductor conectado al nodo Vi. Seleccionando una Rf=330kΩ y resolviendo la ecuación para Ri se tiene:

$$R_i = \frac{R_f}{A_v} = \frac{330\text{k}\Omega}{0.309512} = 1.0066194M\Omega \quad (3.12)$$

Para poder obtener el voltaje Vb=0.170232V, es necesario realizar un divisor de tensión, en donde interviene un voltaje de entrada Vcc=7.7V, el cual es obtenido mediante un circuito regulador de voltaje utilizando un diodo zener de 7.5V. El valor de 7.7V se obtuvo realizando una medición de tensión en terminales del regulador de voltaje debido a que se tiene que considerar los valores reales que se están manejando para evitar errores.

Partiendo de,

$$V_b = \frac{R_f * V_{cc}}{R_o} \quad (3.13)$$

se resuelve para Ro y se tiene,

$$R_o = \frac{R_f * V_{cc}}{V_b} = \frac{330\text{k}\Omega * 7.7\text{V}}{0.170232V} = 14.92668M\Omega \quad (3.14)$$

La resistencia de compensación Rp se define por,

$$R_p = R_i \left|\left| R_f \right|\right| R_o = 248.414k\Omega \quad (3.15)$$

Ahora bien para la segunda etapa del amplificador de instrumentación en configuración de inversor, se requiere simplemente una ganancia unitaria. Por lo tanto se escogieron las resistencias R1=R2=2.2kΩ y su resistencia de compensación es,

$$R = R_1 || R_2 = 1.1k\Omega \quad (3.16)$$

3.3.5 Baterías de Alimentación

Las baterías de alimentación utilizadas en este proyecto son baterías recargables de la marca GP, las cuales proporcionan un voltaje de 8.4V y una corriente 170mA por hora. Debido a las características de estas baterías se puede considerar que cuentan con una buena capacidad para energizar determinados componentes que consumen poca potencia.

De acuerdo con las características de determinados componentes, como el sensor de temperatura LM35 y los amplificadores operacionales LF356, éstos requieren de una alimentación de voltaje negativo para operar de acuerdo a las necesidades de este proyecto. Por lo tanto, en este proyecto se cuenta con dos baterías de este tipo, con las cuales se forma un banco de baterías que proporciona ±8.4V. Debido a las pruebas realizadas con las baterías se observó que cuando estas son recargadas el voltaje que proporcionan es mayor al marcado por el fabricante. Por lo que es necesario regularlo a un valor constante. La regulación de este voltaje se realizó por medio de diodos tipo zener de 7.5V (1N4337A). Por otra parte se realiza lo mismo para obtener el voltaje de alimentación para el microcontrolador pero utilizando un diodo tipo zener de 3.3V [Malvino 2000].

En la Figura 3.17 se muestra el banco de baterías con los diodos zener que proporcionan la regulación del voltaje.

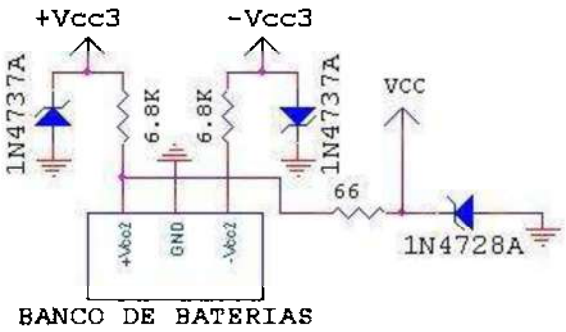


Figura 3.17 Banco de baterías.

Para el cálculo de la resistencia se utiliza el análisis nodal conociendo el voltaje de operación del zener que es de 7.5V, así como su corriente de prueba que es de 34mA y el voltaje de las baterías que es de 8.4V. Valor comercial más cercano a esta resistencia es de 33Ω.

$$R = \frac{V_{cc} - V_{zener}}{I_{zener}} = \frac{8.4V - 7.5V}{34mA} = 26.47\Omega \cong 33\Omega \qquad (3.17)$$

Sin embargo en este trabajo se eligió un valor de R>>33 Ω con el objeto de disminuir el flujo de corriente a través del zener. Se eligió un valor de 6.8kΩ.

De igual manera para el cálculo de la resistencia R2 aplicando análisis nodal conociendo el voltaje del banco de baterías que es de 8.4V y el voltaje de operación del zener que es de 3.3V (1N4728A) así como su corriente que es de 76mA. Calculando R2 se tiene,

$$R2 = \frac{V_{cc} - V_{zener}}{I_{zener}} = \frac{8.4V - 3.3V}{76mA} = 67\Omega \cong 66\Omega \quad (3.18)$$

en donde se utilizan dos resistencias de 33 Ω para obtener el valor más cercano de 66 Ω.

De igual manera que para la resistencia R, se recomienda utilizar una resistencia R>>66Ω de tal manera que el flujo proveniente de la batería se disminuya.

3.3.6 Diagrama Esquemático del Medidor de Presión y Temperatura Ambiente.

En la Figura 3.18 se presenta el diagrama esquemático del diseño del sistema de medición mínimo basado en el microcontrolador en cual se presentan todos los componentes utilizados para el funcionamiento de este prototipo. Este diagrama es el circuito mínimo propuesto para la implementación de este medidor, el cual operaría de manera independiente sin el uso de la tarjeta de desarrollo utilizada en este proyecto. Sin embargo, el trabajo realizado en esta tesis emplea el uso de dicha tarjeta debido a que fue implementado en tabletas de trabajo y no se realizó la implementación del circuito mínimo de forma independiente.

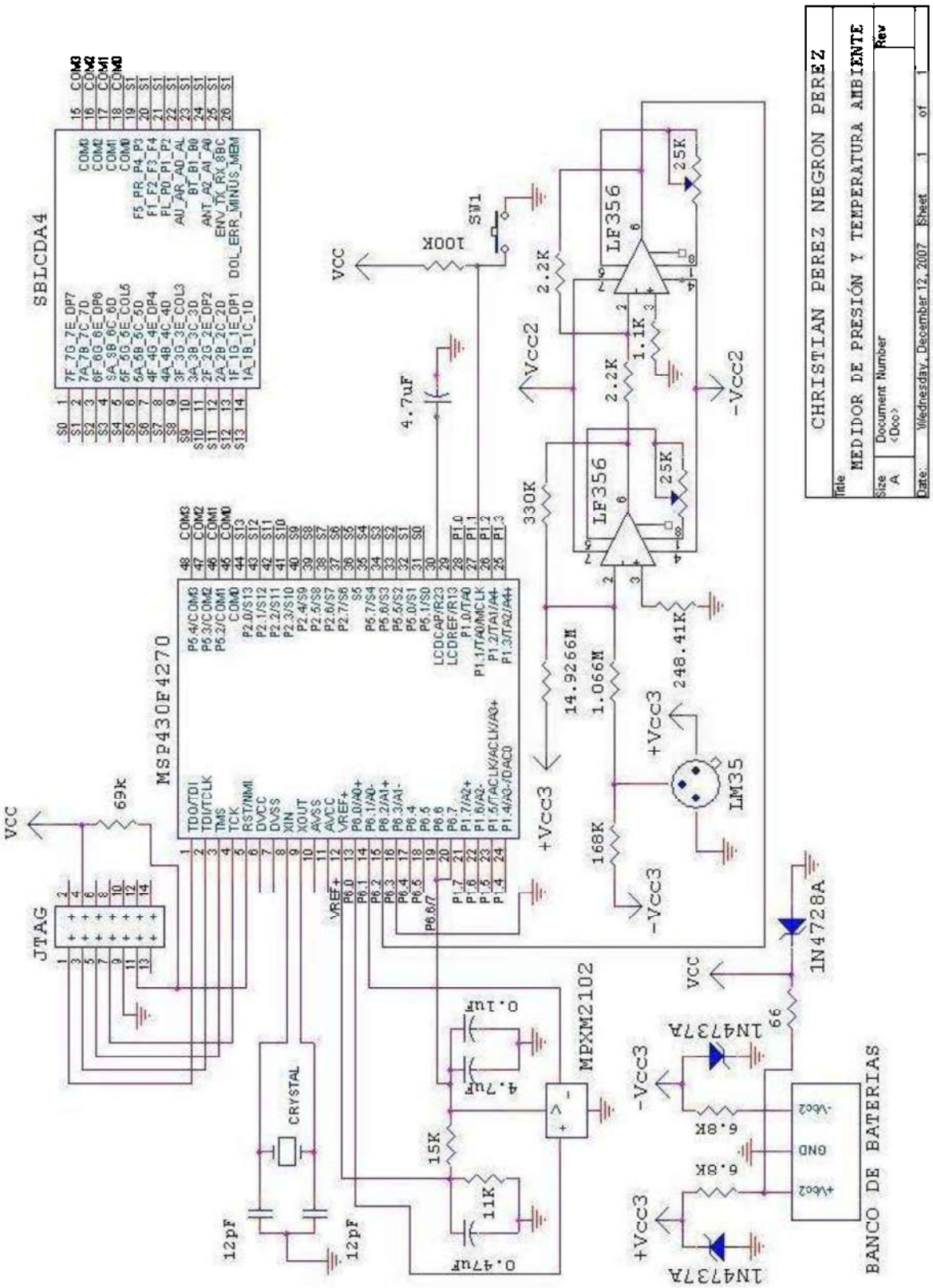


Figura 3.18 Diagrama esquemático del medidor de presión y temperatura ambiente.

3.4 Conclusiones

Debido a las características con las que cuenta el microcontrolador MSP430F4270 se puede apreciar que cumple con los requerimientos para este proyecto porque es un microcontrolador de bajo consumo de potencia y cuenta con memoria *Flash*, la cual es de gran utilidad debido a que en ésta se puede almacenar diferente tipo de información ya sea de código o de mediciones obtenidas. Una gran ventaja de este microcontrolador es que cuenta con un convertidor de analógico a digital, integrado dentro de este, de muy buena resolución lo cual reduce el uso de *hardware* adicional. Este microcontrolador cuenta con cinco modos de operación de bajo consumo de potencia los cuales desactivan diferentes módulos mientras no se estén ocupando, esto hace que las baterías de alimentación tengan un mayor periodo de operación.

El *hardware* complementario que se utilizó consiste en dispositivos que también consumen muy poca potencia como es el caso de los sensores para las mediciones atmosféricas y los amplificadores operacionales utilizados para el acondicionamiento de la señal del sensor de la temperatura.

Algo de importante que se observó durante el desarrollo de este prototipo, es que las baterías, utilizadas para energizar los diferentes dispositivos, muestran variaciones de voltaje que afectan el desempeño de las mediciones. Esto se detectó al realizar las pruebas con el sensor de temperatura, debido a que la etapa de acondicionamiento de este, requiere de voltajes muy precisos porque son voltajes muy pequeños los que se están manipulando y una mínima variación hace que la medición sea errónea. Para remediar este hecho, el voltaje proporcionado por las baterías se regula mediante diodos tipo zener.

Finalmente, en este capítulo se propone un sistema mínimo para la operación del medidor de temperatura y presión ambiental. Este sistema mínimo incluye las etapas básicas para energizar el microcontrolador, la etapa de acondicionamiento de los sensores, la incorporación de la pantalla LCD y el botón de control del medidor.

Capítulo 4

Diseño del Software del Sistema de Medición

En este capítulo se presenta la implementación del *software* para este proyecto de tesis, el cual cuenta con dos versiones de código. Por otro lado, se detalla el proceso a seguir para programar el microcontrolador MSP430F4270 utilizado en este proyecto. También se explica la programación de los módulos más importantes para el adecuado funcionamiento del microcontrolador.

4.1 Proceso de Desarrollo de un Programa para el MSP430F4270

El desarrollo de un programa para el microcontrolador MSP430F4270 consta de varias etapas, las cuales se describen en la Figura 4.1. Este proyecto de tesis cuenta con dos opciones de programa para el microcontrolador. La primera versión del programa se basa en el uso de un botón externo utilizado para controlar las mediciones, es decir, cada vez que el botón es presionado se realiza la medición de alguna de las variables de interés, mostrando los resultados en una pantalla LCD de cristal líquido. La segunda opción consiste en un programa controlado por un reloj, realizando las mediciones de cada una de las variables cada vez que se cumpla con un tiempo determinado por el usuario. Estas mediciones son almacenadas en los bloques A y B de la memoria *Flash* del microcontrolador. Una vez que la memoria se llena los nuevos datos obtenidos se van reescribiendo en las localidades de memoria. El intervalo de tiempo para que se realicen las mediciones es definido en el código y este depende de las necesidades del usuario, es decir, con qué frecuencia se requiere monitorear las variables. Ambas versiones de código fueron desarrollados en el *software IAR Embedded Workbench IDE* [Manual_Texas_2 2003], el cual se puede conseguir de forma gratuita en la página de *Texas Instruments* así como las últimas actualizaciones del mismo. En dicho *software* se desarrollan los proyectos que requiere cualquier microcontrolador de la familia MSP430 de microcontroladores. Además, este *software* cuenta con un compilador que puede utilizar códigos en tres opciones de

lenguajes de programación diferentes: C, C++ y lenguaje ensamblador [Manual_Texas_3 2005]. En la Figura 4.1 se resume el proceso para desarrollar un programa para el microcontrolador MSP430F4270 y en seguida se presenta una breve explicación de cómo se realiza un programa en el software *IAR Embedded Workbench IDE*.

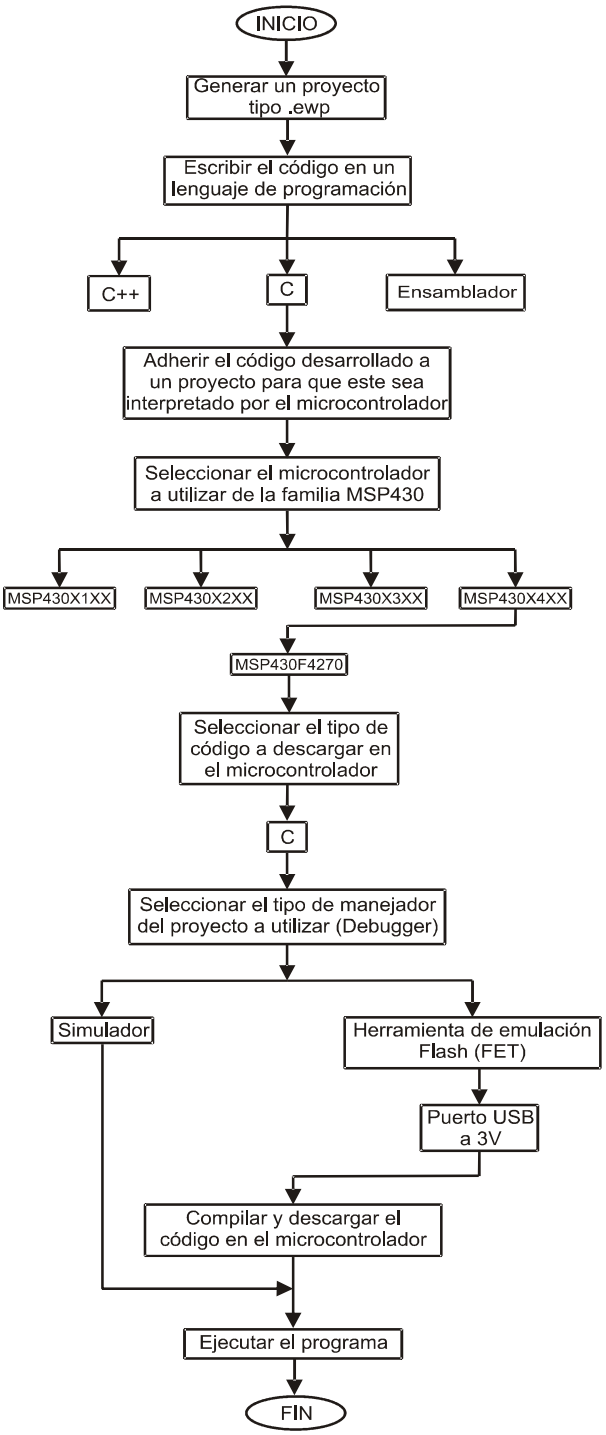


Figura 4.1 Proceso del desarrollo de un programa para el MSP430F4270.

4.1.1 Creación de un Proyecto

El primer paso para la programación del microcontrolador MSP430F4270 consiste en la creación de un proyecto del tipo .ewp el cual se genera seleccionando la opción de "crear un nuevo proyecto en espacio de trabajo actual". Esta opción es proporcionada al momento de abrir el *software IAR Embedded Workbench IDE*. En la Figura 4.2 se muestra la ventana de inicio donde se presentan diferentes opciones para la creación de un proyecto o bien, se presentan ciertas opciones para utilizar con proyectos ya existentes así como ejemplos de aplicaciones en caso de ser nuevo en el uso de este *software*.

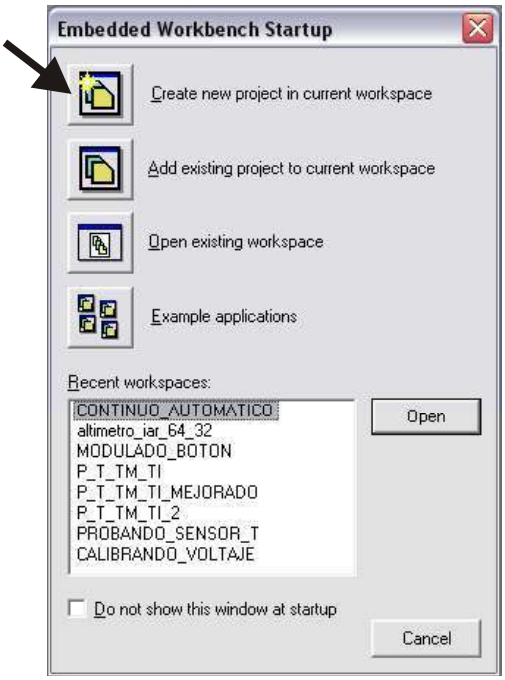


Figura 4.2 Creación de un nuevo proyecto.

Una vez que se ha seleccionado la creación de un nuevo proyecto aparece una nueva ventana , como se observa en la Figura 4.3, en donde se especifica que el nuevo proyecto a crear es un proyecto que se encuentra vacío, es decir, es un proyecto que no contiene las características por defecto que proporciona el *software*. Como se puede observar en la Figura 4.3 se indican tres tipos de plantillas posibles a utilizar, el lenguaje ensamblador, C++ y C.

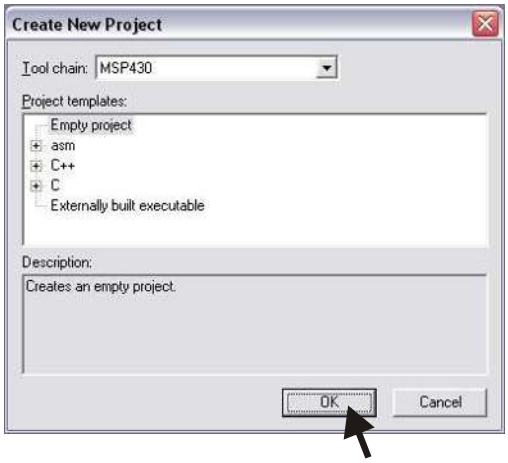


Figura 4.3 Selección de un proyecto vacio.

Una vez que se ha elegido la creación de un proyecto vacio, el siguiente paso es guardar este proyecto como tipo .ewp. Es recomendable seleccionar una ruta en donde se desean ir almacenando los proyectos con nombres que proporcionen una idea de lo que realiza el proyecto ver Figura 4.4.

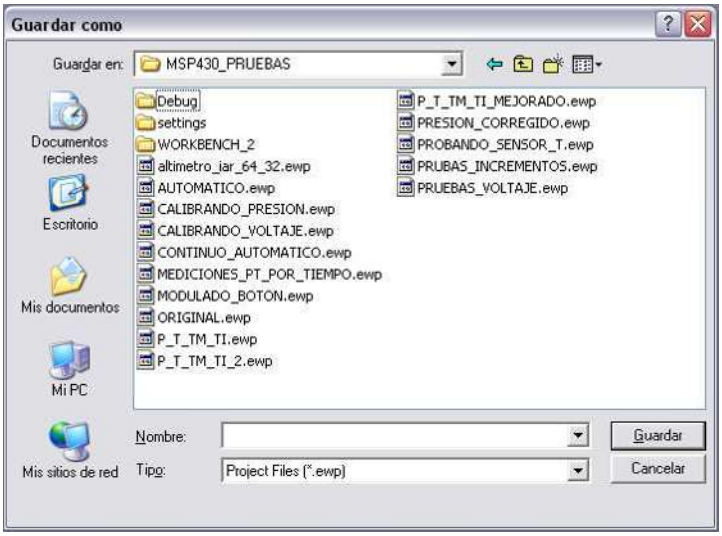


Figura 4.4 Guardar el proyecto como tipo .ewp.

4.1.2 Escritura del Código en un Lenguaje de Programación

Como ya se ha mencionado el compilador de este *software* puede interpretar tres tipos de lenguajes de programación: el ensamblador, el C++ y el C. Los códigos desarrollados para este proyecto de tesis fueron escritos en lenguaje C. Para poder empezar a redactar el código del proyecto, es necesario abrir un nuevo documento en blanco dentro del nuevo proyecto creado. Este documento se puede abrir dando un click en el ícono de la hoja en blanco o bien siguiendo la siguiente ruta: File – New - File. Haciendo esto se crea un editor de texto en blanco el cual viene etiquetado como "Untitled1" (sin título 1), en donde se escribe el código a implementar. Como se puede observar en la Figura 4.5 en la parte izquierda se muestra una ventana llamada *workspace* o área de trabajo, en cuya ventana se presenta el proyecto vacio que fue generado. En este caso se generó un proyecto con el nombre de EJEMPLO.ewp para ilustrar los pasos a seguir.

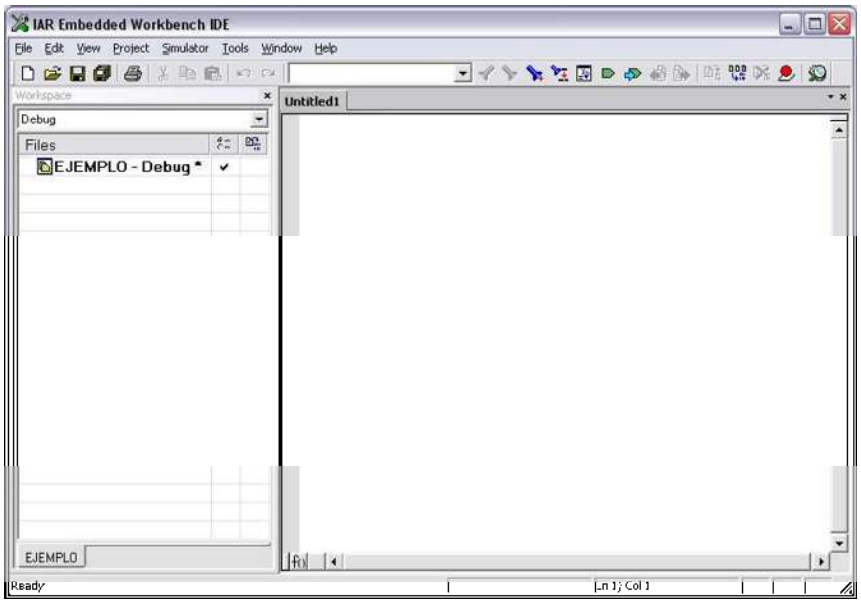


Figura 4.5 Abrir el editor de Texto.

Una vez que ya se tiene el código escrito en el editor de texto es necesario guardarlo con la extensión requerida para el tipo de lenguaje de programación utilizado. Este paso se realiza siguiendo la siguiente ruta: File-Save – Save As, en donde aparece la ventana que se muestra en la Figura 4.6. Es necesario escribir la extensión para evitar errores en el

momento de adherir el código al proyecto o bien evitar que el compilador arroje un error durante la compilación del código donde mencione que el código no es compatible al microcontrolador o bien que no puede encontrar la carpeta donde se encuentra el código. Por ejemplo, si el código fue escrito en lenguaje C como EJEMPLO.C, o bien agregar la extensión requerida para cada lenguaje (.cpp o .asm, para C++ y ensamblador, respectivamente). Por otro lado, cada uno de los códigos desarrollados deben de ser guardados en la misma carpeta donde se encuentran los proyectos .ewp para evitar errores al momento de la ejecución del programa.

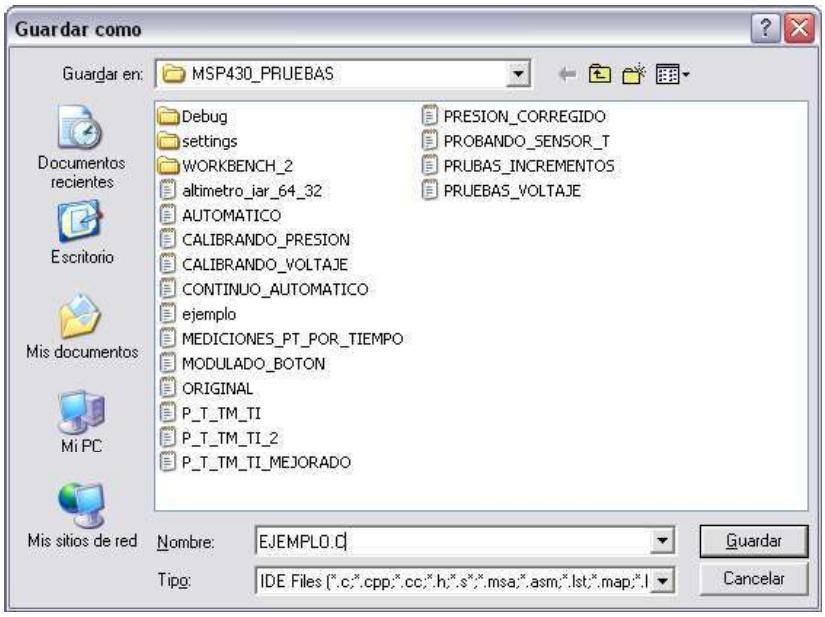


Figura 4.6 Guardar el código con su respectiva extensión.

Es de gran importancia tomar en cuenta la demanda de memoria del código que se va a ejecutar en el microcontrolador, es decir, se debe de cuidar la dimensión del código haciendo que este contenga solamente las líneas de programa necesarias y que además sea un código eficiente para no sobrepasar el tamaño permitido por el dispositivo. Si esto ocurre al momento de la compilación se arroja un error donde indica que el tamaño del código ha excedido la capacidad de la memoria de código y no permite que el programa sea ejecutado.

4.1.3 Adherir el Código al Proyecto

Una vez que se cuenta con el código almacenado en el mismo directorio donde se encuentra el proyecto .ewp, el siguiente paso es adherir el código al proyecto para que el código pueda ser grabado en el microcontrolador. Para realizar este paso se debe de seguir la siguiente ruta: Project – Add Files…, se abre una ventana donde se encuentran los códigos disponibles para que sean adheridos. El código deseado se selecciona así como la opción denominada "Abrir". Una vez que el código es seleccionado se puede observar que este es adherido al área de trabajo como se muestra en la Figura 4.7 en la ventana del *workspace*. El código en uso es marcado con un asterisco el cual indica que es el código actual a ser compilado y ejecutado.

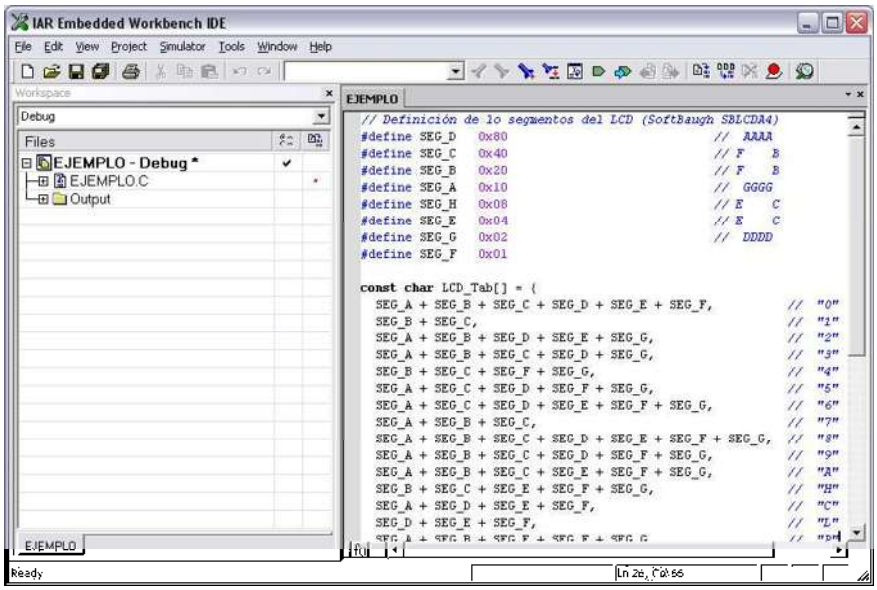


Figura 4.7 Código adherido al proyecto.

Una vez que se ha realizado este paso entonces se ha creado un proyecto para un microcontrolador MSP430F4270. Los siguientes pasos consisten en ajustar las características del proyecto de acuerdo al microcontrolador utilizado.

4.1.4 Selección del Microcontrolador

La selección del microcontrolador es una de las etapas fundamentales en la implementación de un proyecto. Esta elección es realizada cuando se ajustan las características del proyecto. Este paso se realiza abriendo la ventana de opciones del proyecto siguiendo la siguiente ruta: Project – Options, o en su defecto presionando ALT + F7. Cuando se realiza este acceso se abre una ventana en la cual se presentan diferentes categorías, en este paso nos tenemos que ubicar en la categoría de opciones generales, en donde se presenta la selección del dispositivo de la familia MSP340 a utilizar. En la Figura 4.8 se muestra la selección del microcontrolador MSP3430F4270 dentro de la categoría de opciones generales, seleccionando la configuración MSP430x4xx donde se encuentra el microcontrolador utilizado en este proyecto de tesis.

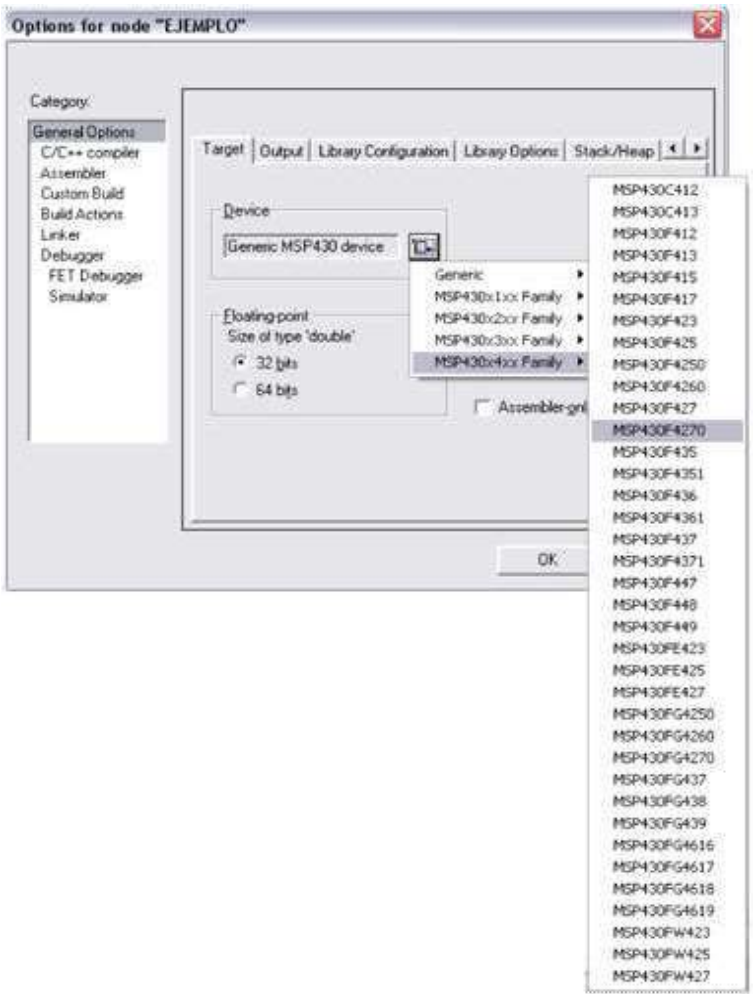


Figura 4.8 Selección del Microcontrolador MSP430F4270.

De acuerdo con los trabajos realizados con este *software* se tiene como experiencia que para poder tener acceso a la ventana de opciones, es necesario que tenga abierta la ventana de área de trabajo o *workspace*, de no ser así no se puede tener un acceso a la ventana de opciones lo cual impide seleccionar el microcontrolador así como otras de las características necesarias para la realización de un proyecto.

4.1.5 Selección del Compilador del Código

Una vez que se ha seleccionado el microcontrolador a utilizar es necesario seleccionar el tipo de compilador de código a requerido. En este caso debido a que el programa fue realizado en lenguaje C se tiene que seleccionar su compilador apropiado. La selección del compilador se realiza dentro de la misma ventana de opciones, solamente que ahora se tiene que seleccionar la categoría de compilador C/C++ como se muestra en la Figura 4.9. Debido a que se ha cargado al proyecto un código tipo .C, la elección del compilador se realiza de forma automática por lo que solo resta asegurarse que se especifique el compilador adecuado.

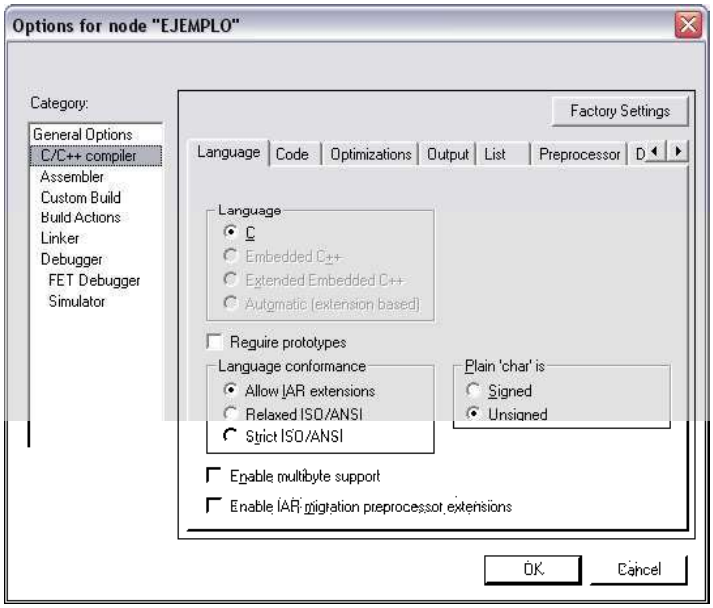


Figura 4.9 Selección del compilador del código.

4.1.6 Selección del Manejador del Proyecto

Este *software* cuenta con dos opciones para la ejecución de los programas del microcontrolador. El primero de ellos consiste en la ejecución del programa mediante simulación y el segundo es por medio de la Herramienta de Emulación *Flash* (FET) en donde se descarga el programa en el microcontrolador. Esto significa que al seleccionar este manejador FET, el usuario puede realizar diferentes actividades cuando el microcontrolador se encuentre en funcionamiento. Por ejemplo, se puede monitorear el estado de cada una de las variables involucradas en el código, así como también se puede tener acceso a las localidades de memoria del microcontrolador. La selección del manejador del proyecto se realiza dentro de la misma ventana de opciones que se ha estado mencionado pero ahora en la categoría *Debugger*. En la Figura 4.10 se muestran los dos tipos de manejadores (FET *Debugger* y *Simulator*) con los que cuenta este *software* y se pude apreciar las selección del manejador *FET*.

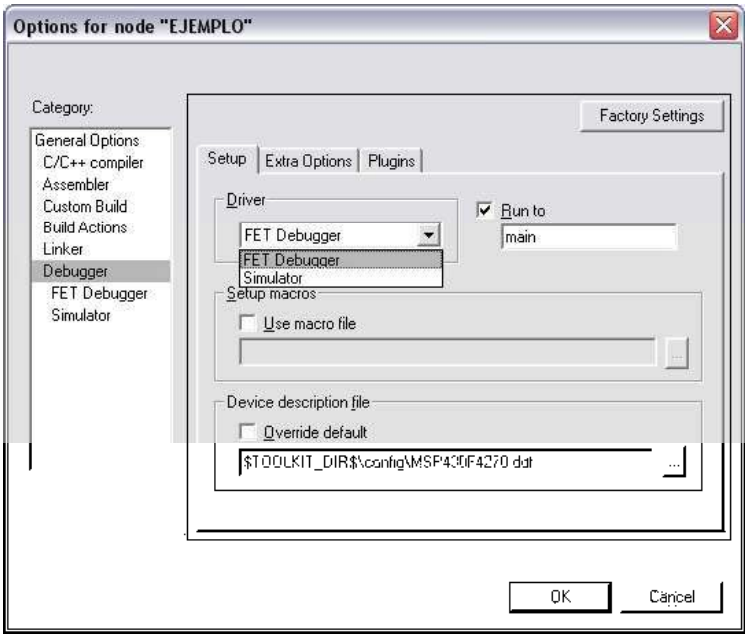


Figura 4.10 Selección del manejador del proyecto.

4.1.7 Selección del Puerto para Programar el Microcontrolador

Una vez que se ha seleccionado la herramienta de emulación *FET*, el siguiente paso es seleccionar el puerto por el que se va a realizar la programación del microcontrolador. Debido a las características de la tarjeta utilizada en este proyecto se tiene que seleccionar el puerto USB, el cual utiliza un voltaje de 3V de CD. En la Figura 4.11 se muestra la selección de dicho puerto.

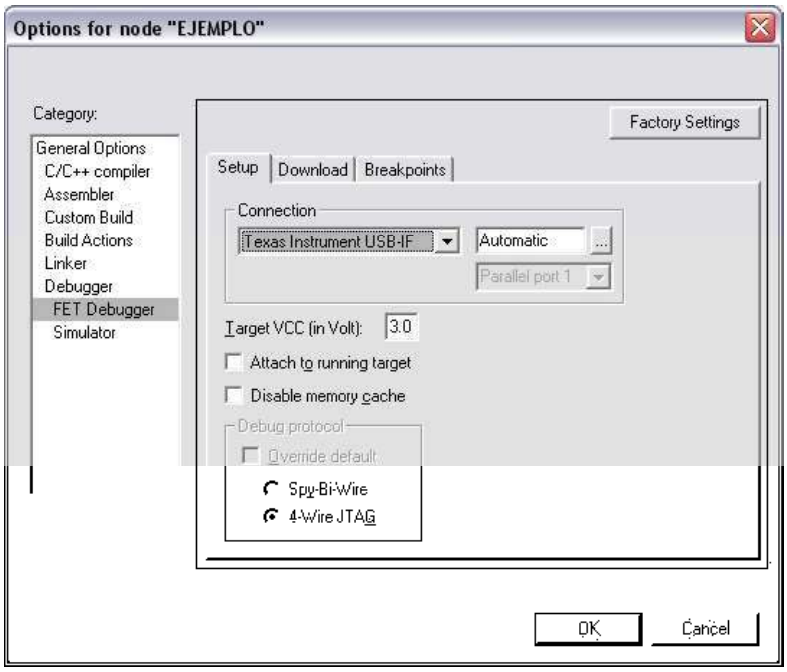


Figura 4.11 Selección del puerto USB para la programación del Microcontrolador.

4.1.8 Compilación y Ejecución del Programa

La última etapa para la programación del microcontrolador consiste en compilar el código una vez que se han ajustado las características necesarias para el proyecto implementado. Para poder realizar la compilación del código es necesario guardar el área de trabajo o *workspace*, el cual se almacena como tipo *.eww*, lo cual se realiza siguiendo la siguiente ruta: File – Save Workspace. Este archivo se guarda en la misma carpeta donde fue almacenado el proyecto .ewp y el código en C. En este punto se cuenta ya con todos los requerimientos necesarios para compilar el programa. En la pestaña *Project* de la pantalla del *software* se encuentran los comandos para compilar y ejecutar el programa (ver

Figura 4.12). El programa se compila con el comando *compile* y los resultados de la compilación son enlistados como advertencias ó errores en caso de que estos existan. Si el código se encuentra libre de estos errores solamente se mostrará una línea donde se indican que no hay errores y entonces el programa se encuentra listo para ser ejecutado.

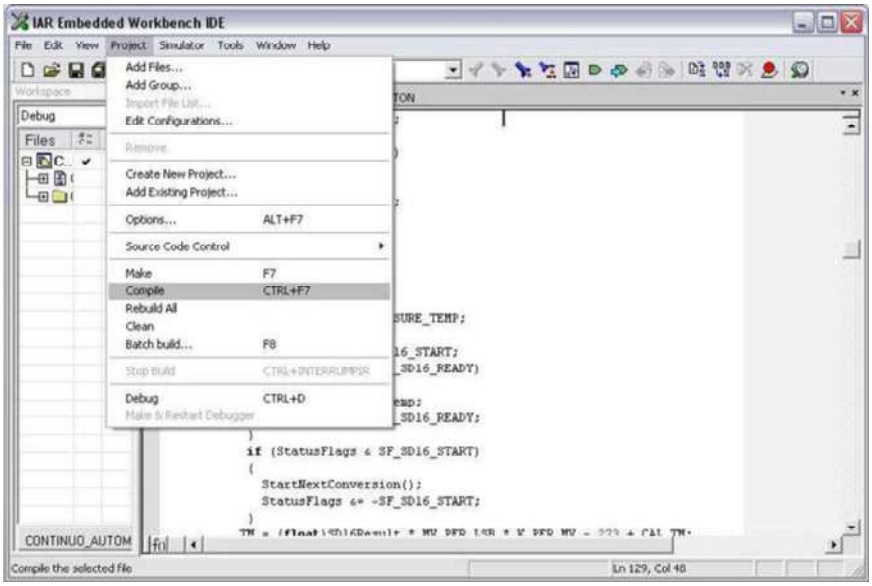


Figura 4.12 Compilación del código.

Una vez que el código ha sido compilado se puede ejecutar el programa. En la Figura 4.12 en la misma pestaña de *Project* se encuentra el comando *Debug*, el cual es el que envía el código al microcontrolador y este genera una nueva pestaña llamada *Debug* en la cual se presenta la opción de *Go* como se muestra en la Figura 4.13.

Ya que se está ejecutando el programa se pueden utilizar diferentes herramientas que este *software* proporciona como la ventana *watch* ubicada en la pestaña *View*, la cual es utilizada en este proyecto para monitorear las variables deseadas. Otra ventana de gran ayuda para este proyecto y esencial para el código en el que se almacenan las mediciones dentro de la memoria *Flash* es la ventana de *memory* que también se encuentra ubicada dentro de la pestaña *View*. Esta ventana permite monitorear las localidades de memoria del microcontrolador. En este proyecto esta ventana fue utilizada para monitorear las localidades de memoria de las bloque A y B de la memoria *Flash* donde se van almacenando la mediciones tomadas en intervalos de tiempo determinados.

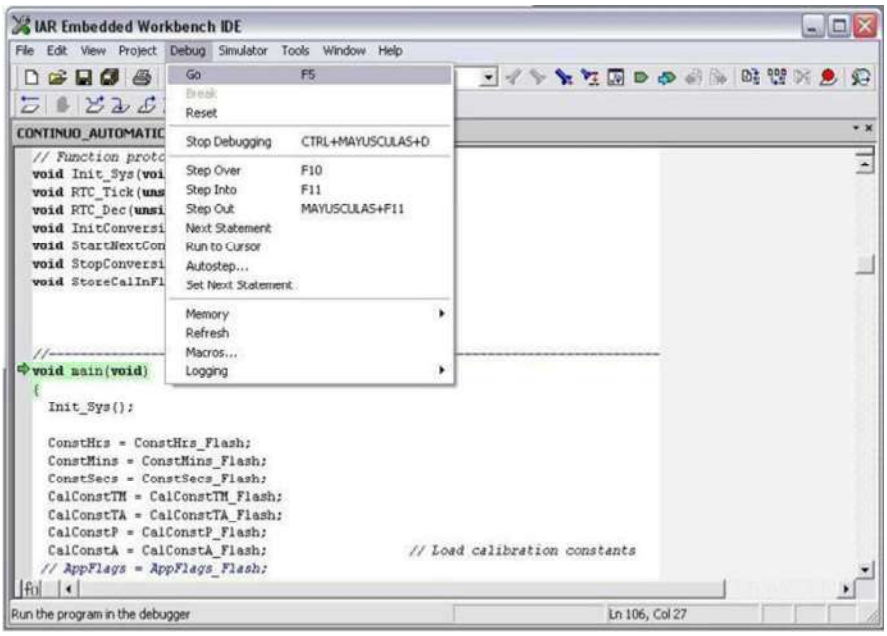


Figura 4.13 Comando *Go* para correr el programa.

4.2 Estructura del Programa en Lenguaje C

En este proyecto se implementaron dos códigos para la medición de las variables meteorológicas de temperatura y presión atmosférica. El primer programa se basa en el control de las mediciones de las variables atmosféricas por medio del uso de un botón conectado al Puerto 1 del microcontrolador. Cada vez que es presionado este botón se realiza una medición de una variable diferente. La segunda versión del programa consiste en un programa que realiza las mediciones de forma automática cada vez que se cumple con un intervalo de tiempo establecido por el usuario. Cuando se tienen todas las mediciones capturadas estas son almacenadas en los bloques de la memoria *Flash*. Ambos códigos basan su funcionamiento en interrupciones, las cuales permiten activar el microcontrolador cada vez que se realiza un llamado a estas. Los programas implementados basan su funcionamiento en cuatro tipos de interrupciones, las cuales son atendidas por la rutina de servicio de interrupciones ISR (*Interrupt Service Routine*). Las interrupciones utilizadas en este proyecto son la interrupción del temporizador básico (*Basic Timer BT*), la del temporizador perro guardián (*Watch Dog Timer WDT*), la del puerto 1 (*Port_1*) y la del convertidor A/D SD16_A. Todas estas interrupciones pueden hacer que el dispositivo sea activado desde su modo de operación de bajo consumo de potencia LPM3 y así disparar el

evento de procesamiento dentro del programa *main()*. A continuación se presenta la descripción de las dos versiones de los programas mencionados.

4.2.1 Programa Basado en un Botón

En la Figura 4.14 se presenta el diagrama de flujo de la función principal del programa. Como se puede observar al inicio de esta función se realizan una serie de habilitaciones a módulos tales como el Puerto_1, el controlador de la pantalla LCD módulo (LCD_A) y el BT. A diferencia de los demás módulos el WDT es deshabilitado. En seguida de esto las variables que son utilizadas para la calibración de las mediciones son cargadas a la memoria *Flash*, se habilitan las interrupciones y se activa el modo de operación de bajo consumo de potencia LPM3. Enseguida se ejecuta un ciclo repetitivo en donde se espera la activación de alguna de las interrupciones para que estas realicen sus actividades correspondientes.

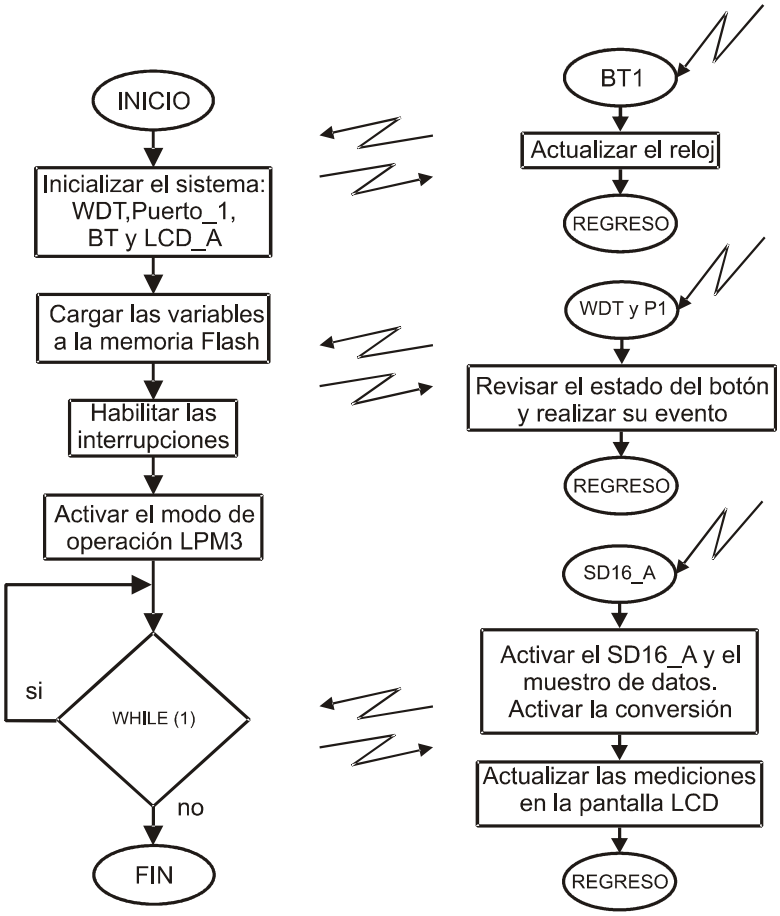


Figura 4.14. Diagrama de flujo de la sección *main()* del programa en C utilizando un botón de control.

La primera interrupción del programa está asociada al reloj de tiempo real, la cual es activada cada segundo y, por lo tanto, ocasiona un incremento en la variable del tiempo siendo la hora actualizada segundo a segundo. La siguiente interrupción corresponde a la del botón conectado al puerto 1, la cual se mantiene revisando el estado del botón y cada vez que este es presionado se genera una interrupción correspondiente a la combinación de las del WDT y del Puerto_1. Por lo tanto, cada vez que el botón es presionado se activa la interrupción del convertidor A/D la cual activa el muestreo la conversión de los datos. Finalmente, el resultado obtenido es desplegado en la pantalla LCD.

Las mediciones de las variables son realizadas en un orden predeterminado, como se muestra en la Figura 4.15. En esta figura se ilustra un diagrama de estados que representa el funcionamiento del botón dentro del programa. Para controlar el flujo a través de los diferentes estados se utilizó una variable que indica el estado lógico del botón (B).

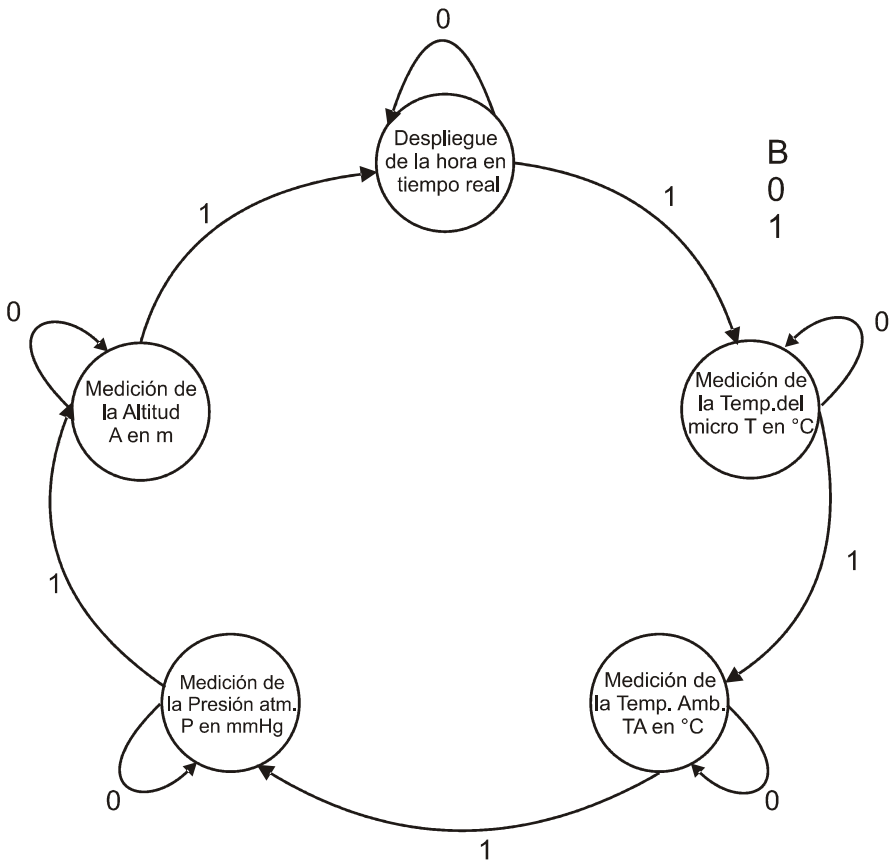


Figura 4.15 Diagrama de estados para la relación del botón con las mediciones.

Cuando se tiene la variable B = 0 entonces el botón no está siendo presionado y cuando B = 1 significa que el botón es presionado por menos de un segundo. En cuanto el programa es ejecutado la pantalla LCD se mantiene desplegando permanentemente la hora, hasta el momento que el botón es presionado por menos de un segundo y entonces se realiza la medición de la temperatura del microcontrolador. Si el botón es nuevamente presionado por menos de un segundo ahora se realiza la medición de la temperatura ambiente, siguiéndole la medición de la presión atmosférica y la medición de la altitud. Una vez que se han realizado todas las mediciones el programa regresa a mostrar la hora y queda en espera de que el botón se presione nuevamente. Es importante mencionar que cada vez que se realiza una medición de alguna de las variables, éstas son desplegadas en la pantalla LCD.

4.2.2 Programa Controlado por Intervalos de Tiempo

En la Figura 4.16 se muestra el diagrama de flujo de la sección principal del programa basado en tiempo en donde al principio de esta sección se realizan un serie de habilitaciones a los módulos el Puerto_1, LCD_A y el BT, mientras que el WDT es deshabilitado. Enseguida las variables que son utilizadas para las mediciones son cargadas a la memoria *Flash*, en donde se van a almacenar los datos obtenidos de las mediciones en las localidades de memoria correspondientes a dichas variables. Además, se habilitan las interrupciones y se activa el modo de operación de bajo consumo de potencia LPM3. Enseguida se ejecuta un ciclo repetitivo en donde se espera la activación de alguna de las interrupciones para que estas realicen sus actividades correspondientes.

Este programa funciona de forma automática, es decir, cada vez que transcurre un intervalo tiempo determinado por el usuario se realiza cada una de las mediciones en donde los valores son retenidos hasta que se cumple con otro tiempo establecido para que estos valores sean almacenados en las localidades de memoria correspondientes. Cuando se vuelve a cumplir con el tiempo para la realización de las mediciones, estas se realizan y los nuevos datos son almacenados en las siguientes localidades de memoria correspondientes, lo cual se logra a través de apuntador que se incrementa para apuntar a la localidad de memoria adecuada. Cuando los bloques de la memoria *Flash* (A y B) se llenan, el

apuntador vuelve a iniciar en la localidad de memoria original siguiendo el mismo procedimiento que siguió con las primeras mediciones.

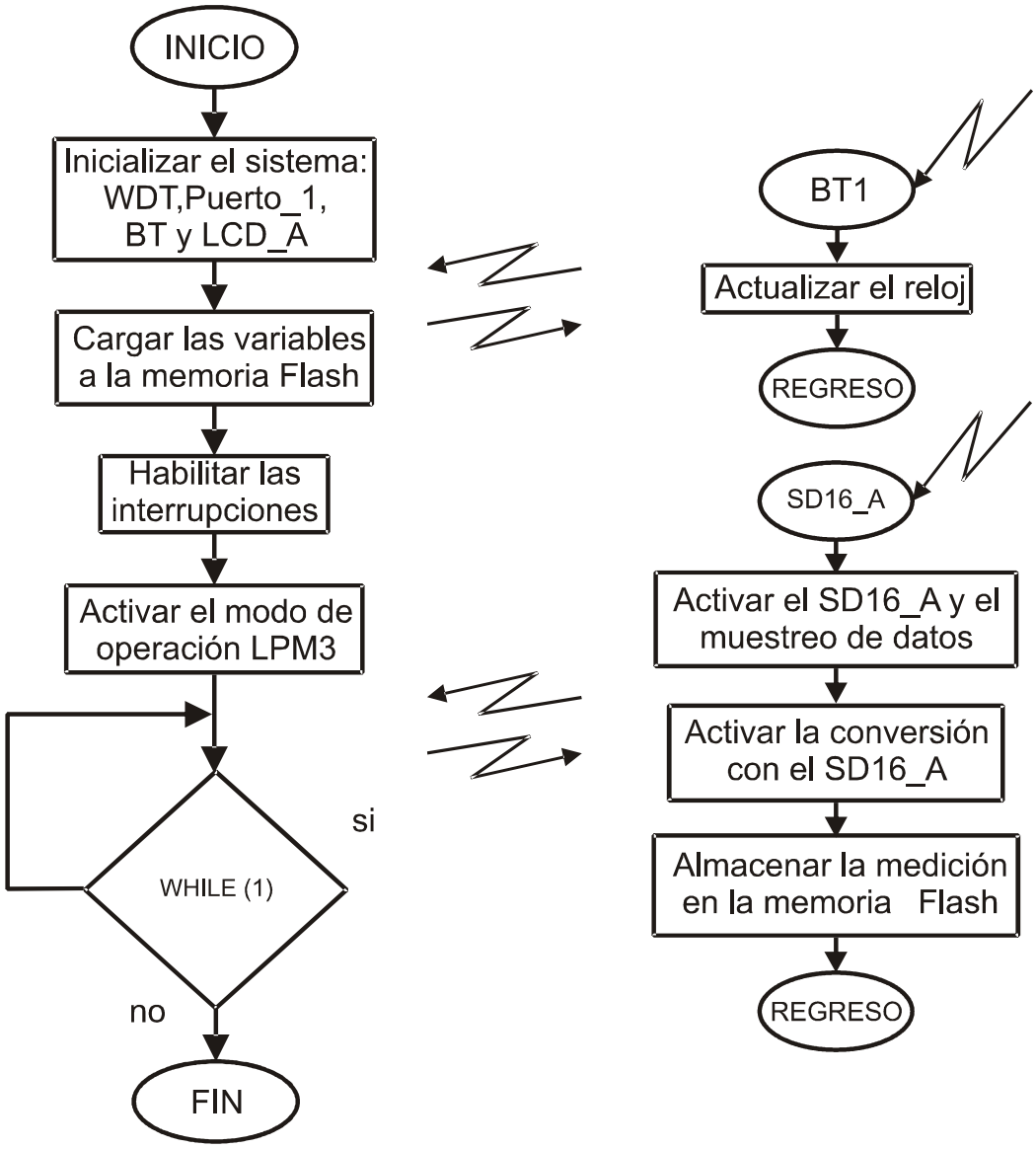


Figura 4.16 Diagrama de flujo del programa principal en C basado en intervalos de tiempos.

A continuación se presenta una tabla donde se especifican los tiempos en los cuales se realizan las diferentes mediciones.

Tabla 4.1 Ordenamiento de las actividades asociadas a las mediciones.

Tiempo	**Actividad**
Tiempo t	Medición Temperatura del microcontrolador
Tiempo t + 1 segundo	Medición Temperatura ambiente
Tiempo t + 2 segundos	Medición Presión atmosférica
Tiempo t + 3 segundos	Medición de la Altitud
Tiempo t + 4 segundos	Almacenamiento de las variables y el dato de la hora en la memoria *Flash*

Este programa realiza las actividades presentadas en la Tabla 4.1 con una frecuencia de un minuto pero esta puede ser modificada de acuerdo a las necesidades del usuario, por ejemplo, se puede cambiar para una frecuencia de cada hora, cada día, etc. Como se puede observar en esta tabla, las mediciones son realizadas en los primeros cuatro segundos de cada minuto y en el quinto segundo se realiza el almacenamiento de las variables en la memoria *Flash*, lo cual se repite permanentemente mientras el microcontrolador se encuentre en operación.

4.2.3 Interrupciones del Medidor

Los programas desarrollados en este proyecto funcionan por medio de cuatro rutinas atendidas por el servicio de interrupción, las cuales son la BT, WDT, Puerto_1 y la del convertidor A/D SD16_A.

4.2.3.1 Temporizador Básico (BT)

En la Figura 4.17 se presenta el diagrama de flujo de la interrupción del temporizador básico. Esta rutina es ejecutada cada segundo lo que provoca que el dato del tiempo sea actualizado cada segundo. Para realizar un análisis más a detalle de este tipo de interrupción se tiene que observar el comportamiento de ésta durante la ejecución de la función principal *main()*. Tan pronto como la bandera SF_BT_TIEMPO se detecte como que está activa, las variables del contador de reloj de tiempo real correspondientes a los

segundos son incrementadas en un valor de uno, para de esta manera hacer la representación de un reloj de tiempo real dentro de los programas. Todo esto es realizado dentro de la interrupción del temporizador básico.

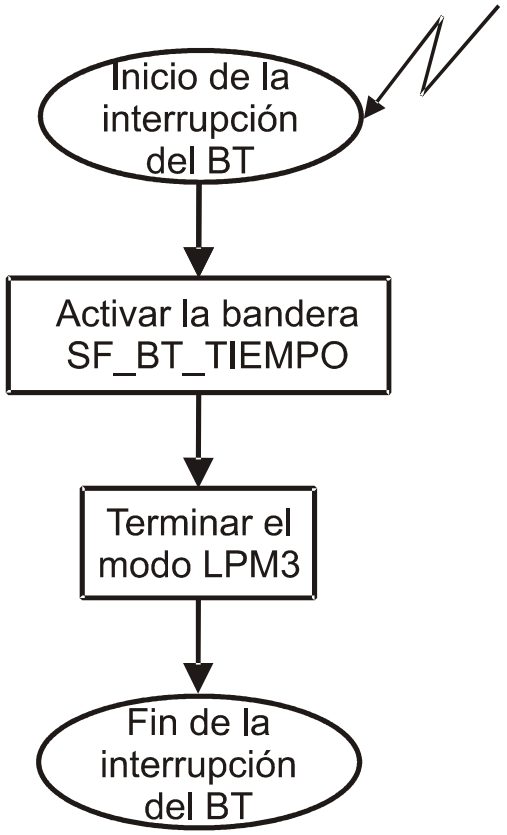


Figura 4.17 Diagrama de flujo de la interrupción del BT.

4.2.3.2 Puerto_1 y del WDT

Este proyecto cuenta con el uso de un botón conectado al puerto 1, el cual está encargado de realizar una actividad específica que consiste en la activación del proceso de medición de las variables meteorológicas de interés. Dentro del código de programación de la versión que utiliza el botón, se encuentra un evento llamado PUSH_BUTTON1, el cual es controlado por dos interrupciones que son el Puerto_1 y el WDT. En la Figura 4.18 se muestra diagrama de flujo de la interrupción del Puerto_1.

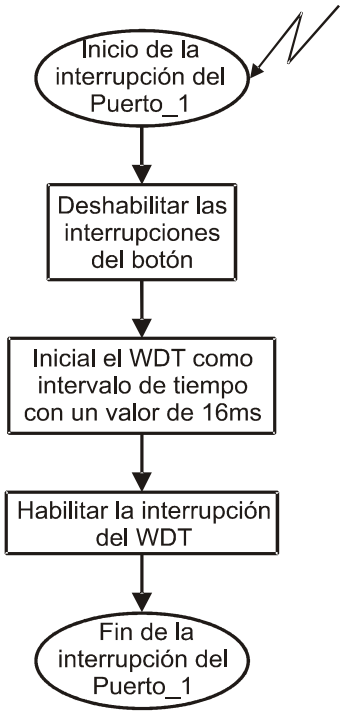


Figura 4.18 Diagrama de flujo de la interrupción del Puerto_1.

Como se puede observar en la Figura 4.18 la interrupción del Puerto_ 1 activa al WDT, haciéndolo funcionar como un temporizador con un valor de 16ms, para monitorear el estado del botón, por lo que si éste es presionado entonces se habilita la conversión de la variable actual. Además, ésta interrupción también activa el modo de operación de bajo consumo de potencia LPM3. La interpretación del estado del botón depende de un conjunto de banderas que indican si el botón está siendo presionado y si es así, se utiliza un contador para determinar por cuánto tiempo o bien si el botón ya fue liberado. Las banderas mencionadas y el contador el contador de tiempo utilizado se presentan en la Tabla 4.2 con su respectiva función.

Tabla 4.2 Banderas del Botón.

Bandera / Contador	**Función**
SF_BOTON_PRESIONADO	Bandera.- indica si el botón fue presionado
SF_BOTON_CERRADO	Bandera.-indica si actualmente el botón está siendo presionado
SF_BOTON_LIBERADO	Bandera.-indica si el botón fue liberado
cont_boton_presionado	Contador.- detecta el tiempo que el botón fue presionado

Además, la interrupción del WDT es habilitada cada vez que el botón es presionado debido a que el WDT activa el modo de operación de bajo consumo de potencia LPM3. Cada vez que las banderas de estado del botón son activadas, los cambios que presentan estas banderas a lo largo de la ejecución del programa son procesados dentro del evento principal del código *main()*. La Figura 4.19 muestra el diagrama de flujo de la interrupción del WDT.

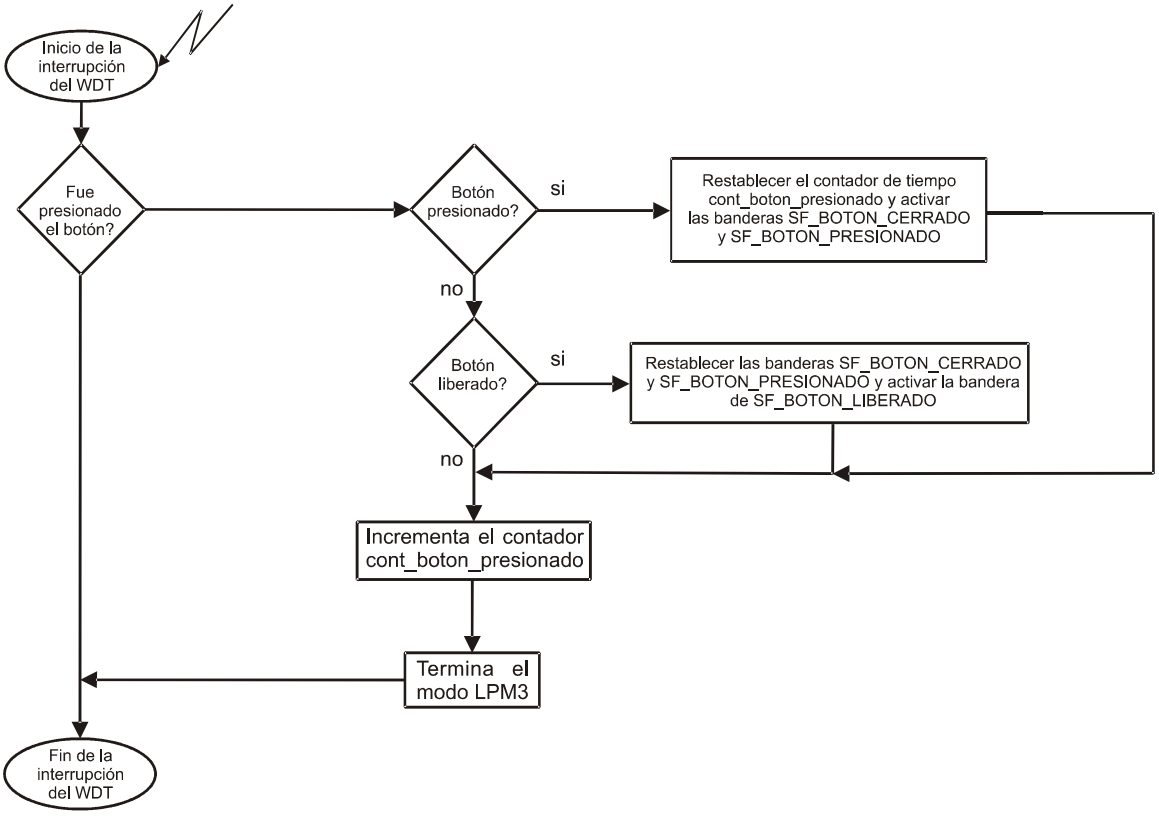


Figura 4.19 Diagrama de la interrupción del WDT.

En la Figura 4.19 se muestra claramente el procedimiento que sigue la interrupción del WDT. Esta interrupción es ejecutada cuando el botón es presionado, por lo que las banderas que indican si el botón está o ha sido presionado adquieren un valor. Ocurre lo mismo cuando el botón no ha sido presionado pero además se activa la bandera que indica que el botón ha sido liberado. Esto se realiza para asegurar que las banderas se encuentren en el estado deseado, porque si alguna de las banderas no es restablecida o activada cuando es requerido, entonces las interrupciones generadas pueden causar problemas durate la ejecución del programa.

4.2.3.3 Convertidor A/D SD16_A

Otro tipo de interrupción utilizada en este proyecto es la generada por el convertidor SD16_A, la cual se muestra en la Figura 4.20. En el caso de que se desee la medición de alguna de las variables estudiadas en este proyecto se involucra el uso del módulo SD16_A. Para realizar esto es necesario hacer un llamado a la función *IniciaConversion ()*, en donde son activados tanto el convertidor de datos A/D, la bandera SF_SD16_INICIO y el modo de conversión continua del SD16_A. Como se puede observar en la Figura 4.20, se consulta si ya se tiene el número de muestras establecido por el usuario y entonces se deshabilita el convertidor A/D y se activa la bandera de SF_SD16_LISTO. Esta bandera indica que ya se tiene el número de conversiones requerido para la medición y con esto se activa el modo de operación de bajo consumo de potencia. De lo contrario se siguen capturando muestras hasta que el número de éstas sea el indicado. El resultado de esta conversión puede ser leído en la variable SD16Temp la cual almacena la sumatoria de las conversiones. Una vez que se termina la interrupción y se transfiere el control del programa a la función *main()*, la bandera SF_SD16_LISTO que indica que se realizó la conversión y habilita que a la variable SD16Result se le aplique un corrimiento de *bits* para realizar el promedio de las muestra tomadas. Una vez que se realiza una conversión se continúan realizando estas continúan realizándose por medio del llamado a la función *IniciaSigConversion()*, esto solamente ocurre en el programa que utiliza el botón. Esto se puede observar cuando se está realizando una medición que es visualizada en pantalla LCD, ya que se pueden observar los cambios que sufre la variable medida cada vez que transcurra un segundo, debido a que las mediciones se calculan en ese tiempo.

El convertidor es configurado en este trabajo para utilizar tres de sus ocho entradas disponibles, es decir, las entradas de canal A0, A1 y A6, las cuales fueron programadas para que la señal de entrada sea amplificada con un ganancia de 32 (para el canal A0) y una ganancia unitaria (para los canales A1 y A6) por medio del módulo PGA (Ganancia de Amplificación Programable). El canal A0 corresponde a la medición de la presión atmosférica, el A1 corresponde a la medición de la temperatura ambiente y el canal A6 corresponde a la medición de la temperatura del microcontrolador. El convertidor requiere del uso del reloj ACLK (reloj auxiliar) con una frecuencia $f_{MOD} = 32.768kHz$ para su operación. Para la lectura de las mediciones en este proyecto se propone capturar 32

muestras de la presión atmosférica y la temperatura ambiente en la versión del código controlado por un botón y 16 muestras para versión controlada por intervalos de tiempo. La diferencia del número de muestras se debe a que en la versión controlada por tiempos es necesario que la medición se obtenga más rápido para que los resultados sean almacenados con un valor correcto debido a que mediante pruebas realizadas al momento de almacenar los resultados estos eran erroneos al momento de almacenarlos en la memoria *Flash* debido a que el tiempo de campura era muy grande. Las muestras son promediadas, mediante un corrimiento de *bits*. Esto permite reducir las pequeñas variaciones en los valores medidos.

Para determinar el tiempo que se invierte en cada una de las mediciones realizadas tenemos que utilizar la frecuencia que se le configura al convertidor OSR=1024 en conjunto con la frecuencia del reloj de cristal y las muestras tomadas. La siguiente relación permite calcular el tiempo que se requiere para que el convertidor realice las operaciones para cada señal de entrada:

$$tiempo = \frac{OSR*MUESTRAS}{f_{MOD}} = \frac{1024*32}{32.768kHz} = 1\ segundo \tag{4.1}$$

$$tiempo = \frac{OSR*MUESTRAS}{f_{MOD}} = \frac{1024*16}{32.768kHz} = 0.5\ segundos \tag{4.2}$$

Como se puede observar los tiempos requeridos para la obtención de una medición es de 1 y 0.5 segundos dependiendo del número de muestras que se tomen.

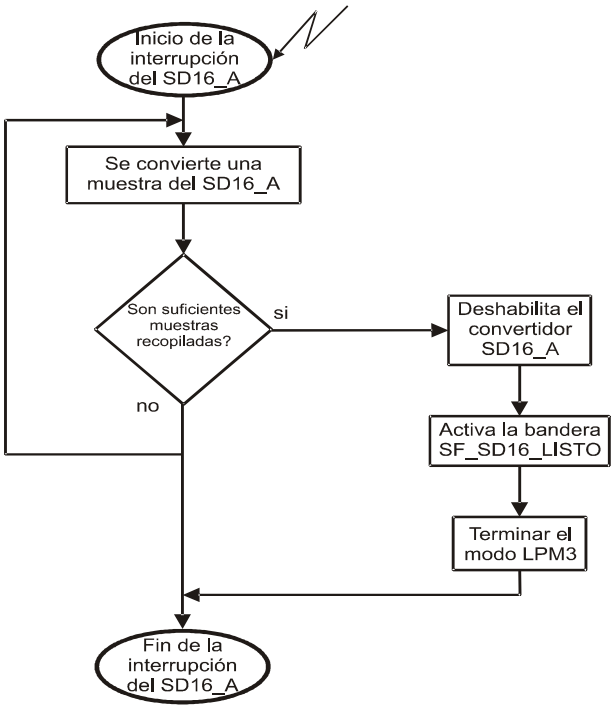


Figura 4.20 Diagrama de flujo de ISR del SD16_A.

4.3 Funciones y Módulos del Programa del Microcontrolador

Para la programación del microcontrolador MSP430F4270 es necesario configurar los módulos principales para que estos realicen las actividades requeridas en este proyecto. Estos módulos son los que realizan las actividades principales del microcontrolador dentro de las dos versiones de código desarrollados, los cuales son: el WDT, LCD_A, FCTLx y el SD16_A. A continuación se presenta la explicación del funcionamiento y la configuración de los módulos mencionados así como sus respectivas líneas de código por medio de las cuales son configurados.

4.3.1 Temporizador Perro Guardián (Watch Dog Timer WDT)

La función principal del módulo WDT es desarrollar un sistema de restablecimiento controlado en caso de que ocurran problemas de *software*. En caso de que el WDT no sea utilizado dentro de algún proyecto como un sistema de restablecimiento, este puede ser configurado como un temporizador o bien como un generador de interrupciones como fue en el caso de este proyecto.

La configuración de los registros del módulo WDT se presenta en la Figura 4.21, los cuales por defecto se encuentran desactivados (todos son igual a cero).

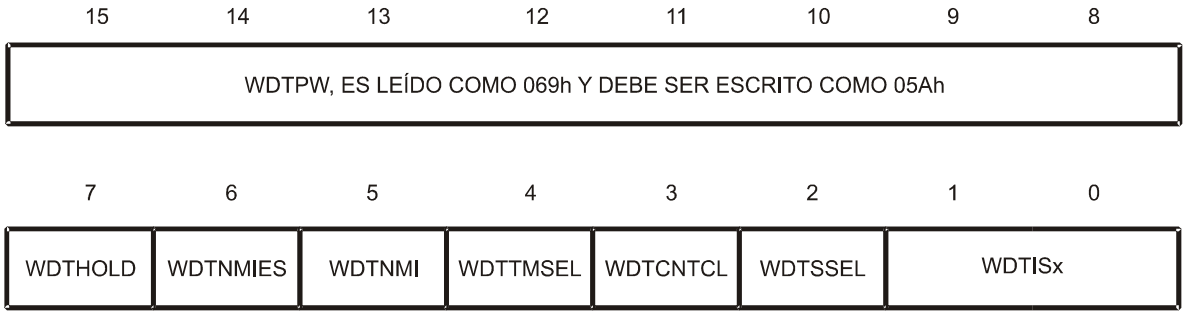


Figura 4.21 Configuración del módulo WDT.

El WDT es configurado dentro de la función *IniSis()* (inicializar sistema), en donde este es detenido, hasta el momento que sea llamado para realizar una interrupción. A continuación se presenta la línea de código que configura el WDT para que sea detenido.

```
WDTCTL = WDTPW + WDTHOLD;                    // Detener WDT
```

en donde:

WDTCTL Registro de control del WDT.

WDTPW Contraseña del WDT.

WDTHOLD Bloqueo del WDT. Este *bit* es el que detiene el WDT, por lo que cuando este *bit* está activado no se consume potencia.

- Si WDTHOLD = 0, el WDT está activado.
- Si WDTHOLD = 1, el WDT está desactivado.

Como ya se ha mencionado el WDT es utilizado como un generador de interrupciones por lo que a continuación se presenta la Figura 4.22 en donde ilustra la configuración de los registros necesarios para la habilitación y deshabilitación de las interrupciones.

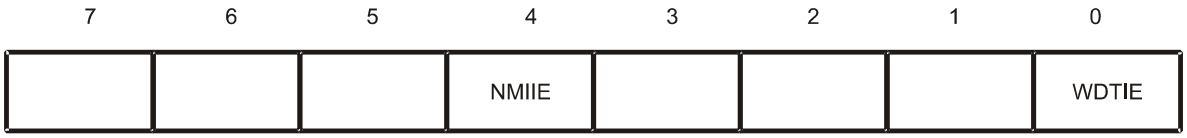


Figura 4.22 Registro de habilitación de interrupciones.

Este registro se encuentra configurado en el programa dentro del vector de interrupciones del WDT, la línea de código para la configuración de este registro es la siguiente:

IE1 |= WDTIE; // Habilitar la interrupción del WDT

en donde:

IE1 Registro de habilitación de interrupciones.

WDTIE Habilitar la interrupción del WDT.

- Si WDTIE = 0, la interrupción es deshabilitada.
- Si WDTIE = 1, se habilita la interrupción.

Esta es la manera como fue configurado el WDT dentro de los códigos desarrollados en esta tesis para ser utilizado como generador de interrupciones.

4.3.2 Controlador de la Pantalla LCD_A

El microcontrolador MSP430F4270 es un dispositivo que tiene integrado un controlador para una pantalla LCD, el cual es llamado LCD_A. Este controlador de pantalla genera las señales requeridas para el control de un LCD. Una de las características principales del LCD_A es que cuenta con una memoria de información la cual se encarga de retener los datos de las mediciones y que cuando se activa la bandera del LCD_A éste despliega los resultados en la pantalla. Otra característica del controlador LCD_A es que proporciona una alimentación de voltaje para el LCD independientemente de la fuente de voltaje. Además, este suministro de voltaje es regulado por medio de *software*. Por otro lado este módulo cuenta con la opción de poder operar en el modo de 2mux, 3mux y 4mux, dependiendo del tipo de pantalla LCD que se esté utilizando. En este proyecto se selecciona el modo de 4mux debido a que el LCD utilizado cuenta con cuatro multiplexores para el control de sus segmentos. La configuración de este módulo se realiza por medio de *software* exclusivamente en la versión del programa que utiliza un botón de control de las mediciones. Esta acción es realizada dentro de la función *IniSis()*.

La Figura 4.23 muestra el registro de control del LCD_A los cuales se encuentran, por defecto, desactivados.

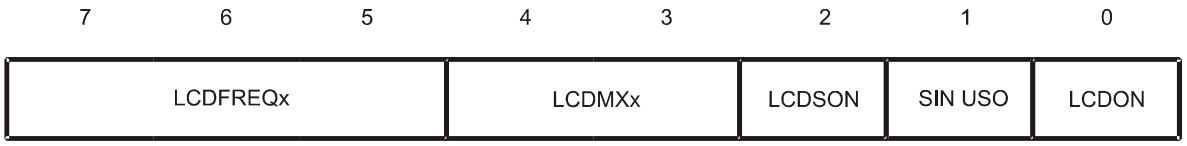


Figura 4.23 Registro de control del LCD_A.

La configuración del módulo utilizada en ese proyecto de presenta en la siguiente línea de código, la cual se encuentra en la función que inicializa el programa en lenguaje C.

LCDACTL = LCDFREQ_96 + LCD4MUX + LCDON; // Modo 4mux, selección de frecuencia

en donde:

LCDACTL Registro de control del LCD_A.

LCDFREQ_96 Selección de la frecuencia de operación del LCD. Este *bit* selecciona el divisor frecuencia ACLK para el LCD adhiriendo el número decimal correspondiente al tipo de división de frecuencia, en este caso este es de 96.

- LCDFREQx = 000, divide la frecuencia en 32.
- LCDFREQx = 001, divide la frecuencia en 64.
- LCDFREQx = 010, divide la frecuencia en 96.
- LCDFREQx = 011, divide la frecuencia en 128.
- LCDFREQx = 100, divide la frecuencia en 192.
- LCDFREQx = 101, divide la frecuencia en 256.
- LCDFREQx = 110, divide la frecuencia en 384.
- LCDFREQx = 111, divide la frecuencia en 512.

LCD4MUX Valoración del multiplexor. Estos *bits* seleccionan el modo del LCD.

- LCDMXx = 00, modo estático.
- LCDMXx = 01, modo de dos multiplexores.
- LCDMXx = 10, modo de tres multiplexores.
- LCDMXx = 11, modo para cuatro multiplexores.

LCDON LDC encendido. Este *bit* se encarga de encender o apagar el módulo LCD_A.

- LCDON = 0, LCD_A apagado.
- LCDON = 1, LCD_A encendido.

Como se puede observar la configuración de los registros de control de este módulo realizan diferentes actividades como la selección de la frecuencia del LCD, la selección del modo de operación dependiendo del tipo de LCD a utilizar y el control de encendido y apagado del módulo.

Otra configuración que es necesaria en este módulo controlador del LCD_A es la configuración de puerto control 0, en el cual se habilitan las terminales del microcontrolador como salidas de datos para conectar la pantalla LCD. Dicha

configuración se muestra en la Figura 4.24 en donde se presentan las terminales correspondientes al LCD.

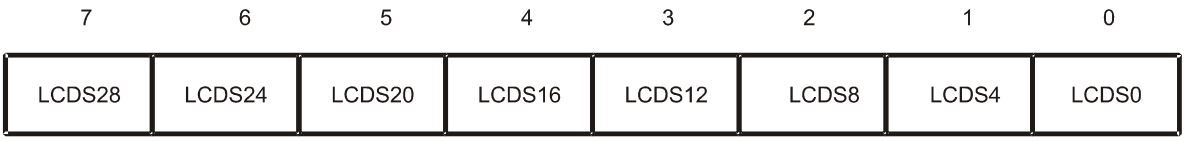


Figura 4.24 Registro de control del puerto 0 del LCD_A.

La configuración de este registro se realiza dentro de la función IniSis(), una vez que se ha configurado el registro de control del LCD_A. A continuación se muestra la línea de código que realiza esta configuración.

LCDAPCTL0 = 0x0F; **// Segmentos S0-S15 = salidas**

en donde:

LCDAPCTL0 Control del puerto 0 del LCD_A.

Como se puede observar en la línea de código simplemente de escribe un número en hexadecimal el cual indica que todos los *bits* de este registro deben de tener un valor de 1, lo cual hace que estos queden configurados como salidas. Es decir, las terminales del microcontrolador correspondientes a este módulo quedan configurados como terminales de funciones del LCD, los cuales se encargan de transmitir los datos a desplegar en la pantalla. En caso de que los *bits* de este registro no sean activados estos quedan disponibles para ser configurados como funciones particulares del puerto.

Una de las características del LCD_A es que cuenta con un módulo de suministro de voltaje para el contraste del LCD que sea conectado, esto sirve para mejorar la calidad en el despliegue de los datos. Este módulo corresponde al registro LCDAVCTL0, el cual puede proporcionar diferentes voltajes, en el caso de este proyecto solamente se utiliza un voltaje fijo de alimentación del LCD utilizado. La Figura 4.25 muestra la configuración del registro LCDAVCTL0.

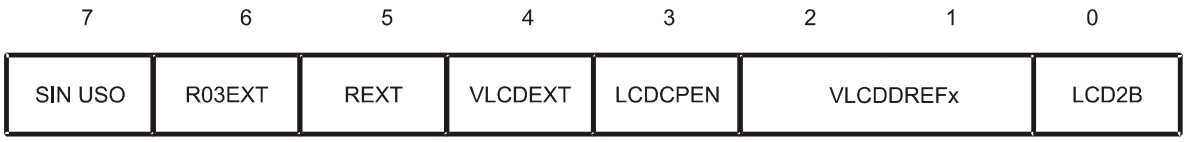


Figura 4.25 Registro LCDAVCTL0.

La configuración de este registro se realiza después de la configuración de las terminales del LCD, la siguiente línea de código ilustra la configuración de este registro, la cual se incluye dentro de la función *IniSis()*. Los *bits* de este registro se encentran desactivados por defecto.

LCDAVCTL0 = LCDCPEN; // Habilita el suministro de voltaje del LCD_A

en donde:

LCDAVCTL0 Registro de control de voltaje 0.

LCDCPEN habilitar el suministro de voltaje para el LCD.

- LCDCPEN = 0, el suministro de voltaje es deshabilitado.
- LCDCPEN = 1, el suministro de voltaje es habilitado.

4.3.3 Registros de Control de la Memoria Flash

Una segunda versión de los programas desarrollados en este proyecto se encarga de almacenar la mediciones obtenidas dentro de lo bloques A y B de la memoria *Flash*. Dentro del código son declaradas las variables a almacenar en la memoria mediante un llamado al bloque inicial de la memoria *Flash* (bloque B), por medio de la comando *#pragma*. Este comando realiza la conexión del código del programa con la memoria *Flash* del microcontrolador. Las variables a almacenar dentro de la memoria *Flash* tienen que ser declaradas mediante el comando que es reconocido por el microcontrolador *__no_init*. Esta palabra proporciona una dirección de memoria a la variable utilizada, con lo cual la variable puede ser utilizada dentro de la memoria *Flash*, donde los valores de ésta pueden ser actualizados o borrados.

En las siguientes líneas de código se ilustra el manejo de las variables de medición utilizadas en este proyecto así como la declaración de dichas variables para ser utilizadas en la memoria *Flash*.

```
#pragma dataseg = INFOB                 // Bloque B de la memoria Flash
__no_init static int ConstHrs_Flash;    // Variable asociada a la hora
__no_init static int ConstMins_Flash;   // Variable asociada a los minutos
__no_init static int ConstSecs_Flash;   // Variable asociada a los segundos
__no_init static int CalConstTM_Flash;  // Variable asociada a la temp.del micro
__no_init static int CalConstTA_Flash;  // Variable asociada a la temp. ambiente
__no_init static int CalConstP_Flash;   // Variable asociada a la presión
__no_init static int CalConstA_Flash;   // Variable asociada a la altitud
```

Con estas líneas de código se logra la creación de las variables que almacenan las mediciones realizadas a las diferentes variables meteorológicas. Ya que se cuenta con las variables relacionadas a la memoria *Flash* es necesario realizar la configuración de los registros de control de la memoria *Flash*. Esta configuración se realiza dentro de la función *AlmacenaFlash()*, la cual realiza el almacenamiento de las mediciones dentro de la memoria *Flash*. El primer registro que se configura dentro de esta función es el registro FCTL2, el cual se muestra en la Figura 4.26.

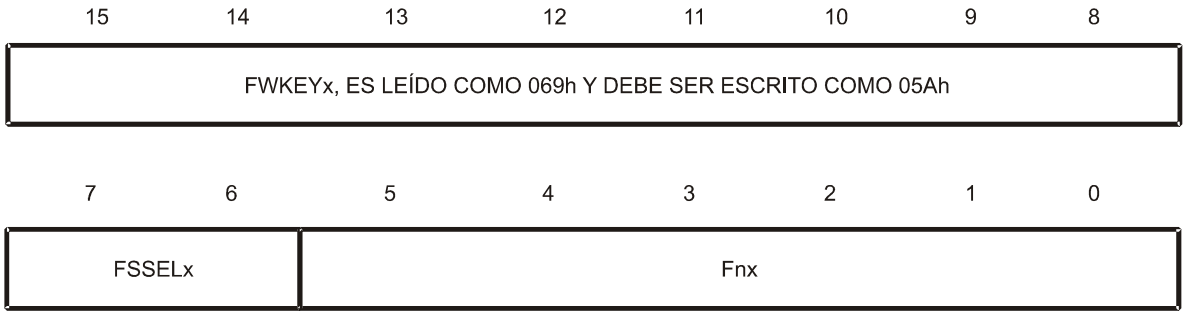


Figura 4.26 Estructura del registro FCTL2.

La configuración de este registro se realiza mediante la siguiente línea de código la cual tiene como funciones el ingreso de la contraseña para permitir manipular este registro, la elección de un reloj para el funcionamiento de este registro y la elección de un controlador *Flash* para el divisor del reloj:

FCTL2 = FWKEY + FSSEL1 + FN1; // SMCLK/3 = ~333kHz

en donde:

FCTL2 Registro de control de la memoria Flash (2).

FWKEY Contraseña para el acceso al FCTLx.

FSSEL1 Selección del reloj a utilizar.FSSEL1 = 00, selección del ACLK.

- FSSEL1 = 01, selección del MCLK.
- FSSEL1 = 10, selección del SMCLK.
- FSSEL1 = 11, selección del SMCLK.

FN1 Controlador Flash para el divisor del reloj.

El siguiente registro de control de la memoria *Flash* que es configurado dentro de la función *AlmacenaFlash()*, corresponde al registro FCTL3, en el cual simplemente desactiva el candado que no permite que los valores almacenados en la memoria *Flash* sean borrados, movidos de lugar o bien modificados. La desactivación del candado permite que los datos sean escritos en la memoria *Flash*. En la Figura 4.27 se muestra la estructura del registro FCTL3.

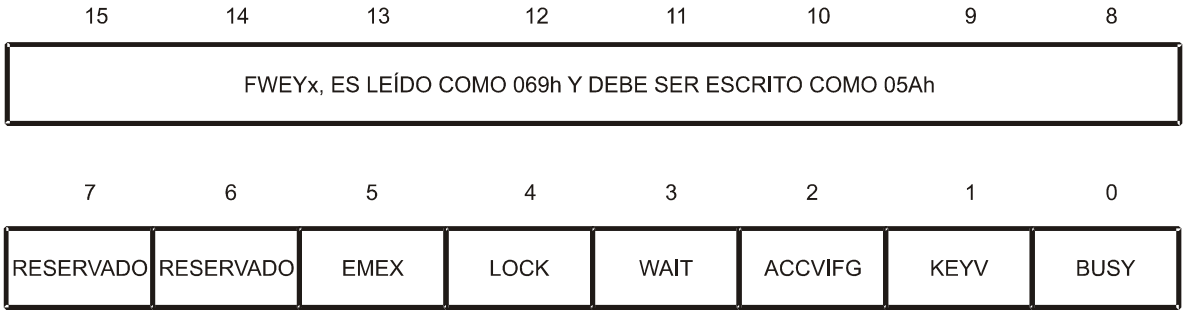


Figura 4.27 Estructura del registro FCTL3.

A continuación se muestra la línea de código que configura a este registro. Como se puede observar solamente se requiere de la activación de un *bit* debido a que lo único que nos interesa realizar con este registro es la desactivación del candado para poder escribir las mediciones obtenidas en la memoria *Flash*.

FCTL3 = FWKEY; **// Desactivación del candado (lock)**

en donde:

FCTL3 Registro de control de la memoria *Flash* (3).

FWKEY Contraseña para el acceso al FCTLx.

El registro que permite que la memoria flash sea escrita o borrada corresponde al FCTL1, el cual una vez que el candado ha sido deshabilitado éste se encarga de escribir los datos en la memoria *Flash*. La Figura 4.28 muestra la configuración del registro FCTL1.

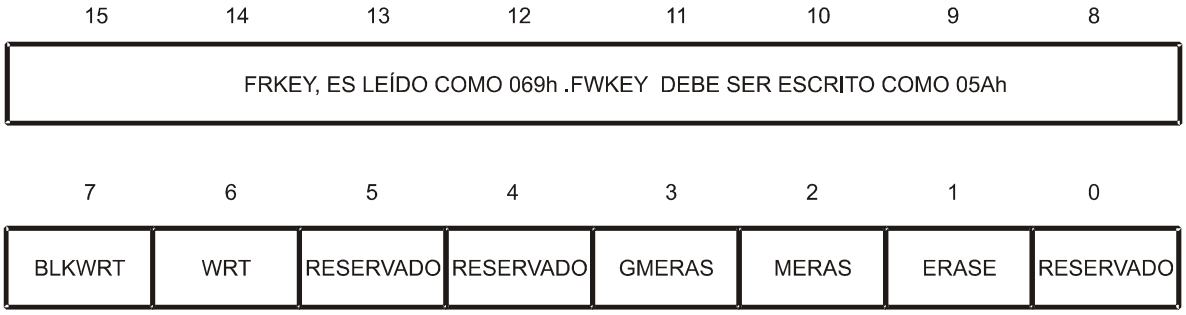


Figura 4.28 Estructura del registro FCTL1.

La siguiente línea de código ilustra la configuración del FCTL1 para que este habilite la escritura en la memoria *Flash*.

FCTL1 = FWKEY + WRT; **// Habilita la escritura**

en donde:

FCTL1 Registro del control de la memoria Flash(1).

FWKEY Contraseña para el acceso al FCTLx.

WRT Habilita la escritura en la memoria *Flash*.

Después de realizar esto, las mediciones pueden ser almacenadas dentro de la memoria *Flash*. Sin embargo, una vez que las variables ya fueron almacenadas es necesario deshabilitar el *bit* que permite la escritura en el FCTL1 con la siguiente instrucción.

FCTL1 = FWKEY; **// Deshabilita la escritura. Cero en el bit WRT**

Ahora bien lo más recomendable una vez que se han almacenado las mediciones en la memoria *Flash* es activar el candado (*bit* lock). Esto permite asegurar que la información que se encuentre en la memoria *Flash* no sea cambiada de dirección, modificada o borrada. Esto se realiza mediante la siguiente instrucción en el FCTL3:

FCTL3 = FWKEY + LOCK; **// Activa el candado. Uno en el bit Lock**

Estas son las instrucciones necesarias para el almacenamiento de información dentro de la memoria *Flash* a las cuales se le puede adherir el uso de apuntadores de memoria en caso de que se desee llenar los bloques de la memoria, como es el caso de una de las versiones desarrolladas en este proyecto. En este código se utilizó un apuntador de memoria el cual va tomando la localidad de memoria correspondiente a cada una de las variables a medir. El valor de la localidad de memoria es incrementado dependiendo del tipo de dato medido con lo que una variable se va almacenando en cada determinado número de localidades de memoria requerido.

4.3.4 Módulo del Convertidor A/D SD16_A

Para la configuración del módulo del convertidor A/D SD16_A es de suma importancia conocer el comportamiento de la señales de entrada para así poder determinar el valor de ganancia que estas requieren en el convertidor. La configuración de este módulo se lleva a cabo dentro de la función *IniciaConversión()*, en donde se configuran tres registros de este módulo los cuales son: SD16CTL, SD16INCTL0 y el SD16CCTL0.

En la Figura 4.29 se presenta la configuración de registro SD16CTL, el cual es configurado para seleccionar el reloj requerido para la operación del convertidor.

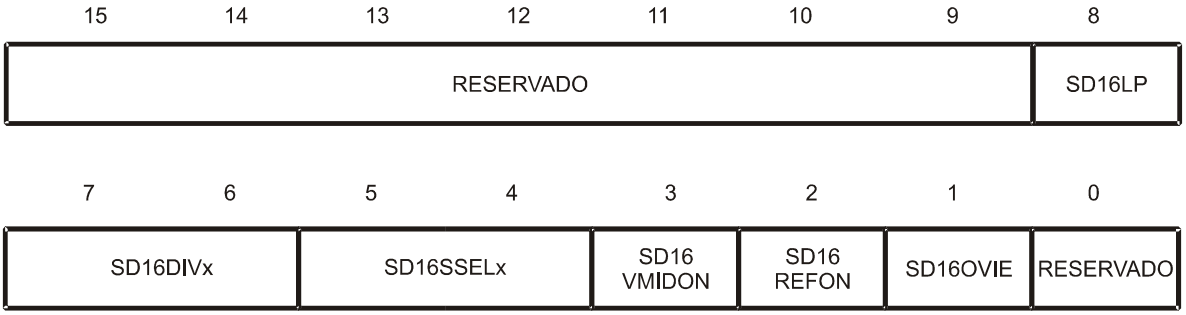


Figura 4.29 Estructura del registro SD16CTL.

Este registro es configurado mediante la siguiente instrucción:

SD16CTL = SD16SSEL_2; // Usar el ACLK.

en donde:

SD16CTL Registro de control del SD16.

SD16SSEL_2 Selección del reloj a utilizar, ACLK.

- SD16SSELx = 00, se selecciona el reloj MCLK.
- SD16SSELx = 01, se selecciona el reloj SMCLK.
- SD16SSELx = 10, se selecciona el reloj ACLK.
- SD16SSELx = 11, se selecciona el reloj externo TACLK.

El siguiente registro que se configura corresponde al SD16INCTL0 en cual se realiza la selección de la ganancia programable para la señal de entrada, así como la selección del canal de entrada. La Figura 4.30 presenta la estructura del registro SD16INCTL0.

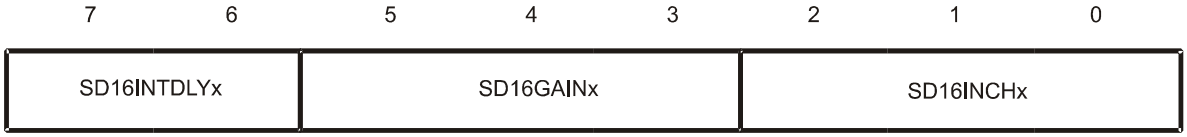


Figura 4.30 Estructura del registro SD16INCTL0.

A continuación se presenta la instrucción que configura este registro para que se seleccione la ganancia y el canal de entrada.

SD16INCTL0 = SD16GAIN_1 + SD16INCH_6; // Ganancia unitaria, canal A6

en donde:

SD16INCTL0 Registro de control de entrada del SD16_A.

SD16GAIN_1 Selección de una ganancia unitaria.

- SD16GAINx = 000, se amplifica en una vez la señal de entrada.
- SD16GAINx = 001, se amplifica dos veces la señal de entrada.
- SD16GAINx = 010, se amplifica cuatro veces la señal de entrada.
- SD16GAINx = 011, se amplifica ocho veces la señal de entrada.
- SD16GAINx = 100, se amplifica dieciséis veces la señal de entrada.
- SD16GAINx = 101, se amplifica treinta y dos veces la señal de entrada.
- SD16GAINx = 110, reservado.
- SD16GAINx = 111, reservado.

SD16INCH_6 Selección del canal A6 del multiplexor.

- SD16INCHx = 000, selección del canal A0.
- SD16INCHx = 001, selección del canal A1.
- SD16INCHx = 010, selección del canal A2.
- SD16INCHx = 011, selección del canal A3.
- SD16INCHx = 100, selección del canal A4.
- SD16INCHx = 101, selección del canal A5.
- SD16INCHx = 110, selección del canal A6 (sensor de temperatura interno).
- SD16INCHx = 111, selección del canal A7 (medición del voltaje de desviación).

El último registro configurado dentro de la función *IniciaConversión()* corresponde al SD16CCTL0, en el cual se configura la frecuencia que utilizada para la realización de las conversiones, el formato de salida de los datos y la habilitación de las interrupciones que genera este módulo. En la Figura 4.31 se presenta la estructura de este registro.

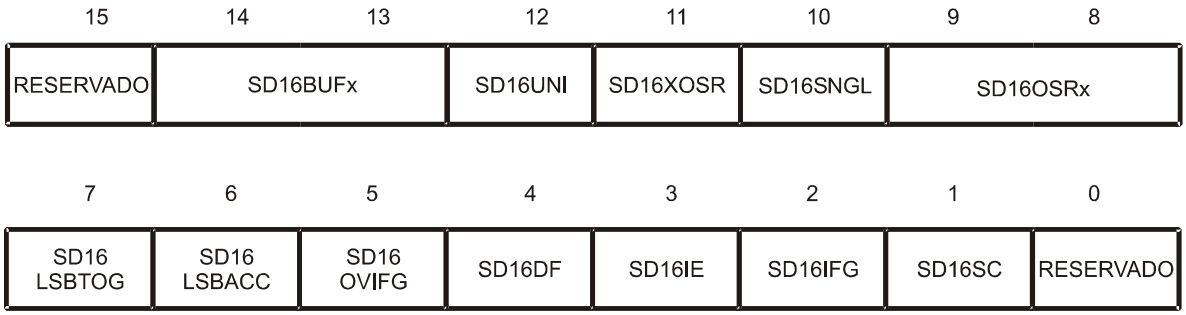


Figura 4.31 Estructura del registro SD16CCTL0.

La siguiente instrucción muestra como se realiza la configuración de este registro para realizar sus actividades correspondientes:

SD16CCTL0 = SD16OSR_1024 + SD16DF + SD16IE; // Conv. Cont, Complemento 2

en donde:

SD16CCTL0 Registro del canal de control 0.

SD16OSR_1024 Selección de una frecuencia de operación de 1024Hz.

SD16DF Selección del formato de salida de la información obtenida (bipolar con complemento a 2).

- SD16DF = 0 selección del formato de información, Offset Binario.
- SD16DF = 1 selección del formato de información, Bipolar Complemento a 2.

SD16IE Habilitar las interrupciones generadas por el SD16_A.

- SD16IE = 0, las interrupciones son deshabilitadas.
- SD16IE = 1, las interrupciones son habilitadas.

4.4 Conclusiones

Los programas desarrollados en este proyecto de tesis fueron realizados en lenguaje C. El primer programa realizado consiste en la utilización de un botón conectado al puerto 1 del microcontrolador el cual controla las mediciones a realizar, mostrando los resultados de estas en una pantalla de cristal líquido. Para el uso de esta pantalla es necesario configurar el controlador LCD_A integrado en el microcontrolador utilizado en

este proyecto. El segundo programa realizado se basa en el uso de intervalos de tiempos para realizar las mediciones de las variables en estudio y el almacenamiento de estas en la memoria *Flash* de microcontrolador. Para poder realizar este almacenamiento de datos se utiliza un puntero el cual va indicando la nueva localidad de memoria en la que se almacena el dato nuevo. Una vez que la memoria se llena esta se borra y se vuelve a empezar a almacenar datos desde la primer localidad de memoria. Para poder escribir y borrar datos en la memoria se tienen que habilitar los *bits* correspondientes antes de realizar cualquiera de estas dos acciones.

Las dos versiones de programa desarrolladas en este trabajo basan su funcionamiento en el uso de interrupciones. El uso de las interrupciones dentro de los códigos implementados en esta tesis, son de gran ayuda para el ahorro en el consumo de energía debido a que el microcontrolador se encuentra fuera de funcionamiento la mayor parte del tiempo.

Capítulo 5

Pruebas y Resultados

En este capítulo se presentan las pruebas realizadas tanto al *software* como al *hardware* del sistema. Las pruebas realizadas al *hardware* consisten en la realización de mediciones de las señales de salida de los sensores y la etapa de acondicionamiento, comparándolos con los cálculos requeridos para las mediciones. Por su parte, las pruebas realizadas al *software* consisten en el buen funcionamiento de las diferentes etapas que conforman el programa del medidor. Estas pruebas son visualizadas por medio de la ventana *watch* del *software Workbench*. Por último los resultados obtenidos con el medidor de presión y temperatura son comparados con una estación meteorológica comercial.

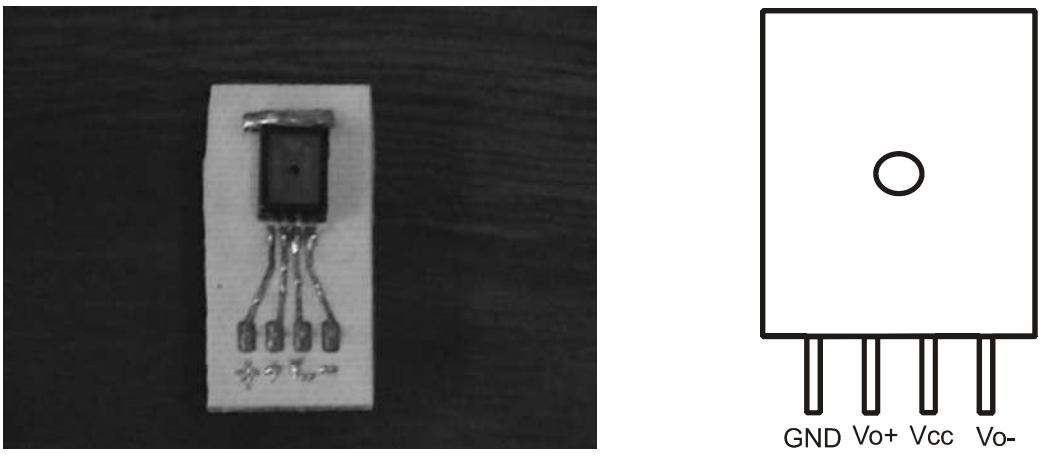
5.1 Pruebas de Hardware

Las pruebas realizadas al *hardware* consisten en diferentes etapas, las cuales son: pruebas al sensor de presión MPXM2102, al sensor de temperatura ambiente LM35 y a la etapa de acondicionamiento de la señal de este último. Estas pruebas consisten en verificar que los valores analógicos de las señales obtenidas por los transductores sean las adecuadas para ser introducidas al convertidor A/D. Una vez que se han adecuado las señales y los resultados obtenidos del convertidor A/D el siguiente paso consiste en la calibración de los resultados obtenidos con repecto a una estación meteorológica comercial (Vantage Pro2). La calibración de las mediciones se realiza internamente en el programa mediante el uso de variables de calibración.

5.1.1 Pruebas al Sensor de Presión MPXM2102

El sensor de presión utilizado en este proyecto es el MPXM2102 se muestra en la Figura 5.1, en donde se observa que el sensor esta montado en una baquelita, a la cual se le integraron una conjunto de terminales para poder ser montado en la tableta de trabajo. Este sensor cuenta con cuatro terminales de tipo superficial las cuales son conectadas directamente a la tira de terminales que se encuentran integradas a la baquelita. Dichas

terminales son: tierra (GND), terminal positiva del voltaje diferencial de salida (+), terminal de voltaje de alimentación (Vcc) y terminal negativa del voltaje diferencial de salida (-). El orden de estas terminales va de izquierda a derecha respectivamente como se muestra en la Figura 5.1. El sensor de presión cuenta con una terminal adicional a las cuatro mencionadas, la cual se encuentra ubicada en la parte superior de la Figura. Esta terminal no se encuentra conectada internamente a ningun elemento del sensor, solamente esta adherida al cuerpo del MPX2102 por lo que sirve para sujetar al sensor con la baquelita en la que fue montado.



a) b)

Figura 5.1 a) Sensor de Presión MPXM2102,b) Diagrama del Sensor de Presión (vista frontal).

Las pruebas realizadas al sensor de presión consistieron en la medición del voltaje diferencial de salida en las terminales del sensor para comparar éste con el voltaje calculado con el código del programa una vez que se realizó la conversión A/D. El voltaje medido en terminales presentó un valor de 9.7mV, como se muestra en la Figura 5.2.

Figura 5.2 Medición de voltaje en terminales del MPXM2102.

El voltaje calculado obtenido en el programa siempre presentó una magnitud menor al voltaje medido, el cual fue monitoreado por medio de la ventana *watch*. Por lo tanto, se agrega una constante de calibración que hace que el voltaje calculado se aproxime al valor medido. Para la elección de esta constante llamada *CAL_P* (dentro de las dos versiones de código desarrollados), fue necesario utilizar diferentes valores hasta encontrar el que hacía el resultado del cálculo se aproximara más al valor medido. Durante el desarrollo de esta prueba se observó que los valores cercanos a las 940 unidades, generaban un resultado muy cercano al medido por lo que se realizaron pruebas con un rango de valores de 940 a 950. Con cada uno de estos valores se realizaron diversas mediciones para apreciar las variaciones del voltaje calculado que se generaron con cada valor de la constante, por lo que se observó que con un valor de 943 unidades las variaciones eran mínimas como se muestra en la Tabla 5.1. Debido a estos resultados se decidió utilizar este valor para la calibración del voltaje de salida del sensor, el cual está involucrado en los cálculos tanto de presión atmosférica como de la altitud. En la Tabla 5.1 se presentan los resultados obtenidos para seis mediciones, en las cuales se observa voltaje calculado sin la constante de calibración y el voltaje calculado con la constante de calibración. Como se puede observar en esta tabla los valores de la variable con calibración presentan variaciones mínimas.

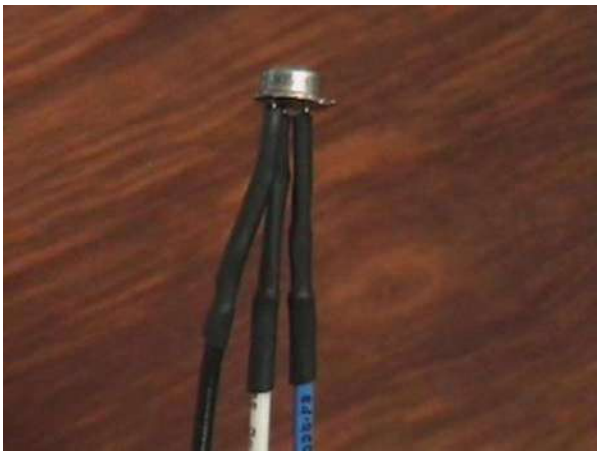
Tabla 5.1 Resultados de la mediciones con P_CAL = 943.

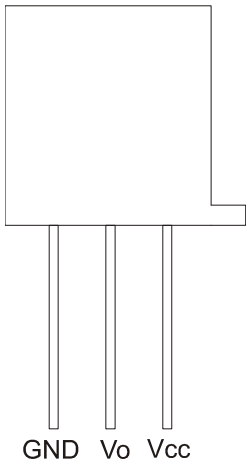
Número de medición	Medición sin calibración	Medición con Calibración
1	8.76354492E+3	9.7065449E+3
2	8.76012892E+3	9.7031289E+3
3	8.76149511E+3	9.7044951E+3
4	8.76764355E+3	9.7106435E+3
5	8.76764355E+3	9.7106435E+3
6	8.77242578E+3	9.7154257E+3

Como se puede observar en la Tabla 5.1, los resultados obtenidos presentan variaciones mínimas con la adición de un valor constante de 943 unidades, el cual aproxima el resultado al valor del voltaje medido de 9.7mV que se muestra en la Figura 5.2. Este valor representa un voltaje en µV internamente, debido a que estas son las unidades en las que se realizan las conversiones de la presión. El resultado obtenido en µV se convierte posteriormente a un valor de la presión en unidades de mB (milibares), los cuales a su vez se pueden convertir a mmHg (milímetros de mercurio).

5.1.2 Pruebas al Sensor de Temperatura LM35

El sensor de temperatura ambiente utilizado en este proyecto es el LM35 de encapsulado metálico el TO-46, el cual se muestra en la Figura 5.3. Este sensor cuenta con tres terminales: la terminal de la muesca (alambre azul) es la terminal de voltaje de alimentación, la terminal del centro (alambre blanco) es la terminal del voltaje de salida y la terminal del extremo izquierdo (alambre negro) es la correspondiente a la tierra. Además, este transductor es de salida no diferencial debido a que solamente cuenta con una terminal de salida.





a) b)

Figura 5.3 a) Sensor de Temperatura LM35, b) Diagrama del Sensor de Temperatura (vista frontal).

El voltaje de salida de este transductor de temperatura corresponde a 10mV/°C, el cual se puede presentar dentro de un rango de voltaje con valores desde los -0.55V hasta 1.5V, lo que en relación a la temperatura representa un rango de medición de -55°C a 150°C. Este transductor tiene características muy buenas debido a que cuenta con una calibración muy precisa en grados centígrados por lo que en este proyecto los resultados obtenidos al realizar mediciones de prueba fueron satisfactorios al momento de compararlos con las mediciones de la estación meteorológica utilizada para la validación de los resultados. Por lo tanto, para este sensor no es necesario utilizar un valor de calibración para el valor de la conversión de la señal analógica de salida proporcionada por este sensor.

Una de las pruebas realizadas a este sensor fue la realización de mediciones del voltaje de salida de la terminal no diferencial con respecto a tierra. Esta medición fue comparada con la medición de temperatura obtenida con el medidor desarrollado en este proyecto de tesis, lo cual se muestra en la Figura 5.4 donde se debe de mencionar que el voltaje medido esta dado en volts y la conversión en grados centígrados se obtiene con 10mV/°C.

Figura 5.4 Comparación de resultados obtenidos.

Como se puede observar en la Figura 5.4 los resultados obtenidos directamente de la conversión A/D son adecuadas, por lo que no es necesario el uso de un valor de calibración. Además, para poder asegurar que no es necesario un valor de calibración, se realizaron un número de mediciones para corroborar que la temperatura correspondiente al voltaje en terminales del sensor, es la misma prácticamente la misma temperatura medida por el medidor desarrollado en este proyecto. En la Tabla 5.2 se presentan los resultados obtenidos de estas mediciones.

Tabla 5.2 Resultados de las mediciones de la temperatura.

Número de medición	Medición de la temperatura en terminales del sensor (°C)	Medición de la temperatura con el medidor desarrollado (°C)
1	21.6	2.18278634E+1
2	21.6	2.19260371E+1
3	21.4	2.16062786E+1
4	21.6	2.20374062E+1
5	21.7	2.19552518E+1
6	21.6	2.17904826E+1

Como se puede observar los resultados de las mediciones son muy buenos. Los resultados de las mediciones realizadas con el medidor fueron monitoreadas en la ventana *watch* del *Workbench*.

Sin embargo debido a que los límites de voltaje de salida proporcionados por este transductor sobrepasan el rango de voltaje de entrada al convertidor A/D, es necesaria la implementación de una etapa de acondicionamiento para esta señal, la cual hace que el voltaje de salida del transductor se ajuste al rango de entrada del convertidor del microcontrolador.

Por otro lado, el sensor LM35 no puede proporcionar por sí mismo voltajes negativos correspondientes a temperaturas bajo cero, por lo que es necesario el uso de una fuente negativa para proporcionar estas mediciones.

5.1.3 Pruebas a la Etapa de Acondicionamiento

La etapa de acondicionamiento implementada en este trabajo asociada a la medición de la temperatura ajusta el rango de voltaje de salida del sensor de -0.55V a 1.5V al rango de voltaje de entrada del convertidor el cual corresponde a ±0.6345V. Sin embargo, en este trabajo el rango de voltaje del convertidor se programa para un margen de 0 a 0.6345V. Para el cálculo de la temperatura con el LM35 se utiliza la Ecuación 5.1, la cual describe el funcionamiento de la etapa de acoplamiento de la señal.

$$Vo = (0.309512) * Vi + 0.170232 \quad (5.1)$$

Esta ecuación representa la recta característica de transferencia obtenida mediante el acoplamiento de la señal acondicionada, como se muestra en la Figura 3.15.

La etapa de acondicionamiento implementada en este proyecto se muestra en la Figura 5.6, en la cual se utilizaron dos amplificadores operacionales LF356 que cuentan con entradas para la corrección del voltaje de desviación (offset), resistencias de valores correspondientes a los calculados mostrados en el Capítulo 4 y dos reguladores de voltaje implementados por diodos tipo zener. Estos reguladores incorporados para disminuir las variaciones de voltaje generadas por la carga y descarga del banco de baterías.

a)

b)

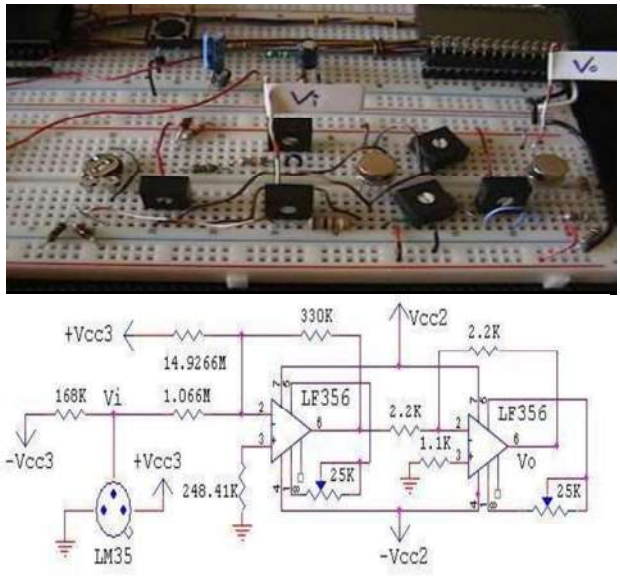


Figura 5.6 a) Etapa de Acondicionamiento, b) Diagrama de la Etapa de Acondicionamiento.

Las pruebas realizadas a la etapa de acondicionamiento consistieron en suministrar diferentes voltajes de entrada Vi y medir los voltajes en la salida Vo. Se destacan tres mediciones realizadas en los puntos que se muestran en la Figura 3.15, los cuales son P1, P2 y P3. En la Figura 5.7 se muestran los resultados de la pruebas al aplicar un voltaje de entrada en la etapa de acondicionamiento de un valor de Vi = -0.55V, obteniendo como resultado un voltaje de salida de Vo = -0.0094V. Esto se corrobora sustituyendo el valor de Vi en la Ecuación 5.1, donde se puede observar que el voltaje de salida es Vo= 0.

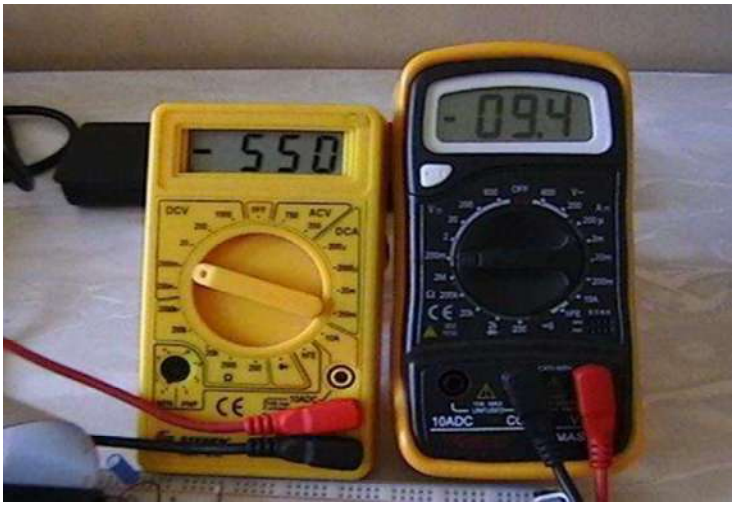


Figura 5.7 Resultados obtenidos con un voltaje Vi= -0.55V.

El siguiente voltaje de entrada aplicado a la etapa de acondicionamiento corresponde a un voltaje de Vi=1.5V el cual al ser sustituido en la Ecuación 5.1 da como resultado un Vo=0.6345V. Estos resultados se validan en la Figura 5.8 para la salida de la etapa de acondicionamiento, los cuales son prácticamente iguales.

Figura 5.8 Resultados obtenidos con un voltaje Vi=1.5V.

El último punto P3 descrito en la Figura 5.8 corresponde a un voltaje de entrada de Vi=0V, el cual da como resultado el valor de la ordenada de la recta. Al sustituir este valor de Vi en la Ecuación 5.1 se obtiene un valor de voltaje de salida Vo=0.170232V. En la Figura 5.9 se muestra el resultado obtenido en esta prueba con un voltaje de entrada de cero volts.

Figura 5.9 Resultados obtenidos con un voltaje Vi=0.

Como se puede observar los resultados obtenidos durante las pruebas de la etapa de acondicionamiento de la señal son satisfactorios debido a que los resultados medidos corresponden a los valores calculados. Además, con estos resultados demuestran que no es necesario agregar un valor de calibración para la medición de la temperatura ambiente debido a las buenas características de sensor LM35 y a los resultados obtenidos de la etapa de acondicionamiento.

5.1.4 Circuitería Utilizada

El *harware* utilizado en este proyecto se muestra en la Figura 5.10 en donde se muestra la tarjeta de desarrollo utilizada en este trabajo, en la cual se monta el microcontrolador. Esta tarjeta cuenta con un dispositivo que es la interface para conectar la tarjeta con la computadora a través del puero USB. También se muestran los sensores utilizados para las mediciones de presión y temperatura los cuales son montados en un tableta de trabajo en conjunto con la etapa de acondicionamiento, circuitos para regular los voltajes, el botón para la activación de mediciones, la pantalla LCD utilizada para el despliegue de mediciones en tiempo real y el banco de baterías para la alimentación del sistema.

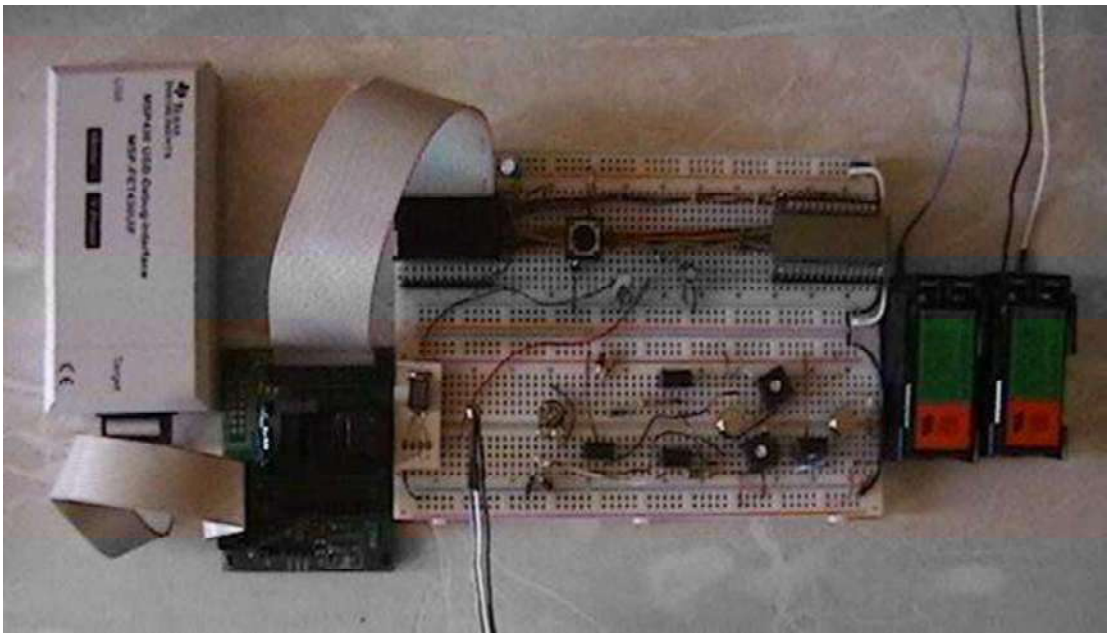


Figura 5.10 Circuitería desarrollada para el proyecto de tesis.

En la Figura 5.11 se muestra la tarjeta de trabajo utilizada para la programación del microcontrolador y para la realización de las pruebas mostradas en este capítulo. Esta tarjeta cuenta con un cristal integrado para generar la frecuencia necesaria para la operación del microcontrolador. El puerto JTAG realiza la conexión de la tarjeta con el dispositivo que sirve de interface para conectar a ésta con la computadora a través del puerto USB.

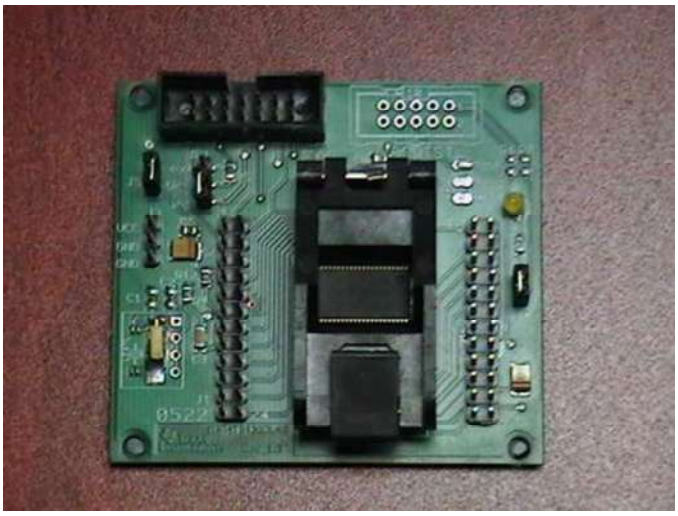


Figura 5.11 Tarjeta de desarrollo de *Texas Instruments* utilizada en este proyecto.

5.2 Pruebas de Software

Dentro de los códigos desarrollados para este proyecto se utilizan diferentes ecuaciones para realizar los cálculos de la temperatura ambiente en °C, la presión atmosférica en mmHg y la altitud en metros, las cuales de describen a continuación.

Cálculo de la Temperatura Ambiente.

El cálculo de la temperatura ambiente se realiza mediante la Ecuación 5.2, para la cual se utiliza la ecuación de la recta obtenida en la etapa de acondicionamiento que corresponde a la Ecuación 5.1. La ecuación de la temperatura consiste en la obtención del voltaje de entrada Vi, el cual corresponde al valor de la temperatura ambiente TA, una vez que se conoce el voltaje de salida Vo generado por la etapa de acondicionamiento de la señal. Por lo tanto, se despeja Vi de la Ecuación 5.1 para obtener la temperatura TA.

$$V_i = [V_o - V_b] * \frac{100}{m} \quad (5.2)$$

en donde:

$$V_o = SD16Result * \frac{V_{REF}}{2*BITS\_RESOL}$$

Sustituyendo Vi y Vo en la Ecuación (5.2) y considerando que Vi=TA, se tiene,

$$TA = \left[SD16Result * \frac{V_{REF}}{2*BITS\_RESOL} - V_b\right] * \frac{100}{m} \quad (5.3)$$

en donde:

TA	Temperatura Ambiente en °C.
SD16Result	Resultado obtenido del convertidor A/D.
Vref	Voltaje de referencia (1.269V).
BITS_RESOL	*Bits* de resolución (32767).
Vb	Ordenada de la Ecuación 5.1 (0.170232V).
m	La pendiente de la Ecuación 5.1 (0.309512 unidades).

Cálculo de la Presión Atmosférica.

El cálculo de la presión se realiza mediante la Ecuación 5.4, en donde se observa que el resultado del convertidor es multiplicado por 1000000, debido a que las unidades para esta medición son µV. Esto se realiza para no utilizar valores tan pequeños de voltaje pero respetando sus unidades. En este cálculo de la presión se utiliza una ganancia de 32, la cual es programada en la selección del canal del convertidor. Esta ganancia tiene un valor típico (proporcionado por el fabricante del microcontrolador) de 28.35 unidades. Para convertir el dato medido en mB, se utiliza la relación de 10mB/120µV, la cual se obtiene de la gráfica proporcionada por el fabricante [Manual_Texas_8 2005].

$$P = SD16Result * 1000000 \left[\frac{V_{ref}}{2*GAIN*BITS\_RESOL}\right] * [MB\_POR\_UV] \quad (5.4)$$

en donde:

SD16Result	Resultado del convertidor A/D.
Vref	Voltaje de referencia (1.269V).
GAIN	Ganancia (28.35).
BITS_RESOL	*Bits* de resolución (32767).
MB_POR_UV	mB por cada µV (10/120).

Cálculo de la Altitud

El cálculo de la altitud se realiza por medio de la Ecuación 5.5 en la que interviene diferentes factores como la gravedad, la constante del aire seco, la temperatura del medio ambiente, la presión atmosférica medida y la presión atmosférica de referencia [Manual_Texas_8 2005].

$$P = Po * e^{-\left(\frac{g*h}{R^*(TM)}\right)} \tag{5.5}$$

en donde:

P Presión atmosférica medida en mB.

Po Presión atmosférica de referencia de 1013.25mB.

R* Constante del aire seco $\left[\frac{287\frac{m^2}{s^2}}{^\circ K}\right]$.

g Gravedad $(-9.8\,\frac{m}{s^2})$

TM Temperatura ambiente en °K.

h Altitud en metros.

Despejando la altitud de la Ecuación 5.5, sustituyendo los valores de las constantes y tomando una TM= 25°C (la cual se convierte a °K), se obtiene la Ecuación 5.5, con la cual se calcula la altitud en metros.

$$h = (20085.2381) * log\left[\frac{Po}{P}\right] \tag{5.6}$$

La elección de la temperatura TM=25°C, es debido a que el fabricante del sensor de presión, proporciona las gráficas del voltaje de salida con respecto a la presión medida a una temperatura ambiente de 25°C.

Cálculo de la Presión Barométrica.

La presión barométrica es una presión que relaciona la presión atmosférica medida con la altitud del lugar en donde se esté realizando la medición. El cálculo de esta presión se desarrolla por medio de la Ecuación 5.7, en conjunto con la Ecuación 5.8 [*La Guia Metas* 2005].

$$P_B = P + P_o - P_s \tag{5.7}$$

$$P_s = P_o * (1 - (22.5569E^{-6}) * h)^{5.256} \tag{5.8}$$

en donde:

P Presión atmosférica local en mB.

P_B Presión barométrica en mB.

P_s Presión atmosférica en relación con la altitud local en mB.

Para la comparación de resultados obtenidos con los resultados de la estación meteorológica utilizada para la validación de las mediciones, la presión en mB se expresa en unidades de mmHg utilizando la Ecuación 5.9.

$$P_{mmHg} = \frac{P_B * 760_{mmHg}}{1013.25_{mB}} \tag{5.9}$$

5.2.1 Pruebas de las Mediciones con la Ventana Watch

En estas pruebas los cálculos realizados mediante el sistema de medición son visualizados en la ventana *watch* del programa *Workbench*. En la Figura 5.12 se muestran los resultados de las variables meteorológicas obtenidas con el sistema de medición. En esta figura se puede observar primeramente la información asociada a la hora, minuto y segundo en los cuales se realizó la medición. Enseguida se muestran las mediciones de la temperatura en el microcontrolador, la temperatura ambiental, presión barométrica y la altitud.

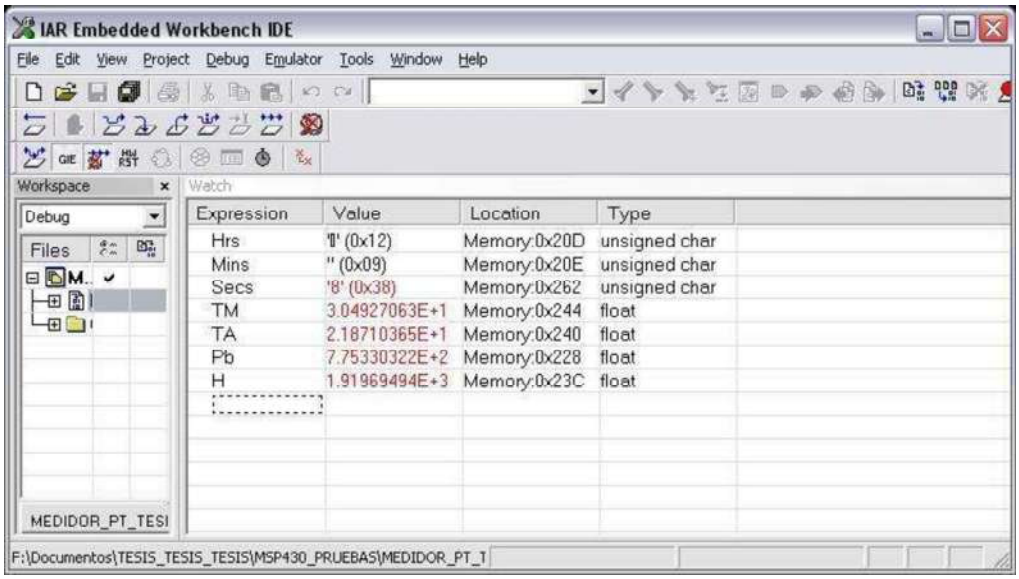


Figura 5.12 Prueba del software desarrollado.

5.3 Validación

Las mediciones realizadas con el medidor de presión y temperatura desarrollado en este proyecto fueron comparadas con la estación meteorológica comercial *Vantage Pro2*, la cual es un equipo que realiza la transmisión de las mediciones de variables atmosféricas de forma inalámbrica, por medio un módulo de radio frecuencia que operan a 900MHz. El alcance de transmisión de esta estación comercial alcanza los 150m, incluso con la presencia de muros que se obstaculicen entre el módulo de los sensores y el módulo de medición. Esta estación cuenta con diferentes sensores para las mediciones de variables meteorológicas como son, velocidad y dirección de viento, temperatura ambiente, humedad, precipitación pluvial, presión barométrica, radiación solar y sensor de radiación ultravioleta. La carátula de la estación meteorológica es energizada a través de baterías, lo cual la convierte en un medidor portátil. El precio de esta estación es aproximadamente de $20,000 (veinte mil pesos).

Además, en el lugar donde se realizaron las mediciones este módulo se encuentra conectado al internet para que las mediciones sean visualizadas por algún usuario. En este caso en particular esta mediciones son monitoreadas por el OOAPAS. Además esta estación meteorológica puede ser configurada para que proporcione las mediciones de las variables atmosféricas en diferentes unidades de medida. A continuación se presentan fotografías en donde se muestra la comparación de los resultados de ambos equipos.

Comparación de la medición de Presión

En la Figura 5.13 se muestra la comparación de las mediciones de la presión barométrica en mmHg. En la Figura 5.14 se muestra una ampliación de estas mediciones.

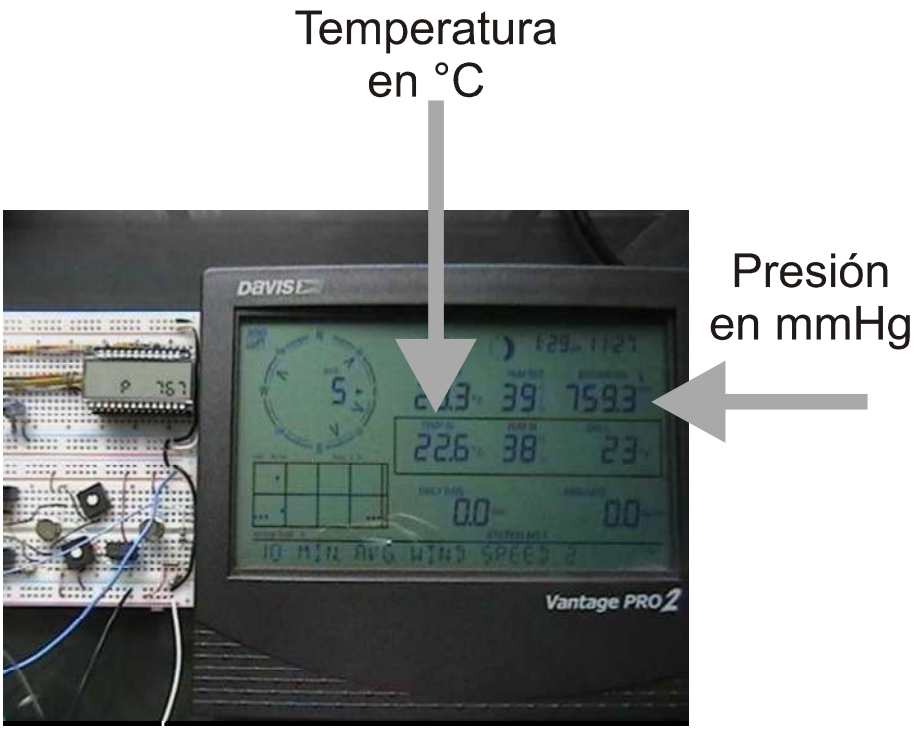


Figura 5.13 Comparación de la Presión Barométrica.

Figura 5.14 Ampliación de la mediciones.

Como se puede observar existe diferencia entre ambas mediciones por lo que la medición realizada con el proyecto realizado en esta tesis se calibró por medio de un valor constante mediante *software*, con lo que el valor medido se aproximo en gran similitud con el de la estación meteorológica.

En la Figura 5.15 se muestran los resultados una vez que el medidor de presión fue calibrado. En la Figura 5.16 se muestra una ampliación de los resultados.

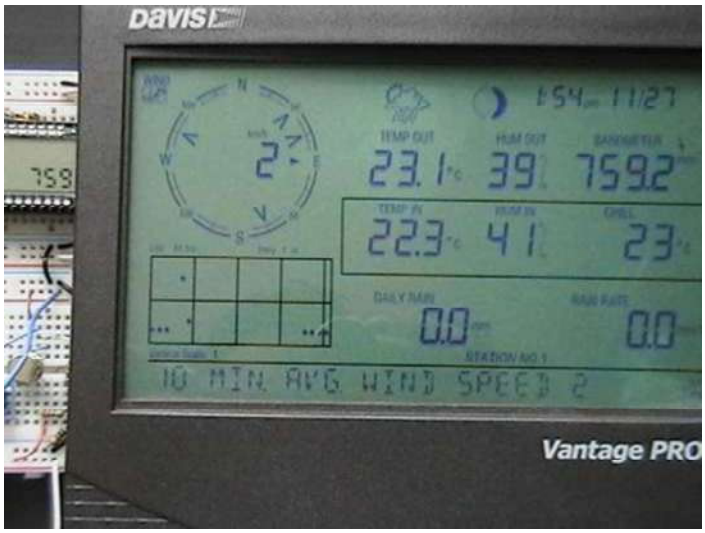


Figura 5.15 Comparación de la presión con el medidor calibrado.

Figura 5.16 Acercamiento a las mediciones de presión.

Como se puede observar las mediciones son muy similares una vez que se realizo la calibración de la presión obteniendo un porcentaje de error de 0.03951%.

Comparación de la medición de la Temperatura

En la Figura 5.17 se presenta la comparación de la medición de la temperatura entre ambos medidores. En la Figura 5.18 se muestra una ampliación de los resultados obtenidos.

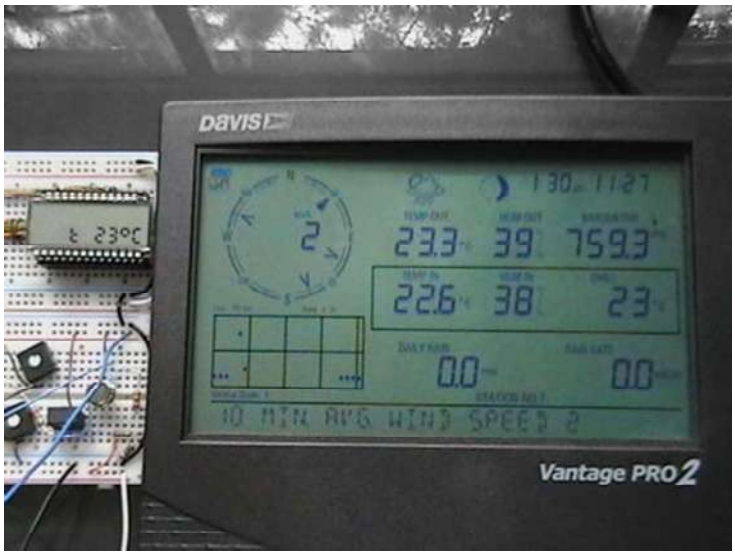


Figura 5.17 Comparación de las Temperatura Ambiente.

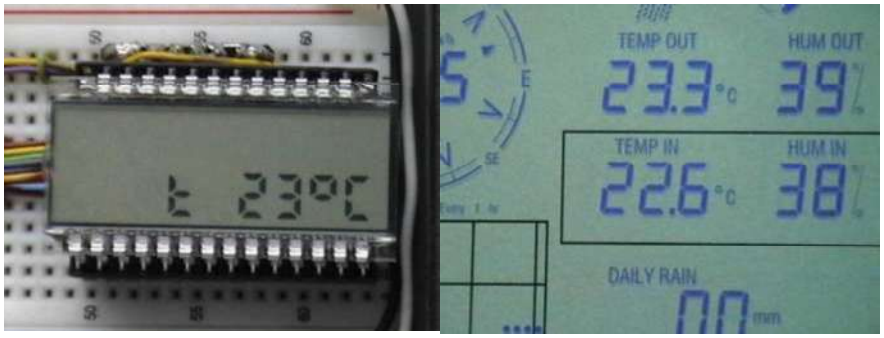


Figura 5.18 Acercamiento a las mediciones de temperatura.

Como se puede observar en el caso de la medición de la temperatura los resultados fueron satisfactorios debido a la similitud de ambas mediciones con un porcentaje de 1.76%. Por lo tanto, no fue necesario calibrar esta variable de medición.

En la Figura 5.19 se muestran los equipos de medición en el lugar donde fueron realizadas las pruebas con el uso de todas las herramientas necesarias para el funcionamiento de este proyecto como el circuito implementado en la tableta de trabajo, la tarjeta de desarrollo del microcontrolador junto interface que la conecta con la computadora, el banco de baterías para energizar el circuito y la computadora en la que se desarrollaron los códigos y las pruebas de estos.

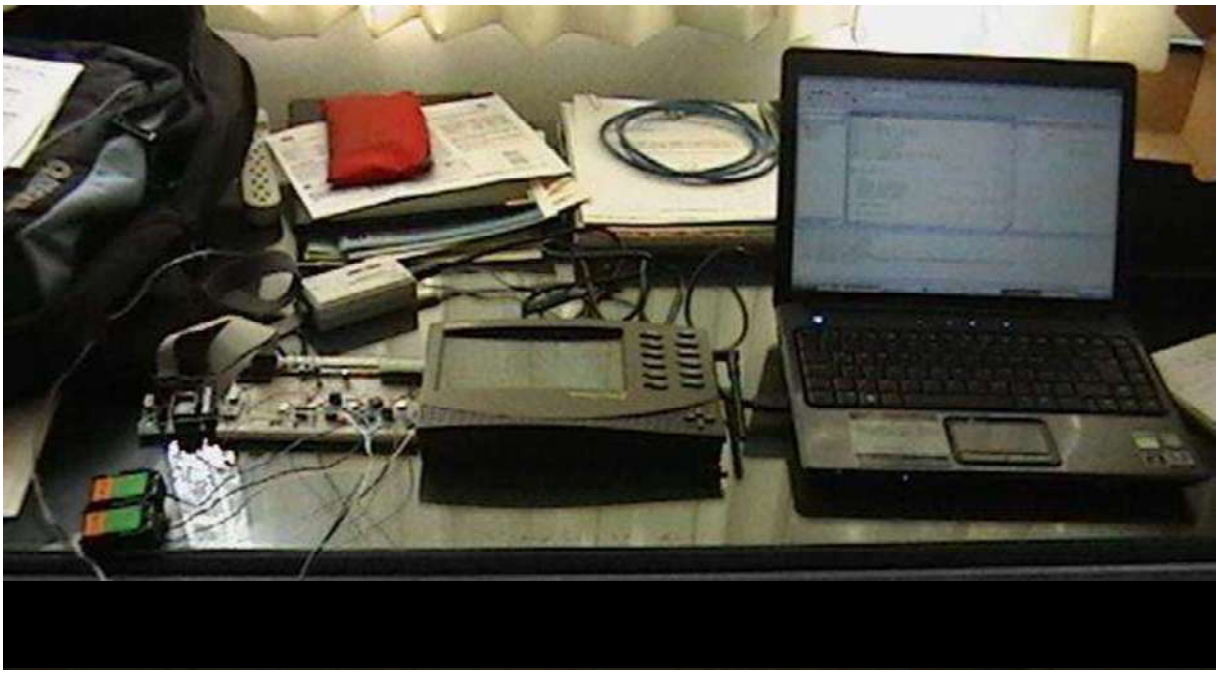


Figura 5.19 Equipos de medición desarrollado y estación comercial.

Otras pruebas realizadas al sensor de temperatura fueron la medición de temperaturas bajo cero las cuales consistieron en introducir el sensor a un congelador junto con un termostato que viene con un multímetro para visualizar la lectura de mediciones negativas. Esta prueba es para la validación de las mediciones negativas proporcionadas por el medidor desarrollado en comparación con un instrumento de medición comercial. En la Figura 5.20 se muestra la medición de temperaturas bajo, en la cual se presenta el resultado de dicha prueba, en donde se observa que los resultados son buenos.

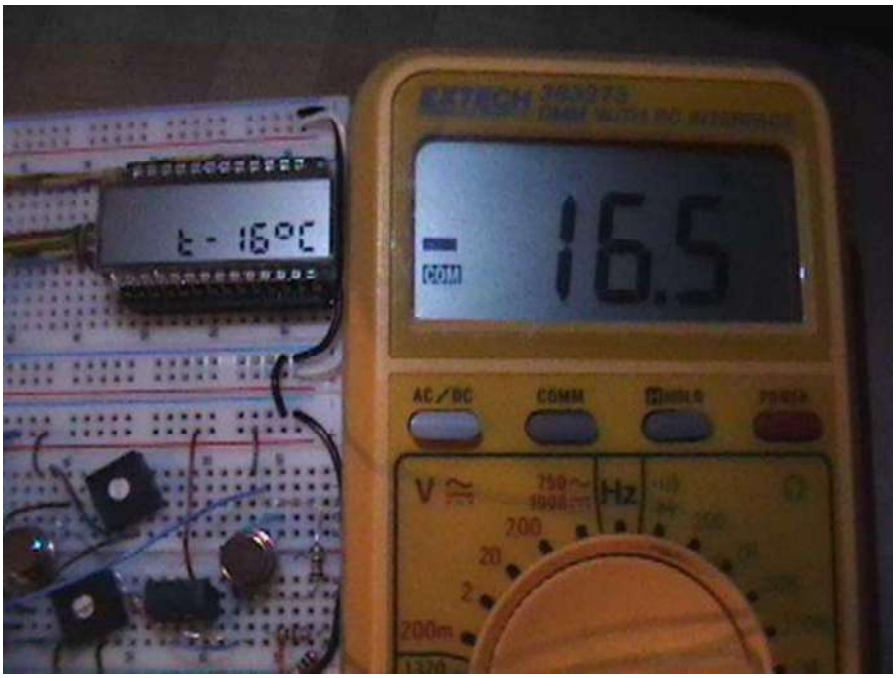


Figura 5.20 Prueba en mediciones negativas.

Pruebas del almacenamiento de datos en la memoria del microcontrolador

Los resultados presentados en esta sección están relacionados con las pruebas realizadas a la versión de código de programa que realiza las mediciones de forma automática, almacenando las mediciones en la memoria *Flash* del microcontrolador. En la Figura 5.21 se muestra la ventana del programa *Workbench* utilizado en este proyecto en el cual se realizaron los códigos del programa. En esta figura se observa el uso de dos ventanas para el monitoreo de las variables. Una de ellas es la llamada *watch* en donde se monitorean las variables deseadas. Esta ventana permite observar la expresión a monitorear así como su dirección de memoria, su valor en decimal y tipo de dato.

La ventana llamada *memory* se utiliza para monitorear las direcciones de memoria de las variables deseadas así como el valor guardado en dichas localidades. Este valor se expresa en hexadecimal. La ventana *memory* permite monitorear determinados segmentos de la memoria del microcontrolador. En este caso se monitorea el segmento llamado *INFO*,

el cual es el correspondiente a los bloques A y B de la memoria *Flash*, mostrando las direcciones en la primer columna de la ventana. Como se puede observar en la figura, al inicio de la ejecución del programa la memoria *Flash* se encuentra vacía (teniendo valores de unos por defecto), por lo que el llenado de esta va a realizarse de izquierda a derecha.

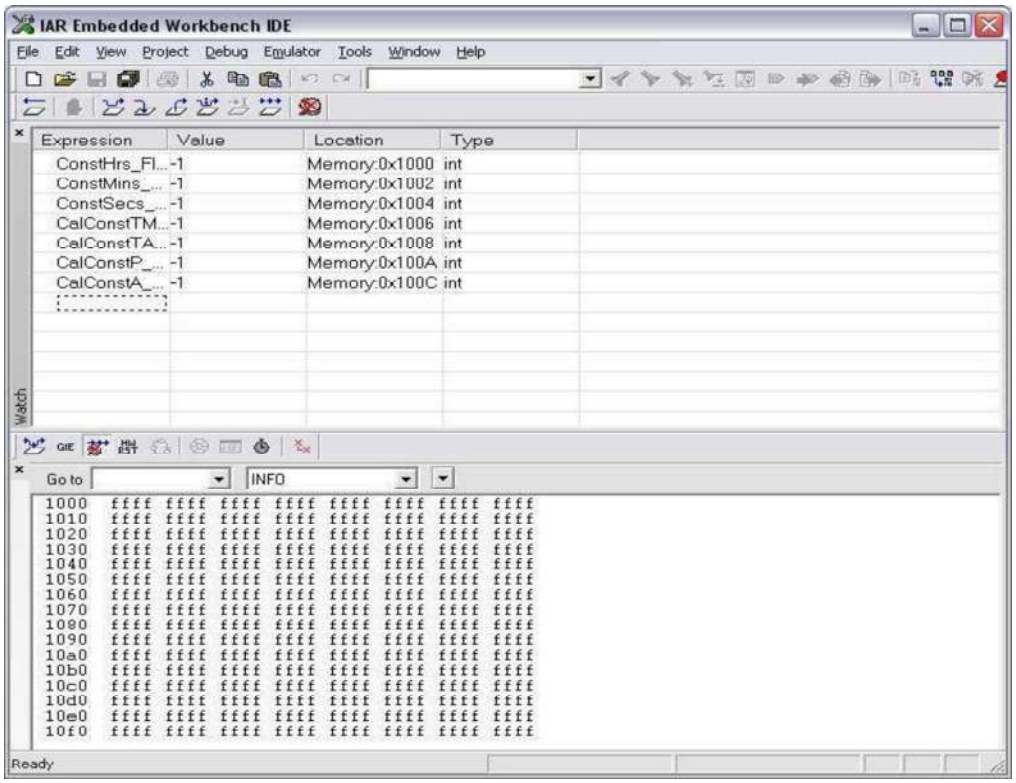


Figura 5.21 Ventanas *watch* y *memory* del programa *Workbench*.

En la Figura 5.22 se muestra el inicio de la escritura o almacenamiento de los datos obtenidos de las mediciones en la memoria *Flash* del microcontrolador. Como se puede observar en la ventana *watch* se muestran los valores de las mediciones de las diferentes variables. En este caso la lista está formada por la medición de la variable que almacena el dato de las horas, minutos, segundos, temperatura del microcontrolador, temperatura ambiente, presión barométrica y altitud.

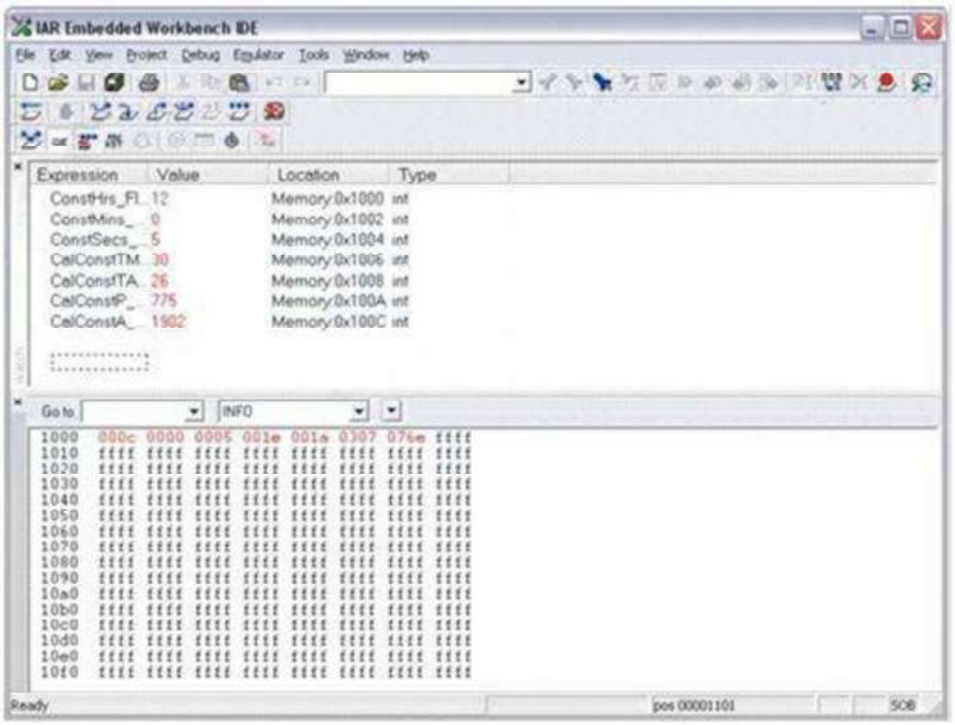


Figura 5.22 Primer escritura de las mediciones en la memoria *Flash*.

La Figura 5.22 los primeros siete datos que se encuentra en la ventana *watch* son los valores correspondientes a las mediciones en el orden ya mencionado. Estos datos se encuentran desplegados en sistema decimal. La ventana inferior de la Figura 5.23, correspondiente a la ventana *memory*, el primer dato almacenado (000c) corresponde al dato de hora, los siguientes son las demás mediciones en el orden ya mencionado. Las mediciones de las variables se realizan cada minuto al igual que el almacenamiento de estas en la memoria *Flash*. En la Figura 5.23 se muestra la memoria una vez que se llenó.

Debido a la cantidad de datos almacenados en este proyecto la memoria se llena cuando se han realizado 18 mediciones de todas las variables. Por lo tanto en este proyecto una vez que la memoria se llena, ésta es borrada y nuevamente se empiezan a escribir los siguientes nuevos datos.

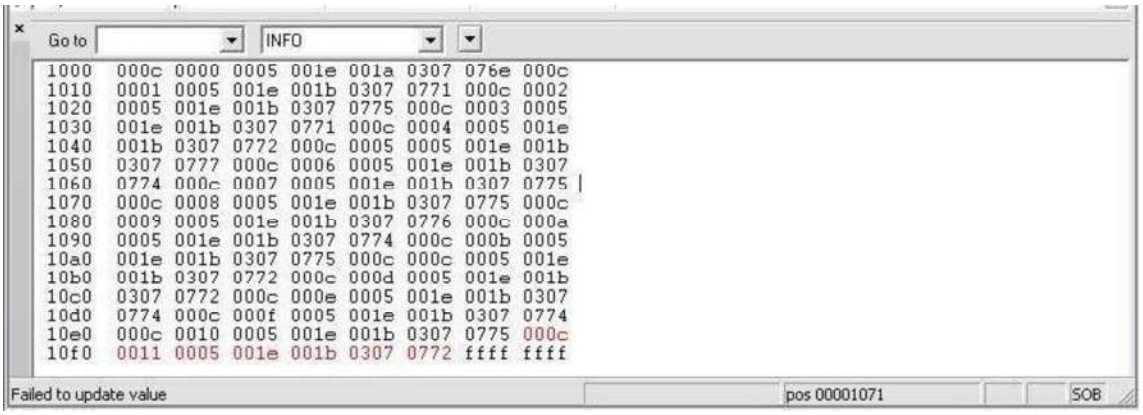


Figura 5.23 Memoria llena.

5.5 Conclusiones

En este capítulo se presentó el desarrollo de las pruebas aplicadas tanto al *software* como al *hardware* del sistema de medición Las pruebas realizadas al *hardware* fueron satisfactorias. En estas pruebas se observó que las mediciones realizadas en el *hardware* arrojan los resultados esperados durante el diseño. Las pruebas realizadas al *software* también fueron satisfactorias. Estas pruebas fueron visualizadas en la ventana *watch*. Las variables que son agregadas a esta ventana proporcionan las localidades de memoria en que se encuentran ubicadas las variables medidas, por lo que es de gran ayuda para el monitoreo de la variables almacenadas en la memoria *Flash* visualizadas en la ventana *memory*.

Los resultados del medidor desarrollado en este proyecto y los resultados obtenidos en la estación meteorológica comercial se comparan adecuadamente. En el caso de la medición de la presión los resultados difirieron en algunas unidades por lo que la estación meteorológica se utilizó para la calibración de esta variable. Por su parte, la medición de la temperatura ambiente mostró resultados muy buenos al momento de ser comparados con la estación comercial, por lo que no fue necesario realizar una calibración de esta medición.

Capítulo 6

Conclusiones y Trabajos Futuros

6.1 Conclusiones

En este trabajo de tesis se presentó la implementación de un medidor de temperatura y presión ambiental basado en un microcontrolador. La construcción de este medidor de variables meteorológicas permite contar con un equipo eficiente y económico en lugar de las costosas estaciones meteorológicas comerciales.

El sistema de medición basa su funcionamiento en un microcontrolador de bajo consumo de potencia, lo cual permite operarlo de forma autónoma y remota. En concordancia con las características del microcontrolador, los componentes adicionales como sensores y amplificadores operacionales se eligieron tomando siempre en cuenta niveles bajos de consumo de potencia. Debido a que la demanda de potencia de todos estos dispositivos en conjunto es bajo, es posible alimentar al sistema de medición con un banco de baterías recargables.

El desarrollo de este proyecto se basó en tres etapas importantes: La implementación de una etapa de acondicionamiento de señales, la programación del microcontrolador y la manipulación de la información capturada. Se implementó una etapa de acondicionamiento de señales para la medición de la temperatura debido a que la señal proporciona por el sensor LM35 sobrepasa los límites de voltaje de entrada al convertidor A/D. Por otra parte, se desarrollaron dos versiones del programa del microcontrolador en lenguaje C. El primer programa utiliza un botón de control que activa las mediciones y estas se despliegan en una pantalla LCD. La segunda versión funciona de manera automática realizando las mediciones en intervalos de tiempo pre-determinado y almacenando estas en la memoria *Flash* del microcontrolador asociándole el dato de tiempo que indica la hora en que se tomaron las mediciones.

Una vez que se implementó el sistema de medición, se realizaron pruebas correspondientes del *hardware* y *software*. Las pruebas del *hardware* consistieron en la realización de un conjunto de mediciones de voltajes en las terminales de salida de los sensores utilizados. Esto se realizó con la finalidad de comprobar estas mediciones con la

información que proporcionan los fabricantes de estos dispositivos. Las pruebas realizadas al *software* consistieron en la comparación de los cálculos de las mediciones obtenidas con los códigos desarrollados y los resultados calculados analíticamente. Los resultados obtenidos con la etapa de acondicionamiento asociada al sensor de temperatura fueron adecuadas, ya que se obtuvieron mediciones de temperatura correctas para el rango de temperaturas pre-establecidas en la etapa de diseño. Cabe mencionar que la etapa de acondicionamiento de señales desarrollada en este trabajo de tesis es muy básica debido a que se buscó simplificar al mínimo el uso de componentes electrónicos adicionales al microcontrolador. En este sentido es necesario enfatizar que en este trabajo se dio prioridad a explotar por medio de *software* los recursos disponibles en el microcontrolador.

Finalmente se realizaron las pruebas al *hardware* y *software* para comparar los resultados obtenidos de este proyecto con las mediciones proporcionadas por la estación meteorológica comercial *Vantage Pro2*. La comparación de los resultados permitió mostrar que la medición de la presión barométrica difería en algunas unidades, por lo cual se utilizó esta estación meteorológica comercial como referencia para calibrar esta medición. Por su parte, la variable asociada a la temperatura ambiente presentó buenos resultados comparado con la estación comercial por lo que no fue necesario calibrar esta medición. El uso del microcontrolador de baja potencia permitió desarrollar un sistema mínimo de medición que cumpliera las necesidades planteadas originalmente. Por ejemplo el consumo de potencia demandada por el sistema fue de 87.707mW (mili-watts), considerando la operación del microcontrolador, la etapa de acondicionamiento y los sensores. Además, el uso del convertidor analógico a digital de 16 *bits* permitió tener una resolución de 19.36µV/°C en la temperatura y 0.683µV/mB en la presión.

6.2 Trabajos Futuros

A continuación se enlistan los trabajos futuros para continuar el desarrollo presentado en esta tesis:

- Incorporar la medición de la velocidad y dirección del viento, radiación solar y precipitación pluvial.
- Almacenaje masivo de mediciones meteorológicas en una memoria *Flash* comercial.

- Transmisión remota de las mediciones meteorológicas por medio de mensajes de texto utilizando un teléfono celular.

Bibliografía

[AAGM 2001]

AAGM National Mountain Guide, Aconcagua 2006/7, 8 de Enero de 2008. www.aconcagua2002.com.ar

[SIC y TAR 2007]

Sistema de Ciencia y Tecnología Argentino, secyt, 8 de Enero de 2008. www.sicytar.secyt.gov.ar

[IGA 2007]

IGA, Instituto de Geofísica y Astronomía, 8 de Enero de 2008. www.iga.cu

[Dialnet 2002]
Dialnet, Universidad de la Rioja, 8 de Enero de 2008. www.dialnet.unirioja.es

[UNAM 2005]

UNAM, Geofísica UNAM, 8 de Enero de 2008. www.geofisica.unam.mx

[Davis Instruments 2004]

Davis Instruments, Vantage Pro2 Console Manual, Estados Unidos: Davis Instruments, 2004.

[Deitel y Deitel 2004]

H. M. Deitel y P. J. Deitel, Como programar en C/C++ y Java, Estados Unidos: Pretince Hall, 2004.

[RMCP 2007]

RMCP, Red Mundial de Científicos Peruanos, 8 de Enero de 2008. www.rmcp-peru.org

[Freescale Semiconductor 2005]

Freescale Semiconductor, 100kPa On-Chip Temperature Compensated & Calibrated Silicon Pressure Sensors, Estados Unidos: Freesale Semiconductor, 2005.

[Universidad de San Martín y Universidad de Buenos Aires 2006]

Universidad de San Martín y Universidad de Buenos Aires, Física re-Creativa, 8 de Enero de 2008, www.fisicarecreativa.com

[GEONICA 2007]

GEONICA S.A., Estaciones Meteorológicas Automáticas Serie MetoData 3000C, España: GEONICA S.A., 2007.

[Club Aéreo 2001]

Club Aéreo, Club Aéreo de Santiago, 8 de Enero de 2008. www.tobalabaereo.cl

[UNAM 2006]

UNAM, Dirección General de Planeación, 8 de Enero de 2008. www.planeacion.unam.mx

[Irwin 1997]

J. Irwin, Análisis Básico de Circuitos en Ingeniería, México: Pearson Educación, 1997.

[La Guía Metas 2005]

La Guía Metas, "Presión atmosférica, presión barométrica y altitud Conceptos y aplicaciones", *La Guía Metas*, Vol. 2, no. 2, págs. 1-5, Febrero 2005.

[Lázaro et al. 2001]

Lázaro Castillo Isidro Ignacio, Anzurez Marin Juan, Gaspar Valle Ciro Jesús, "Monitoreo de Variables Meteorológicas Usando LabVIEW", *XXIII Congreso Internacional de Ingeniería Electrónica, Chihuahua Chih. México*, Octubre 2001, págs. 219-224.

[Nagy 2003]

C. Nagy, Embedded Systems Design using the TI MSP430 Series, Estados Unidos: Elsevier, 2003.

[Navarro 2003]

Antonio Navarro, "Estación Meteorológica Automática", *CQ Radio Amateur*, Vol. 234, págs. 24-27, Mayo 2003.

[Malvino 2000]

A. Malvino, Principios de Electrónica, España: Mc Graw Hill, 2000.

[Manual_Texas_1 2003]

Texas Instruments, MSP430 IAR Assembler, Estados Unidos: Texas Instruments, 2003.

[Manual_Texas_2 2004]

Texas Instruments, MSP430 IAR C/C++, Estados Unidos: Texas Instruments, 2004.

[Manual_Texas_3 2005]

Texas Instruments, MSP430 IAR Embedded Workbench, Estados Unidos: Texas Instruments, 2005.

[Manual_Texas_4 2005]

Texas Instruments, MSP430F42x0 Mixed Signal Microcontroler, Estados Unidos: Texas Instruments, 2005.

[Manual_Texas_5 2005]

Texas Instruments, MSP-FET430 FLASH Emulation Tool, Estados Unidos: Texas Instruments, 2005.

[Manual_Texas_6 2006]

Texas Instruments, MSP430x1xx Family, Estados Unidos: Texas Instruments, 2006.

[Manual_Texas_7 2006]

Texas Instruments, MSP430x4xx Family, Estados Unidos: Texas Instruments, 2006.

[Manual_Texas_8 2005]

Texas Instruments, *Altimeter Demo*, Estados Unidos: Texas Instruments, 2005.

[Manual_National_A 2000]

National Semiconductor, Precision Centigrade Temperature Sensors LM35, Estados Unidos: National Semiconductor, 2000.

[Manual_National_B 2000]

National Semiconductor, LF155/LF156/LF355/LF356/LF357
JFET Input Operational Amplifiers, Estados Unidos: National Semiconductor,2000.

[*RAM* 2005]

RAM, Revista al Aficionado a la Meteorología, 8 de Enero de 2008. ram.meteored.com

[SoftBaugh 2003]

SoftBaugh, SBLCDA4, Estados Unidos: SoftBaugh, 2003.

[Wikipedia 2007]
Wikipedia, Artículo, 8 de Enero de 2008. www.wikipedia.org

Apéndice A

A continuación se presenta el listado del código desarrollado en lenguaje C para el programa basado en un botón.

```
#include <math.h>
#include "msp430x42x0.h"

//DEFINICIONES RELACIONADAS AL CIRCUITO
#define PUENTE_FUENTE          (0xc0)      //terminales P6.6/P6.7 I/O para fuente positiva 3V
#define PUSH_BUTTON            (0x02)      //botón conectado en la terminal P1.1

//CONSTANTES PARA LOS CÁLCULOS MATEMÁTICOS DE LAS MEDICIONES PRESIÓN Y ALTITUD
#define VOL_REF_EXT        (1.26923077f)         //voltaje de referencia externo
#define GAIN_32            (28.35f)              //valor de ganancia típico para 32x
#define BITS_RESOL         (32767)               //15 bits de resolucion (2^15)
#define UV_POR_LSB         (1000000.0f * VOL_REF_EXT / 2 / GAIN_32 / BITS_RESOL)    //voltaje de resolucion (P/A)
#define MB_POR_UV          (10.0f / 120.0f)            // pendiente para la presion con 3V (10mbar / 120uV)
#define P_ATMOS            (1013.25f)            // presión atmosférica a nivel del mar mbar
#define CTE_A              (9.81 / (287 * (273.16 + 25)))       //constante para el cálculo de la altitud

//CONSTANTES PARA LOS CÁLCULOS MATEMÁTICOS DE LA TEMPERATURA DEL MICRO Y LA AMBENTAL
#define VOL_REF_INT        (1.20f)             //voltaje de referencia interno
#define MV_POR_LSB         (1000.0f * VOL_REF_INT / 2 / 32767) //voltaje de resolución (temperatura del micro TM)
#define K_POR_MV           (1.0f / 1.32f)  //constante para el calculo de la temperatura del micro
#define MV_POR_LSB_TA      (VOL_REF_EXT / 2 / BITS_RESOL)      //voltaje de resolucion en mV

//DEFINICIÓN DE UN INTERVALO DE TIEMPO PARA EL ALMACENAMIENTO DE LAS CTES. DE CALIBRACIÓN
#define MODO_TIEMPO        (1)                 //indicador de tiempo

//LISTADO DE LOS MODOS DE PROGRAMA PARA LAS MEDICIONES A REALIZAR
enum
{
  MP_MEDICION_A,                    // modo: medición de altitud
  MP_MEDICION_P,                    // modo: medición de presión
  MP_DESPLIEGA_TIEMPO,              // modo: despliegue de hora
  MP_MEDICION_TEMP,                 // modo: medición de temperatura del micro
  MP_MEDICION_TA                    // modo: medición de temperatura ambiente
};


// DEFINICIÓN DE LOS SEGMENTOS DEL LCD (SoftBaugh SBLCDA4)
#define SEG_D   0x80                //  AAAA
#define SEG_C   0x40                // F    B
#define SEG_B   0x20                // F    B
#define SEG_A   0x10                //  GGGG
#define SEG_H   0x08                // E    C
#define SEG_E   0x04                // E    C
#define SEG_G   0x02                //  DDDD
```

```
#define SEG_F   0x01

//GENERACIÓN DE NÚMEROS Y LETRAS DE INDICACIÓN
const char LCD_Tab[] = {
  SEG_A + SEG_B + SEG_C + SEG_D + SEG_E + SEG_F,         // "0"
  SEG_B + SEG_C,                                        // "1"
  SEG_A + SEG_B + SEG_D + SEG_E + SEG_G,                // "2"
  SEG_A + SEG_B + SEG_C + SEG_D + SEG_G,                // "3"
  SEG_B + SEG_C + SEG_F + SEG_G,                        // "4"
  SEG_A + SEG_C + SEG_D + SEG_F + SEG_G,                // "5"
  SEG_A + SEG_C + SEG_D + SEG_E + SEG_F + SEG_G,        // "6"
  SEG_A + SEG_B + SEG_C,                                // "7"
  SEG_A + SEG_B + SEG_C + SEG_D + SEG_E + SEG_F + SEG_G,   // "8"
  SEG_A + SEG_B + SEG_C + SEG_D + SEG_F + SEG_G,        // "9"
  SEG_A + SEG_B + SEG_C + SEG_E + SEG_F + SEG_G,        // "A"
  SEG_B + SEG_C + SEG_E + SEG_F + SEG_G,                // "H"
  SEG_A + SEG_D + SEG_E + SEG_F,                        // "C"
  SEG_D + SEG_E + SEG_F,                                // "L"
  SEG_A + SEG_B + SEG_E + SEG_F + SEG_G,                // "P"
  0x00                                                  // vacio
};

#define DIG_MENOS   (SEG_G)                                         // '-'
#define DIG_T       (SEG_D + SEG_E + SEG_F + SEG_G)                 // 't'
#define DIG_DEGR    (SEG_A + SEG_B + SEG_F + SEG_G)                 // 'º'
#define DIG_L       (SEG_D + SEG_E + SEG_F)                         // 'L'
#define DIG_O       (SEG_A + SEG_B + SEG_C + SEG_D + SEG_E + SEG_F)   // 'O'
#define DIG_H       (SEG_B + SEG_C + SEG_E + SEG_F + SEG_G)         // 'H'
#define DIG_I      (SEG_B + SEG_C)                                  // 'I'
#define DIG_C       (SEG_A + SEG_D + SEG_E + SEG_F)                 // 'C'

//VARIEBLES PARA LA OPERACIÓN DEL SD16_A
static unsigned int SD16TempCont;                   // contador del número de muestras colectadas
static long SD16Temp;                               // variable temporal para la suma de las muestras
static long SD16Result;                             // resultado del promedio final de
static unsigned int SD16NumConversion               //numero de muestras a colectar
//VARIABLES PARA LOS CÁLCULOS Y CALIBRACIÓN DE LAS MEDICIONES
float P;
float PA;
float Ps;
float Pb;
float a = 44365.5723;
float b = 5.256;
float X;
float X_CAL;
float T;
float AUX1;
float AUX2;
float H;
float TA;
float TM;
float CAL_TM = 0.0;
float CAL_TA = 0.0;
float CAL_P = 943;//3500       //3500.-guia meta
float CAL_mB = 10.26;
float CAL_A = 0.0;
```

```
// VARIABLES PARA EL CONTROL DEL FLUJO DEL PROGRAMA
static unsigned char ModoPrograma = MP_DESPLIEGA_TIEMPO;   // modo de programa inicial
static unsigned char ContBotonPres = 0;                    // detecta cuanto tiempo duró presionado el botón
static unsigned int StatusFlags = 0;                       // registro de la bandera de estado (status flags)
static unsigned char ModoCont = 0;

#define SF_BT_TIEMPO       0x0001                          // definiciones de bits de las banderas de edo.
#define SF_PB1_LIBERADO    0x0002
#define SF_SD16_INICIO     0x0004
#define SF_SD16_LISTO      0x0008
#define SF_ACT_PANTALLA    0x0010
#define SF_PB1_PRESIONADO  0x0020
#define SF_PB1_CERRADO     0x0040
#define SF_CONSER_DESP     0x0080

//VARIABLES PARA CALIBRACIÓN
static int CalConstA;                                      // variables temporales para la calibración
static int CalConstP;
static int CalConstTM;
static int CalConstTA;

//VARIABLES GLOBALES PARA EL RELOJ DE TIEMPO REAL (RTR)
static unsigned char Hrs  = 0x12;                          // hora definida 12:00:00
static unsigned char Mins = 0x00;
static unsigned char Secs = 0x00;

//VARIABLES PARA ALMACENAR SU VALOR EN LA MEMORIA FLASH
#pragma dataseg = INFOA                    // bloque B de información de la mem. Flash
__no_init static int CalConstTM_Flash;
__no_init static int CalConstTA_Flash;
__no_init static int CalConstP_Flash;
__no_init static int CalConstA_Flash;

#pragma dataseg = default

//FUNCIONES
void IniSis(void);
void RTR_Inc(unsigned int CambioMins);
void IniciaConversion(void);
void IniciaSigConversion(void);
void DetenerConversion(void);
void AlmacenaFlash(void);
void Desp_Valor(unsigned int RecorrerIzq, int Valor);
void Desp_BCD(unsigned long Valor);

//-------------------------------------------------------------------------------
void main(void)
{
  IniSis();                              //se inicializa el sistema
  CalConstTM = CalConstTM_Flash;
  CalConstTA = CalConstTA_Flash;
  CalConstP = CalConstP_Flash;
  CalConstA = CalConstA_Flash;           //se cargan las variables de calibración

  __enable_interrupt();
```

```
Desp_BCD(0x08888888);                    //prueba que todos los seg. del LCD se enciendan

while (1)
{
 __bis_SR_register(LPM3_bits);            //se activa el LPM3, espera a ser llamado

 if (StatusFlags & SF_BT_TIEMPO)
 {
  RTR_Inc(0);                             // incremento de los segundos del RTR

  if(ModoPrograma == MP_DESPLIEGA_TIEMPO)
  {
     StatusFlags |= SF_ACT_PANTALLA;      //se actualiza el despliegue de la Hora del RTR
  }
  StatusFlags &= ~SF_BT_TIEMPO;           // Evento procesado
 }

 if ((StatusFlags & SF_PB1_LIBERADO) && ( ContBotonPres < 62))
 {
  switch (ModoPrograma)                   //se realizan las mediciones una a una
  {                                       //cada vez que se presiona el botón
    case MP_MEDICION_P :
      DetenerConversion();
      ModoPrograma = MP_MEDICION_A;
      IniciaConversion();
      StatusFlags |= SF_SD16_INICIO;
      break;
    case MP_MEDICION_A :
      DetenerConversion();
      ModoPrograma = MP_DESPLIEGA_TIEMPO;
      break;
    case MP_DESPLIEGA_TIEMPO :
      ModoPrograma = MP_MEDICION_TEMP;
      IniciaConversion();
      StatusFlags |= SF_SD16_INICIO;
      break;
    case MP_MEDICION_TEMP :
      DetenerConversion();
      ModoPrograma = MP_MEDICION_TA;
      IniciaConversion();
      StatusFlags |= SF_SD16_INICIO;
      break;
    case MP_MEDICION_TA :
      DetenerConversion();
      ModoPrograma = MP_MEDICION_P;
      IniciaConversion();
      StatusFlags |= SF_SD16_INICIO;
      break;


  }
  StatusFlags &= ~SF_PB1_LIBERADO;
 }

 if (StatusFlags & SF_SD16_LISTO)
```

```
{
 switch (ModoPrograma)
 {
  case MP_MEDICION_A :
  case MP_MEDICION_P :
    SD16Result = SD16Temp >> 5;            //cálculo del promedio dividiendo 32 muestras
   // SD16Result = SD16Temp;
    StatusFlags |= SF_SD16_INICIO;         //se dispara la conversion del SD16_A
    break;
    case MP_MEDICION_TA :
    SD16Result = SD16Temp >> 5;
   //SD16Result = SD16Temp;
    StatusFlags |= SF_SD16_INICIO;
    break;
  case MP_MEDICION_TEMP :
    SD16Result = SD16Temp;                 //mover el resultado
    break;
 }
 StatusFlags |= SF_ACT_PANTALL       //actualizar despliegue de pantalla
 StatusFlags &= ~SF_SD16_LI             // evento procesado
}

if (StatusFlags & SF_SD16_INICIO)
{
 IniciaSigConversion();                  //iniciar las conversiones con el SD16_A
 StatusFlags &= ~SF_SD16_INICIO;         //evento procesado
}

if (StatusFlags & SF_ACT_PANTALLA)         //se requiere actualizar la pantalla??
{
 if (StatusFlags & SF_CONSER_DESP)         //conservar el contenido de la pantalla??
 {
  StatusFlags &= ~SF_CONSER_DESP;          //evento procesado
 }
 else
 {

  switch (ModoPrograma)
  {


    case MP_MEDICION_P :
      Desp_BCD(0x0feffffff;                              //'P     '

      //Desp_Valor(0, (((float)(SD16Result) * UV_POR_LSB + CAL_P ))*MB_POR_UV);

      P = (((float)(SD16Result) * UV_POR_LSB + CAL_P))*MB_POR_UV + CAL_mB; //cálculo de la
presion atmosférica
      PA = (((float)(SD16Result) * UV_POR_LSB + CAL_P ))*MB_POR_UV; //cálculo de la presion
atmosférica
      X = (((float)(SD16Result) * UV_POR_LSB ));                 // voltaje calculado
      X_CAL = (((float)(SD16Result) * UV_POR_LSB + CAL_P));      //voltaje calibrado

      AUX1 = 1 / CTE_A;                                  //cálculo de la altitud
      AUX2 = 1 / AUX1 * (log(exp(1)) / log(10));
      H = 1 / AUX2 * (log(P_ATMOS / PA) / log(10)) + CAL_A;
```

```
        Ps = P_ATMOS*pow(1-22.5569E-6*H,b);
        Pb= (P + 1013.25-Ps);//*0.75006168;
        Desp_Valor(0,Pb);                                   //desplegar la medición en mB

        //Po = P * (pow(((a)/(a-H)),b))*.75006168;
        //Desp_Valor(0,Po);

          if (++ModoCont > MODO_TIEMPO)                     // almacenar las ctes de calibración en la Flash?
       {
        CalConstP= CAL_P;
        AlmacenaFlash();
       }
        break;
       case MP_MEDICION_A :
         Desp_BCD(0x0faffff);                               //'A    '

         //AUX1 = 1 / CTE_A;

         //AUX2 = 1 / AUX1 * (log(exp(1)) / log(10));

        // H = 1 / AUX2 * (log(P_ATMOS / PA) / log(10)) + CAL_A;     //cálculo de la altitud

         Desp_Valor(0, H);                                  //desplegar la medición en metros
         if (++ModoCont > MODO_TIEMPO)                      //almacenar las ctes de calibración en la Flash?
       {
        CalConstA= CAL_A;
        AlmacenaFlash();
       }
        break;
       case MP_DESPLIEGA_TIEMPO :
         Desp_BCD(0x0f000000 + ((long)Hrs << 16) + ((int)Mins << 8) + Secs);  //desplegar la hora en
tiempo real
        break;

       case MP_MEDICION_TEMP :
        Desp_BCD(0x0fffffff);                               //limpiar lapantalla

        if (Secs & 0x01)                                    //parpadeo del signo de grados
          LCDM6 = DIG_DEGR;
          LCDM7 = DIG_C;                                    // 'C   '
           TM = (float)SD16Result * MV_POR_LSB * K_POR_MV - 273 + CAL_TM;  //calculo de la
temperatura del micro
           Desp_Valor(2, TM);
           if (++ModoCont > MODO_TIEMPO)                    //almacenar las ctes de calibración en la
Flash?
       {
        CalConstTM= CAL_TM;
        AlmacenaFlash();
       }

        break;

       case MP_MEDICION_TA :                                //cálculo de la temperatura ambiente
        LCDM2 = DIG_T;                                      //'t'
        LCDM6 = DIG_DEGR;
```

```
          LCDM7 = DIG_C;                                        //'C'
          TA  =  ((float)(SD16Result)  *  MV_POR_LSB_TA  -  0.170232)*100/0.309512  +  CAL_TA;       //
DONDE LA T VIENE SIENDO EL VOLTAJE DE SALIDA DE LA INSTRUMENTACION
          Desp_Valor(2,TA);
          if (++ModoCont > MODO_TIEMPO)                                 //almacenar las ctes de calibración en la
Flash?
         {
          CalConstTA= CAL_TA;
          AlmacenaFlash();
         }
          break;
       }
       StatusFlags &= ~SF_ACT_PANTALLA;                                 //evento procesado
     }
    }
  }
}

//-------------------------------------------------------------------------------
void IniSis(void)
{
  int i;
  char *pLCD = (char *)&LCDM1;

  WDTCTL = WDTPW + WDTHOLD;                    // detener el  WDT
  FLL_CTL0 |= XCAP18PF;                        // activar la carga de los capacitores externos del cristal

  for (i = 0; i < 20; i++)                     // limpiar la memoria del LCD
    *pLCD++ = 0;

  LCDACTL = LCDFREQ_96 + LCD4MUX + LCDON;       // 4mux LCD, ativar la frecuancia del LCD
  LCDAPCTL0 = 0x0F;                            // Segs S0-S15 = salidas
  LCDAVCTL0 = LCDCPEN;                         // habilitar la carga del LCD_A

  BTCTL = BTDIV + BT_fCLK2_DIV128;
  IE2 |= BTIE;                                 // habilitar la interrupciones del BT

  P1OUT = 0x00;                                // P1OUTs = 0
  P1DIR = 0xff & ~(PUSH_BUTTON);               //terminales sin usar se configuran como salidas
  P1IES = PUSH_BUTTON;
  P1IFG = 0x00;                                // restalecer las banderas del P1
//  P1IE = PUSH_BUTTON1 + PUSH_BUTTON2;          //habilitar las interrupciones del boton
  P1IE = PUSH_BUTTON;
  P5SEL = 0xff;                                // activar las terminales del LCD Rxx y COM
  P6OUT = 0x00;                                // P6OUTs = 0
  P6DIR = 0xff;                                // terminales sin usar se configuran como salidas
  P6SEL = 0x0f;                                // seleción de los canales analógicos de SD16_A
//  P6SEL = 0x33;
}
//-------------------------------------------------------------------------------
// INCREMENTO EN LAS VARIABLES DEL RTR
//
// CambioMins == 0 --> incrementan seg
// CambioMins == 1 --> incrementan min
//-------------------------------------------------------------------------------
void RTR_Inc(unsigned int CambioMins)
```

```
{
  Secs = __bcd_add_short(Secs, 1);
  if ((Secs == 0x60) || CambioMins)
  {
    Secs = 0x00;
    Mins = __bcd_add_short(Mins, 1);
    if (Mins == 0x60)
    {
      Mins = 0x00;
      Hrs = __bcd_add_short(Hrs, 1);
      if (Hrs == 0x13)
        Hrs = 0x01;
    }
  }
}

//-------------------------------------------------------------------------
void AlmacenaFlash(void)
{

  FCTL2 = FWKEY + FSSEL1 + FN1;           // SMCLK/3 = ~333kHz
  FCTL3 = FWKEY;                          // se quita el candado
  *(unsigned int *)0x1080 = 0;                 // indicación de la localidad de memoria en la que comienza a
escribir
  FCTL1 = FWKEY + WRT;                    // habilitar la escritura
  CalConstTM_Flash = CalConstTM;
  CalConstTA_Flash = CalConstTA;
  CalConstP_Flash = CalConstP;
  CalConstA_Flash = CalConstA;            // constantes de calibración
  FCTL1 = FWKEY;                          // deshabilitar la escritura
  FCTL3 = FWKEY + LOCK;                   // activar candado

  StatusFlags |= SF_CONSER_DESP;          // mantener el despliegue de los datos
}
//-------------------------------------------------------------------------
//CONVERSIÓN DEL NUMERO ENTERO DE 16 BITS A BCD PARA SER DESPLEGADO EN EL LCD
//EL DIGITO DE MAS A LA IZQUIERDA ES USADO COMO INFORMACIÓN DE SIGNO
//-------------------------------------------------------------------------
void Desp_Valor(unsigned int RecorrerIzq, int Valor)
{
  unsigned int i;
  unsigned int salida;
  char fNeg = 0;
  char *pLCD = (char *)&LCDM7 - RecorrerIzq;

  if (Valor < 0)                          // condición de valores negativos
  {
    Valor = -Valor;                       // valor negativo
    fNeg = 1;                             // activar la bandera de signo negtivo
  }

  if (Valor > 9999)                       // condición de valor máximo
  {
    if (fNeg)
    {
      *pLCD-- = DIG_O;                    //  condicion para indicar que el valor es muy pequeño
```

```
      *pLCD = DIG_L;
    }
    else
    {
      *pLCD-- = DIG_I;                          // condición para indicar que el valor es muy grande
      *pLCD = DIG_H;
    }
    return;
  }

  for (i = 16, salida = 0; i; i--)              // conversión BCD de 16-bits
  {
    salida = __bcd_add_short(salida, salida);
    if (Valor & 0x8000)
      salida = __bcd_add_short(salida, 1);
    Valor <<= 1;
  }

  for (i = 4; i; i--)                           // Proceso de 4 digitos
  {
    *pLCD-- = LCD_Tab[salida & 0x0f];           // Segmentos del LCD
    salida >>= 4;                               // Proceso del siguiente digit
    if (salida == 0 || i == 1)                  // llevar el cero o el último digito
    {
      if (fNeg)                                 // Añadir el signo de información
        *pLCD = DIG_MENOS;
      break;
    }
  }
}

//---------------------------------------------------------------------------
// DESPLEGAR EL VAOR EN BCD EN EL LCD
//---------------------------------------------------------------------------
void Desp_BCD(unsigned long Valor)
{
  char *pLCD = (char *)&LCDM7;
  int i;

  for (i = 0; i < 7; i++)                       // Proceso de 7 digitos
  {
    *pLCD-- = LCD_Tab[Valor & 0x0f];            // enviar segmentos al LCD
    Valor >>= 4;                                // Procesar el siguiente digito
  }
}

//---------------------------------------------------------------------------
void IniciaConversion(void)
{
  switch (ModoPrograma)                         // Preparar la conversión del SD16
  {
    case MP_MEDICION_A :
    case MP_MEDICION_P :
      P6OUT |= PUENTE_FUENTE;                   // activar el puente de alimentación
      SD16CTL = SD16SSEL_2;                     // Utilizar el ACLK para el SD16
      SD16INCTL0 = SD16GAIN_32 + SD16INCH_0;    // ganancia de 32x , canal A0
```

```
    SD16CCTL0 = SD16BUF_1 + SD16OSR_1024 + SD16DF + SD16IE;// conversión continua,
complemento a dos
     break;
   case MP_MEDICION_TEMP :
     SD16CTL = SD16SSEL_2;                              // Usar ACLK
     SD16INCTL0 = SD16GAIN_1 + SD16INCH_6;    // ganancia de 1x , canal A6
     SD16CCTL0 = SD16OSR_1024 + SD16DF + SD16IE;
     break;

  case MP_MEDICION_TA :
     P6OUT |= PUENTE_FUENTE;
     SD16CTL = SD16SSEL_2;
     SD16INCTL0 = SD16GAIN_1 + SD16INCH_1;
     SD16CCTL0 = SD16BUF_1 + SD16OSR_1024 + SD16DF + SD16IE;
     break;
  }
}
//-------------------------------------------------------------------------------
void IniciaSigConversion(void)
{
  SD16Temp = 0;
  SD16TempCont = 0;

  switch (ModoPrograma)                    // comenzar la conversion del SD16
  {
    case MP_MEDICION_A :
    case MP_MEDICION_P :
      SD16NumConversiones = 32;              // promediar 32 mediciones
      break;
    case MP_MEDICION_TA :
      SD16NumConversiones = 32;
      break;
    case MP_MEDICION_TEMP :
       SD16NumConversiones = 1;              // no promediar
      SD16CTL |= SD16VMIDON + SD16REFON;     // habilitar el Vref interno
      TAR = 65535 - 163;                     // retardar 5ms
      TACTL = TASSEL_1 + MC_2;               // ACLK, iniciar el modo continuo
      while (!(TACTL & TAIFG));
      TACTL = 0x00;                          // detener el reloj A
      break;
  }

  SD16CCTL0 |= SD16SC;                       // comenzar la conversión
}
//-------------------------------------------------------------------------------
void DetenerConversion(void)
{
  SD16CCTL0 &= ~SD16SC;                      // Deshabilitar las conversiones

  switch (ModoPrograma)
  {
    case MP_MEDICION_A :
    case MP_MEDICION_P :
    case MP_MEDICION_TA :

      P6OUT &= ~PUENTE_FUENTE;               // apagar el puente de alimentación
```

```
    break;
   case MP_MEDICION_TEMP :
    SD16CTL = 0x00;                    // deshabitar la referencia del SD16
    break;
  }
}
//-------------------------------------------------------------------------------
// ISR DEL SD16
//-------------------------------------------------------------------------------
#pragma vector = SD16_VECTOR
__interrupt void SD16_ISR(void)
{
 SD16Temp += (long)(int)SD16MEM0;                    // leer 15+1 bits

 if (++SD16TempCont >= SD16NumConversiones)      // son suficientes muestras?
 {
  SD16CCTL0 &= ~SD16SC;                             // deshabilitar las conversiones
  SD16CTL &= ~(SD16VMIDON + SD16REFON);     // deshabilitar la referencia del SD16
  StatusFlags |= SF_SD16_LISTO;                     // la conversión resultante esta lista
  __bic_SR_register_on_exit(LPM3_bits);             // fin de la ISR del SD16
 }
}
//-------------------------------------------------------------------------------
//ISR del BT ejecutada cada segundo
//-------------------------------------------------------------------------------
#pragma vector = BASICTIMER_VECTOR
__interrupt void BT_ISR(void)
{
 StatusFlags |= SF_BT_TIEMPO;                  // indica el proceso del BT
 __bic_SR_register_on_exit(LPM3_bits);        // fin de la ISR del BT
}
//-------------------------------------------------------------------------------
// ISR del Puerto_1, ejecutada cada vez que se presiona el botón
//-------------------------------------------------------------------------------
#pragma vector = PORT1_VECTOR
__interrupt void PORT1_ISR(void)
{

 P1IE &= ~(PUSH_BUTTON);                   // Deshabilitar las interrupciones del botón
 WDTCTL = WDT_ADLY_16;                     // WDT funcionado como intervalo de tiempo 16ms
 IFG1 &= WDTIFG;
 IE1 |= WDTIE;                          // habilitar las interrupciones del WDT
}
//-------------------------------------------------------------------------------
// ISR del WDT, es ejecutada cada vez que se presiona el botón
//y se encarga de monitorear el estado del botón
//-------------------------------------------------------------------------------
#pragma vector = WDT_VECTOR
__interrupt void WDT_ISR(void)
{


 if (P1IFG & PUSH_BUTTON)                        // el botón fue presionado?
 {
  if (!(StatusFlags & SF_PB1_CERRADO))          // el botón está siendo presionado?
  {
```

```
    ContBotonPres = 0;                          // se restablece el contador de tiempo
    StatusFlags |= SF_PB1_CERRADO;
    StatusFlags |= SF_PB1_PRESIONADO;           // el botón fue presionado
    StatusFlags &= ~SF_PB1_LIBERADO;
   }
   if (P1IN & PUSH_BUTTON)                      // Botón liberado?
   {
    P1IFG &= ~PUSH_BUTTON;                      // restablecer la banderas
    StatusFlags &= ~SF_PB1_CERRADO;
    StatusFlags &= ~SF_PB1_PRESIONADO;
    StatusFlags |= SF_PB1_LIBERADO;
   }
   if (ContBotonPres != 0xff) ContBotonPres++;      // Incrementar el contador de tiempo
    __bic_SR_register_on_exit(LPM3_bits);           // fin de la ISR del WD
  }

  if (!(P1IFG & (PUSH_BUTTON)))
  {
   WDTCTL = WDTPW + WDTHOLD;                    // detener el WDT
   P1IE |= PUSH_BUTTON;                         // hablitar las interrupciones del botón
  }
}
//-------------------------------------------------------------------------------
```

A continuación se presenta el listado del código desarrollado en lenguaje C para el programa basado en intervalos de tiempo.

```
#include <math.h>
#include "msp430x42x0.h"

//DEFINICIONES RELACIONADAS AL CIRCUITO
#define PUENTE_FUENTE      (0xc0)

//CONSTANTES PARA LOS CÁLCULOS MATEMÁTICOS DE LAS MEDICIONES PRESIÓN Y ALTITUD
#define VOL_REF_EXT   (1.26923077f)          //voltaje de referencia externo
#define GAIN_32      (28.35f)                //valor de ganancia típico para 32x
#define BITS_RESOL    (32767)                //15 bits de resolucion (2^15)
#define UV_POR_LSB        (1000000.0f * VOL_REF_EXT / 2 / GAIN_32 / BITS_RESOL)//voltaje de resolucion (P/A)
#define MB_POR_UV      (10.0f / 120.0f)       // pendiente para la presion con 3V (10mbar / 120uV)
#define P_ATMOS        (1013.25f)             // presion atmosférica a nivel del mar mbar
#define CTE_A          (9.81 / (287 * (273.16 + 25)))  //constante para el cálculo de la altitud

//CONSTANTES PARA LOS CÁLCULOS MATEMÁTICOS DE LA TEMPERATURA DEL MICRO Y LA AMBENTAL
#define UV_POR_LSB_TA   (VOL_REF_EXT / 2 / 32767)  //voltaje de resolucion en mV
#define VREF_VOLT_INT   (1.20f)              //voltaje de referencia interno
#define MV_POR_LSB      (1000.0f * VREF_VOLT_INT / 2 / 32767) //voltaje de resolución (temperatura del micro TM)
#define K_POR_MV       (1.0f / 1.32f)    //constante para el calculo de la temperatura del micro

enum
{
  MP_MEDICION_A,                  // modo: medición de altitud
  MP_MEDICION_P,                  // modo: medición de presión
  MP_MEDICION_TEMP,               // modo: medición de temperatura del micro
  MP_MEDICION_TA                  // modo: medición de temperatura ambiente

};

//VARIEBLES PARA LA OPERACIÓN DEL SD16_A
static unsigned int SD16TempCont;            // contador del número de muestras colectadas
static long SD16Temp;                        // variable temporal para la suma de las muestras
static long SD16Result;                      // resultado del promedio final de
static unsigned int SD16NumConversiones;     //numero de muestras a colectar

//VARIABLES PARA LOS CÁLCULOS Y CALIBRACIÓN DE LAS MEDICIONES
float P;
float PA;
float Ps;
float Pb;
float b=5.256;
float T;
float X;
float X_CAL;
float AUX1;
float AUX2;
float H;
```

```
float A;
float TA;
float TM;
int i;
int j;
float CAL_TM = 0;
float CAL_TA = 0;
float CAL_P = 943;
float CAL_mB = 10.26;
float CAL_A = 0;

static unsigned char ModoPrograma;              // modo de programa
static unsigned int StatusFlags = 0;            // registro de la bandera de estado (status flags)


#define SF_BT_TIEMPO     0x0001                 // definiciones de bits de las banderas de edo.
#define SF_SD16_INICIO   0x0002
#define SF_SD16_LISTO    0x0004

static int ConstHrs;                            //variables temporales para las mediciones
static int ConstMins;
static int ConstSecs;
static int CalConstA;
static int CalConstP;
static int CalConstTM;
static int CalConstTA;

//APUNTADOR DE LOCAIDAD DE MEMORIA
int *p1;

static unsigned int Secs = 0;                   // hora definida 12:00:00
static unsigned int Mins = 0;
static unsigned int Hrs  = 12;

#pragma dataseg = INFOB                         //variables correspondientes para el almacenamiento
__no_init static int ConstHrs_Flash;            //de mediciones en la memoria Flash
__no_init static int ConstMins_Flash;
__no_init static int ConstSecs_Flash;
__no_init static int CalConstTM_Flash;
__no_init static int CalConstTA_Flash;
__no_init static int CalConstP_Flash;
__no_init static int CalConstA_Flash;

#pragma dataseg = default

//FUNCIONES
void IniSis(void);
void RTR_Inc(unsigned int CambioMins);
void IniciaConversion(void);
void IniciaSigConversion(void);
void DetenerConversion(void);
void AlmacenaFlash(void);



//-------------------------------------------------------------------------------
```

```
void main(void)
{
 IniSis();                      //se inicializa el sistema

 ConstHrs = ConstHrs_Flash;         //se cargan las variables en la memoria Flash
 ConstMins = ConstMins_Flash;
 ConstSecs = ConstSecs_Flash;
 CalConstTM = CalConstTM_Flash;
 CalConstTA = CalConstTA_Flash;
 CalConstP = CalConstP_Flash;
 CalConstA = CalConstA_Flash;


 __enable_interrupt();             //se habilita el vector de inerrupciones

 while (1)
 {
  __bis_SR_register(LPM3_bits);     //se activa el LPM3, espera a ser llamado


  if (StatusFlags & SF_BT_TIEMPO)
  {
   RTR_Inc(0);
   StatusFlags &= ~SF_BT_TIEMPO;
  }

      if (Secs == 1)//(Secs == 0x01)          medicion de la tem. del micro
      {

        ModoPrograma = MP_MEDICION_TEMP;
        IniciaConversion();
        StatusFlags |= SF_SD16_INICIO;
        if (StatusFlags & SF_SD16_LISTO)
        {
          SD16Result = SD16Temp;
          StatusFlags &= ~SF_SD16_LISTO;
        }
        if (StatusFlags & SF_SD16_INICIO)
        {
          IniciaSigConversion();
          StatusFlags &= ~SF_SD16_INICIO;
        }
       TM = (float)SD16Result * MV_POR_LSB * K_POR_MV - 273 + CAL_TM;
       CalConstTM = TM;
      }
      if (Secs == 2)//(Secs == 0x02)        medición de la temperatura ambiente
      {

        ModoPrograma = MP_MEDICION_TA;
        IniciaConversion();
        StatusFlags |= SF_SD16_INICIO;
        if (StatusFlags & SF_SD16_LISTO)
        {
          SD16Result = SD16Temp >> 4;              //cálculo del promedio
          StatusFlags &= ~SF_SD16_LISTO;
        }
```

```
        if (StatusFlags & SF_SD16_INICIO)
        {
          IniciaSigConversion();
          StatusFlags &= ~SF_SD16_INICIO;                    }
        TA = ((float)(SD16Result) * UV_POR_LSB_TA - 0.170232)*100/0.309512 + CAL_TA;
        CalConstTA = TA;
      }
      if (Secs == 3)//(Secs == 0x03)              medicion de la presión
      {
        ModoPrograma = MP_MEDICION_P;
        IniciaConversion();
        StatusFlags |= SF_SD16_INICIO;
        if (StatusFlags & SF_SD16_LISTO)
        {
          SD16Result = SD16Temp >> 4;              //cálculo del promedio
          StatusFlags &= ~SF_SD16_LISTO;
        }
        if (StatusFlags & SF_SD16_INICIO)
        {
          IniciaSigConversion();
          StatusFlags &= ~SF_SD16_INICIO;
        }

        P = (((float)(SD16Result) * UV_POR_LSB + CAL_P))*MB_POR_UV + CAL_mB; //cálculo de la
presion atmosférica
        PA = (((float)(SD16Result) * UV_POR_LSB + CAL_P ))*MB_POR_UV; //cálculo de la presion
atmosférica
        X = (((float)(SD16Result) * UV_POR_LSB ));                 // voltaje calculado
        X_CAL = (((float)(SD16Result) * UV_POR_LSB + CAL_P));      //voltaje calibrado

        AUX1 = 1 / CTE_A;                                  //cálculo de la altitud
        AUX2 = 1 / AUX1 * (log(exp(1)) / log(10));
        H = 1 / AUX2 * (log(P_ATMOS / PA) / log(10)) ;

        Ps = P_ATMOS*pow(1-22.5569E-6*H,b);
        Pb= (P + (1013.25-Ps))*.75006168;
        CalConstP = Pb;
      }
      if (Secs == 4)//(Secs == 0x04)                medicion de la altitud
      {
        ModoPrograma = MP_MEDICION_A;
        IniciaConversion();
        StatusFlags |= SF_SD16_INICIO;
        if (StatusFlags & SF_SD16_LISTO)
        {
          SD16Result = SD16Temp >> 4;              //cálculo del promedio
          StatusFlags &= ~SF_SD16_LISTO;
        }
        if (StatusFlags & SF_SD16_INICIO)
        {
          IniciaSigConversion();
          StatusFlags &= ~SF_SD16_INICIO;
        }
        AUX1 = 1 / CTE_A;
        AUX2 = 1 / AUX1 * (log(exp(1)) / log(10));
        A = 1 / AUX2 * (log(P_ATMOS / PA) / log(10)) ;
```

```
        CalConstA = A;
      }
      if (Secs == 5)//(Secs == 0x05)
      {
       ConstHrs = Hrs;
       ConstMins = Mins;
       ConstSecs = Secs;
       AlmacenaFlash();
       DetenerConversion();
      }

  }
}

   //-------------------------------------------------------------------------
void IniSis(void)
{

  WDTCTL = WDTPW + WDTHOLD;                  // detener el  WDT
  FLL_CTL0 |= XCAP18PF;                      // activar la carga de los capacitores externos del cristal

  BTCTL = BTDIV + BT_fCLK2_DIV128;           // habilitar la interrupciones del BT
  IE2 |= BTIE;

  P1OUT = 0x00;                               // P1OUTs = 0
  P1DIR = 0xff;                              //terminales sin usar se configuran como salidas
  P1IFG = 0x00;                               // restablecer las banderas del P1
  P5SEL = 0xff;                              // activar las terminales del LCD Rxx y COM
  P6OUT = 0x00;                               // P6OUTs = 0
  P6DIR = 0xff;                              // terminales sin usar se configuran como salidas
  P6SEL = 0x0f;                               // selección de los canales analógicos de SD16_A
}
//-------------------------------------------------------------------------
//INCREMENTO EN LAS VARIABLES DEL RTR
//
// CambioMins == 0 --> incrementan seg
// CambioMins == 1 --> incrementan min
//-------------------------------------------------------------------------

void RTR_Inc(unsigned int CambioMins)
{
  Secs = Secs+1;
  if ((Secs == 60) || CambioMins)
  {
    Secs = 0;
    Mins = Mins+1;
    if (Mins == 60)
    {
      Mins = 0;
      Hrs = Hrs+1;
      if (Hrs == 13)
        Hrs = 1;
    }
  }
}
//-------------------------------------------------------------------------
```

```
void AlmacenaFlash(void)
{

 FCTL2 = FWKEY + FSSEL1 + FN1;              // SMCLK/3 = ~333kHz
 FCTL3 = FWKEY;                          // se quita el candado
 FCTL1 = FWKEY + WRT;                       // habilitar la escritura
 i++;                                  //control de llenado de la memoria
 if (i==1)
 {
  ConstHrs_Flash = ConstHrs;               //almacenar todas la mediciones en la memoria Flash
  ConstMins_Flash = ConstMins;
  ConstSecs_Flash = ConstSecs;
  CalConstTM_Flash = CalConstTM;
  CalConstTA_Flash = CalConstTA;
  CalConstP_Flash = CalConstP;
  CalConstA_Flash = CalConstA;
 }

 else
 {
  j +=7;                               //almacenar en las nuevas localidades de memoria
  p1 = &ConstHrs_Flash;
  p1 +=(j);
  *p1 = ConstHrs;
  p1 = &ConstMins_Flash;
  p1 +=(j);
  *p1 = ConstMins;
  p1 = &ConstSecs_Flash;
  p1 +=(j);
  *p1 = ConstSecs;
  p1 = &CalConstTM_Flash;
  p1 +=(j);
  *p1= CalConstTM;
  p1 = &CalConstTA_Flash;
  p1 +=(j);
  *p1= CalConstTA;
  p1 = &CalConstP_Flash;
  p1 +=(j);
  *p1= CalConstP;
  p1 = &CalConstA_Flash;
  p1 +=(j);
  *p1= CalConstA;

 }
 //i==18 se llena la memoria
 if (i > 18)
 {
  i=0;
  j=0;
 FCTL1 = FWKEY + ERASE;                     //borrar la memoria Flash (segmento B)
*(unsigned int *)0x1000 = 0;
 FCTL1 = FWKEY + ERASE;
*(unsigned int *)0x1080 = 0;                //borrar la memoria Flash (segmento A)
 }
```

```
  FCTL1 = FWKEY;                          //colocar candado
  FCTL3 = FWKEY + LOCK;

}

void IniciaConversion(void)
{
  switch (ModoPrograma)                   // Preparar la conversión del SD16
  {
    case MP_MEDICION_TEMP :
      SD16CTL = SD16SSEL_2;               // Utilizar el ACLK para el SD16
      SD16INCTL0 = SD16GAIN_1 + SD16INCH_6;   // ganancia de 32x , canal A6
      SD16CCTL0 = SD16OSR_1024 + SD16DF + SD16IE;  // conversión continua, complemento a dos
      break;

     case MP_MEDICION_TA :
      P6OUT |= PUENTE_FUENTE;             // activar el puente de alimentación
      SD16CTL = SD16SSEL_2;
      SD16INCTL0 = SD16GAIN_1 + SD16INCH_1;   // ganancia de 1x , canal A1
      SD16CCTL0 = SD16BUF_1 + SD16OSR_1024 + SD16DF + SD16IE; // conversión continua,
complemento a dos
      break;
    case MP_MEDICION_P :
    case MP_MEDICION_A :
      P6OUT |= PUENTE_FUENTE;
      SD16CTL = SD16SSEL_2;
      SD16INCTL0 = SD16GAIN_32 + SD16INCH_0;  // ganancia de 32x , canal A0
      SD16CCTL0 = SD16BUF_1 + SD16OSR_1024 + SD16DF + SD16IE; // conversión continua,
complemento a dos
      break;
  }

}


//-------------------------------------------------------------------------------
void IniciaSigConversion(void)
{
  SD16Temp = 0;
  SD16TempCont = 0;

  switch (ModoPrograma)                   // comenzar la conversion del SD16
  {
    case MP_MEDICION_TEMP :
       SD16NumConversiones = 1;
      SD16CTL |= SD16VMIDON + SD16REFON;      // habilitar el Vref interno

      TAR = 65535 - 163;                  // retardar 5ms
      TACTL = TASSEL_1 + MC_2;            // detener el reloj A
      while (!(TACTL & TAIFG));
      TACTL = 0x00;
      break;
    case MP_MEDICION_TA :
      SD16NumConversiones = 16;           // promediar 16 mediciones
      break;
    case MP_MEDICION_P :
```

```
   case MP_MEDICION_A :
     SD16NumConversiones = 16;
     break;
  }

  SD16CCTL0 |= SD16SC;
}
//-------------------------------------------------------------------------
void DetenerConversion(void)
{
  SD16CCTL0 &= ~SD16SC;                 // Deshabilitar las conversiones

  switch (ModoPrograma)
  {
    case MP_MEDICION_TEMP :
      SD16CTL = 0x00;
      break;
    case MP_MEDICION_TA :
      P6OUT &= ~PUENTE_FUENTE;          // apagar el puente de alimentación
      break;
    case MP_MEDICION_P :
    case MP_MEDICION_A :
      P6OUT &= ~PUENTE_FUENTE;
      break;
  }
}
//-------------------------------------------------------------------------
//
//-------------------------------------------------------------------------
#pragma vector = SD16_VECTOR
__interrupt void SD16_ISR(void)
{
  SD16Temp += (long)(int)SD16MEM0;          // leer 15+1 bits

  if (++SD16TempCont >= SD16NumConversiones)      // son suficientes muestras?
  {
    SD16CCTL0 &= ~SD16SC;                       // deshabilitar las converiones
    SD16CTL &= ~(SD16VMIDON + SD16REFON);       // deshabilitar la referencia del SD16
    StatusFlags |= SF_SD16_LISTO;               // la conversión resultante esta lista
    __bic_SR_register_on_exit(LPM3_bits);       // fin de la ISR del SD16
  }
}
//-------------------------------------------------------------------------
//-------------------------------------------------------------------------
#pragma vector = BASICTIMER_VECTOR
__interrupt void BT_ISR(void)
{
  StatusFlags |= SF_BT_TIEMPO;                  // indica el proceso del BT
  __bic_SR_register_on_exit(LPM3_bits);         // fin de la ISR del BT
}
```

Apéndice B

A continuación se presentan las hojas técnicas de los sensores atmosféricos, pantalla de cristal líquido LCD y amplificadores operacionales utilizados en este proyecto de tesis.

100 kPa On-Chip Temperature Compensated & Calibrated Silicon Pressure Sensors

MPXM2102 SERIES

**COMPENSATED AND CALIBRATED PRESSURE SENSOR
0 TO 100 kPA (0 TO 14.5 psi)
40 mV FULL SCALE SPAN
(TYPICAL)**

The MPXM2102 device is a silicon piezoresistive pressure sensors providing a highly accurate and linear voltage output - directly proportional to the applied pressure. The sensor is a single, monolithic silicon diaphragm with the strain gauge and a thin-film resistor network integrated on-chip. The chip is laser trimmed for precise span and offset calibration and temperature compensation.

Features

- Temperature Compensated Over 0°C to +85°C
- Available in Easy-to-Use Tape & Reel
- Ratiometric to Supply Voltage
- Gauge Ported and Non Ported Options

Typical Applications

- Pump/Motor Controllers
- Robotics
- Level Indicators
- Medical Diagnostics
- Pressure Switching
- Barometers
- Altimeters

MPAK PACKAGES

**MPXM2102D/A
CASE 1320-02**

**MPXM2102GS/AS
CASE 1320A-02**

ORDERING INFORMATION					
Device Type	Options	Case No.	MPX Series Order No.	Packing Options	Device Marking
Non-ported	Gauge	1320	MPXM2102D	Rails	MPXM2102D
	Gauge	1320	MPXM2102DT1	Tape & Reel	MPXM2102D
	Absolute	1320	MPXM2102A	Rails	MPXM2102A
	Absolute	1320	MPXM2102AT1	Tape & Reel	MPXM2102A
Ported	Gauge, Axial Port	1320A	MPXM2102GS	Rails	MPXM2102G
	Gauge, Axial Port	1320A	MPXM2102GST1	Tape & Reel	MPXM2102G
	Absolute, Axial Port	1320A	MPXM2102AS	Rails	MPXM2102A
	Absolute, Axial Port	1320A	MPXM2102AST1	Tape & Reel	MPXM2102A

Pin Number			
1	Gnd	3	V_S
2	$+V_{out}$	4	$-V_{out}$

freescale semiconductor

Figure 1 shows a block diagram of the internal circuitry on the stand-alone pressure sensor chip.

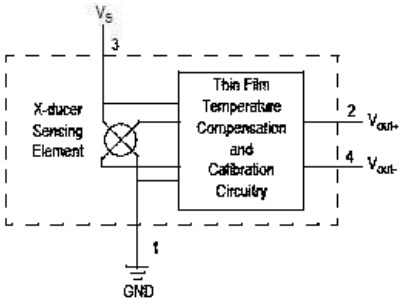


Figure 1. Temperature Compensated Pressure Sensor Schematic

VOLTAGE OUTPUT VERSUS APPLIED DIFFERENTIAL PRESSURE

The differential voltage output of the sensor is directly proportional to the differential pressure applied.

The output voltage of the differential or gauge sensor increases with increasing pressure applied to the pressure side relative to the vacuum side. Similarly, output voltage increases as increasing vacuum is applied to the vacuum side relative to the pressure side.

Table 1. Maximum Ratings[1]

Rating	Symbol	Value	Unit
Maximum Pressure	P_{max}	200	kPa
Storage Temperature	T_{stg}	-40 to +125	°C
Operating Temperature	T_A	-40 to +125	°C

1. Exposure beyond the specified limits may cause permanent damage or degradation to the device.

Table 2. Operating Characteristics (V_S = 10 Vdc, T_A = 25°C.)

Characteristic		Symbol	Min	Typ	Max	Unit
Pressure Range[1]		P_{OP}	0	—	100	kPa
Supply Voltage[2]		V_S	—	10	16	Vdc
Supply Current		I_o	—	6.0	-	mAdc
Full Scale Span[3]		V_{FSS}	38.5	40	41.5	mV
Offset[4]	MPXM2102D/G Series MPXM2102A Series	V_{off}	-1.0 -2.0	— —	1.0 2.0	mV
Sensitivity		ΔV/ΔP	—	0.4	—	mV/kPa
Linearity[5]	MPXM2102D/G Series MPXM2102A Series	— —	-0.6 -1.0	— —	0.4 1.0	$\%V_{FSS}$
Pressure Hysteresis[5] (0 to 100 kPa)		—	—	±0.1	—	$\%V_{FSS}$
Temperature Hysteresis[5] (-40°C to +125°C)		—	—	±0.5	—	$\%V_{FSS}$
Temperature Effect on Full Scale Span[5]		TCV_{FSS}	-2.0	—	2.0	$\%V_{FSS}$
Temperature Effect on Offset[5]		TCV_{off}	-1.0	—	1.0	mV
Input Impedance		Z_{in}	1000	—	2500	Ω
Output Impedance		Z_{out}	1400	—	3000	Ω
Response Time[6] (10% to 90%)		t_R	—	1.0	—	ms
Warm-Up		—	—	20	—	ms
Offset Stability[7]		—	—	±0.5	—	$\%V_{FSS}$

1. 1.0 kPa (kiloPascal) equals 0.145 psi.
2. Device is ratiometric within this specified excitation range. Operating the device above the specified excitation range may induce additional error due to device self-heating.
3. Full Scale Span (V_{FSS}) is defined as the algebraic difference between the output voltage at full rated pressure and the output voltage at the minimum rated pressure.
4. Offset (V_{off}) is defined as the output voltage at the minimum rated pressure.
5. Accuracy (error budget) consists of the following:
   - Linearity: Output deviation from a straight line relationship with pressure, using end point method, over the specified pressure range.
   - Temperature Hysteresis: Output deviation at any temperature within the operating temperature range, after the temperature is cycled to and from the minimum or maximum operating temperature points, with zero differential pressure applied.
   - Pressure Hysteresis: Output deviation at any pressure within the specified range, when this pressure is cycled to and from the minimum or maximum rated pressure, at 25°C.
   - TcSpan: Output deviation at full rated pressure over the temperature range of 0 to 85°C, relative to 25°C.
   - TcOffset: Output deviation with minimum rated pressure applied, over the temperature range of 0 to 85°C, relative to 25°C.
6. Response Time is defined as the time for the incremental change in the output to go from 10% to 90% of its final value when subjected to a specified step change in pressure.
7. Offset stability is the product's output deviation when subjected to 1000 hours of Pulsed Pressure, Temperature Cycling with Bias Test.

LINEARITY

Linearity refers to how well a transducer's output follows the equation: $V_{out} = V_{off} + \text{sensitivity} \times P$ over the operating pressure range. There are two basic methods for calculating nonlinearity: (1) end point straight line fit (see Figure 2) or (2) a least squares best line fit. While a least squares fit gives the "best case" linearity error (lower numerical value), the calculations required are burdensome.

Conversely, an end point fit will give the "worst case" error (often more desirable in error budget calculations) and the calculations are more straightforward for the user. The specified pressure sensor linearities are based on the end point straight line method measured at the midrange pressure.

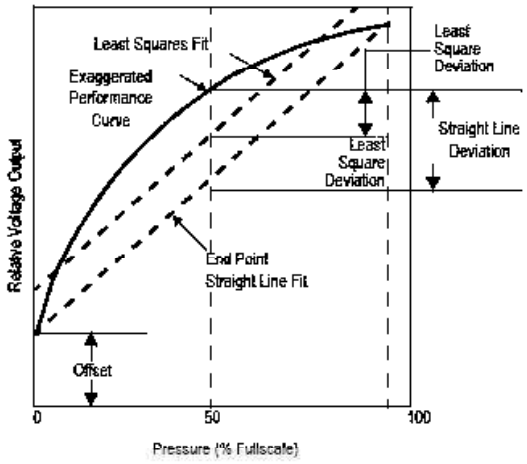


Figure 2. Linearity Specification Comparison

ON-CHIP TEMPERATURE COMPENSATION AND CALIBRATION

Figure 3 shows the minimum, maximum and typical output characteristics of the MPXM2120 series at 25°C. The output is directly proportional to the differential pressure and is essentially a straight line.

A silicone gel isolates the die surface and wire bonds from the environment, while allowing the pressure signal to be transmitted to the silicon diaphragm.

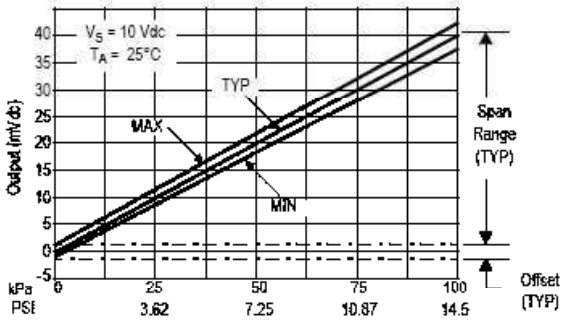
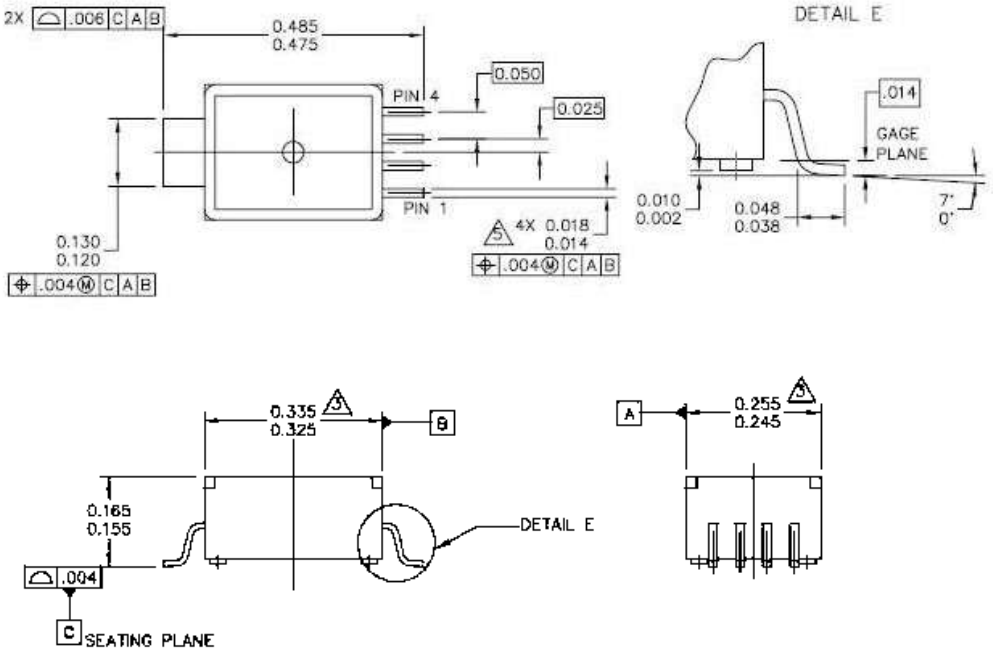


Figure 3. Output versus Pressure Differential

PACKAGE DIMENSIONS



© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.	MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE
TITLE: 5 LD M-PAC	DOCUMENT NO: 98ARH99088A	REV: B
	CASE NUMBER: 1320-02	22 JUL 2005
	STANDARD: NON-JEDEC	

PAGE 1 OF 2

CASE 1320-02
ISSUE B

PACKAGE DIMENSIONS

NOTES:

1. DIMENSIONS ARE IN INCHES.
2. INTERPRET DIMENSIONS AND TOLERANCES PER ASME Y14.5M–1994.
3. DIMENSION DOES NOT INCLUDE MOLD FLASH OR PROTRUSION. MOLD FLASH OR PROTRUSION SHALL NOT EXCEED .006" PER SIDE.
4. ALL VERTICAL SURFACES TO BE 5° MAXIMUM.
5. DIMENSION DOES NOT INCLUDE DAMBAR PROTRUSION. ALLOWABLE DAMBAR PROTRUSION SHALL BE .008 MAXIMUM.

PIN 1: GND
PIN 2: +Vout
PIN 3: Vs
PIN 4: −Vout

© FREESCALE SEMICONDUCTOR, INC. ALL RIGHTS RESERVED.	MECHANICAL OUTLINE	PRINT VERSION NOT TO SCALE
TITLE: 5 LD M–PAC	DOCUMENT NO: 98ARH99088A	REV: B
	CASE NUMBER: 1320–02	22 JUL 2005
	STANDARD: NON–JEDEC	

PAGE 2 OF 2

CASE 1320-02
ISSUE B

National Semiconductor

November 2000

LM35
Precision Centigrade Temperature Sensors

General Description

The LM35 series are precision integrated-circuit temperature sensors, whose output voltage is linearly proportional to the Celsius (Centigrade) temperature. The LM35 thus has an advantage over linear temperature sensors calibrated in ° Kelvin, as the user is not required to subtract a large constant voltage from its output to obtain convenient Centigrade scaling. The LM35 does not require any external calibration or trimming to provide typical accuracies of ±¼°C at room temperature and ±¾°C over a full −55 to +150°C temperature range. Low cost is assured by trimming and calibration at the wafer level. The LM35's low output impedance, linear output, and precise inherent calibration make interfacing to readout or control circuitry especially easy. It can be used with single power supplies, or with plus and minus supplies. As it draws only 60 µA from its supply, it has very low self-heating, less than 0.1°C in still air. The LM35 is rated to operate over a −55° to +150°C temperature range, while the LM35C is rated for a −40° to +110°C range (−10° with improved accuracy). The LM35 series is available packaged in hermetic TO-46 transistor packages, while the LM35C, LM35CA, and LM35D are also available in the plastic TO-92 transistor package. The LM35D is also available in an 8-lead surface mount small outline package and a plastic TO-220 package.

Features

- Calibrated directly in ° Celsius (Centigrade)
- Linear + 10.0 mV/°C scale factor
- 0.5°C accuracy guaranteeable (at +25°C)
- Rated for full −55° to +150°C range
- Suitable for remote applications
- Low cost due to wafer-level trimming
- Operates from 4 to 30 volts
- Less than 60 µA current drain
- Low self-heating, 0.08°C in still air
- Nonlinearity only ±¼°C typical
- Low impedance output, 0.1 Ω for 1 mA load

Typical Applications

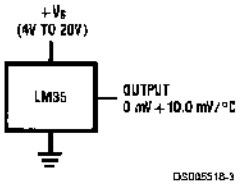
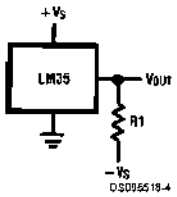


FIGURE 1. Basic Centigrade Temperature Sensor (+2°C to +150°C)

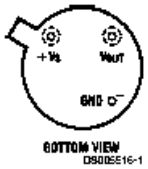


Choose $R_1 = -V_S/50\ \mu A$
V_{OUT} = +1,500 mV at +150°C
= +250 mV at +25°C
= −550 mV at −55°C

FIGURE 2. Full-Range Centigrade Temperature Sensor

Connection Diagrams

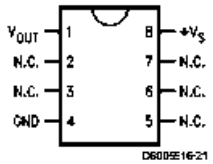
TO-46
Metal Can Package*



*Case is connected to negative pin (GND)

Order Number LM35H, LM35AH, LM35CH, LM35CAH or LM35DH
See NS Package Number H03H

SO-8
Small Outline Molded Package



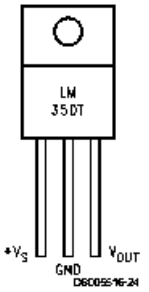
N.C. = No Connection

Top View
Order Number LM35DM
See NS Package Number M08A

TO-92
Plastic Package

Order Number LM35CZ, LM35CAZ or LM35DZ
See NS Package Number Z03A

TO-220
Plastic Package*



*Tab is connected to the negative pin (GND).

Note: The LM35DT pinout is different than the discontinued LM35DP.

Order Number LM35DT
See NS Package Number TA03F

Absolute Maximum Ratings (Note 10)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage	+35V to −0.2V
Output Voltage	+6V to −1.0V
Output Current	10 mA
Storage Temp.;	
TO-46 Package,	−60°C to +180°C
TO-92 Package,	−60°C to +150°C
SO-8 Package,	−65°C to +150°C
TO-220 Package,	−65°C to +150°C
Lead Temp.:	
TO-46 Package, (Soldering, 10 seconds)	300°C
TO-92 and TO-220 Package, (Soldering, 10 seconds)	260°C
SO Package (Note 12)	
Vapor Phase (60 seconds)	215°C
Infrared (15 seconds)	220°C
ESD Susceptibility (Note 11)	2500V
Specified Operating Temperature Range: T_{MIN} to T_{MAX} (Note 2)	
LM35, LM35A	−55°C to +150°C
LM35C, LM35CA	−40°C to +110°C
LM35D	0°C to +100°C

Electrical Characteristics

(Notes 1, 6)

Parameter	Conditions	LM35A			LM35CA			Units (Max.)
		Typical	Tested Limit (Note 4)	Design Limit (Note 5)	Typical	Tested Limit (Note 4)	Design Limit (Note 5)	
Accuracy (Note 7)	T_A=+25°C	±0.2	±0.5		±0.2	±0.5		°C
	T_A=−10°C	±0.3			±0.3		±1.0	°C
	T_A=T_{MAX}	±0.4	±1.0		±0.4	±1.0		°C
	T_A=T_{MIN}	±0.4	±1.0		±0.4		±1.5	°C
Nonlinearity (Note 8)	T_{MIN}≤T_A≤T_{MAX}	±0.18		±0.35	±0.15		±0.3	°C
Sensor Gain (Average Slope)	T_{MIN}≤T_A≤T_{MAX}	+10.0	+9.9, +10.1		+10.0		+9.9, +10.1	mV/°C
Load Regulation (Note 3) 0≤I_L≤1 mA	T_A=+25°C	±0.4	±1.0		±0.4	±1.0		mV/mA
	T_{MIN}≤T_A≤T_{MAX}	±0.5		±3.0	±0.5		±3.0	mV/mA
Line Regulation (Note 3)	T_A=+25°C	±0.01	±0.05		±0.01	±0.05		mV/V
	4V≤V_S≤30V	±0.02		±0.1	±0.02		±0.1	mV/V
Quiescent Current (Note 9)	V_S=+5V, +25°C	56	67		56	67		µA
	V_S=+5V	105		131	91		114	µA
	V_S=+30V, +25°C	56.2	68		56.2	68		µA
	V_S=+30V	105.5		133	91.5		116	µA
Change of Quiescent Current (Note 3)	4V≤V_S≤30V, +25°C	0.2	1.0		0.2	1.0		µA
	4V≤V_S≤30V	0.5		2.0	0.5		2.0	µA
Temperature Coefficient of Quiescent Current		+0.39		+0.5	+0.39		+0.5	µA/°C
Minimum Temperature for Rated Accuracy	In circuit of *Figure 1*, I_L=0	+1.5		+2.0	+1.5		+2.0	°C
Long Term Stability	T_J=T_{MAX}, for 1000 hours	±0.08			±0.08			°C

Electrical Characteristics

(Notes 1, 6)

Parameter	Conditions	LM35			LM35C, LM35D			Units (Max.)
		Typical	Tested Limit (Note 4)	Design Limit (Note 5)	Typical	Tested Limit (Note 4)	Design Limit (Note 5)	
Accuracy,	T_A=+25°C	±0.4	±1.0		±0.4	±1.0		°C
LM35, LM35C	T_A=−10°C	±0.5			±0.5		±1.5	°C
(Note 7)	T_A=T_{MAX}	±0.8	±1.5		±0.8		±1.5	°C
	T_A=T_{MIN}	±0.8		±1.5	±0.8		±2.0	°C
Accuracy, LM35D	T_A=+25°C				±0.6	±1.5		°C
(Note 7)	T_A=T_{MAX}				±0.9		±2.0	°C
	T_A=T_{MIN}				±0.9		±2.0	°C
Nonlinearity (Note 8)	T_{MIN}≤T_A≤T_{MAX}	±0.3		±0.5	±0.2		±0.5	°C
Sensor Gain (Average Slope)	T_{MIN}≤T_A≤T_{MAX}	+10.0	+9.8, +10.2		+10.0		+9.8, +10.2	mV/°C
Load Regulation	T_A=+25°C	±0.4	±2.0		±0.4	±2.0		mV/mA
(Note 3) 0≤I_L≤1 mA	T_{MIN}≤T_A≤T_{MAX}	±0.5		±5.0	±0.5		±5.0	mV/mA
Line Regulation	T_A=+25°C	±0.01	±0.1		±0.01	±0.1		mV/V
(Note 3)	4V≤V_S≤30V	±0.02		±0.2	±0.02		±0.2	mV/V
Quiescent Current	V_S=+5V, +25°C	56	80		56	80		µA
(Note 9)	V_S=+5V	105		158	91		138	µA
	V_S=+30V, +25°C	56.2	82		56.2	82		µA
	V_S=+30V	105.5		161	91.5		141	µA
Change of	4V≤V_S≤30V, +25°C	0.2	2.0		0.2	2.0		µA
Quiescent Current	4V≤V_S≤30V	0.5		3.0	0.5		3.0	µA
(Note 3)								
Temperature Coefficient of Quiescent Current		+0.39		+0.7	+0.39		+0.7	µA/°C
Minimum Temperature for Rated Accuracy	In circuit of *Figure 1*, I_L=0	+1.5		+2.0	+1.5		+2.0	°C
Long Term Stability	T_J=T_{MAX}, for 1000 hours	±0.08			±0.08			°C

Note 1: Unless otherwise noted, these specifications apply: −55°C≤T_J≤+150°C for the LM35 and LM35A; −40°≤T_J≤+110°C for the LM35C and LM35CA; and 0°≤T_J≤+100°C for the LM35D. V_S=+5Vdc and I_{LOAD}=50 µA, in the circuit of *Figure 2*. These specifications also apply from +2°C to T_{MAX} in the circuit of *Figure 1*. Specifications in **boldface** apply over the full rated temperature range.

Note 2: Thermal resistance of the TO-46 package is 400°C/W, junction to ambient, and 24°C/W junction to case. Thermal resistance of the TO-92 package is 180°C/W junction to ambient. Thermal resistance of the small outline molded package is 220°C/W junction to ambient. Thermal resistance of the TO-220 package is 90°C/W junction to ambient. For additional thermal resistance information see table in the Applications section.

Note 3: Regulation is measured at constant junction temperature, using pulse testing with a low duty cycle. Changes in output due to heating effects can be computed by multiplying the internal dissipation by the thermal resistance.

Note 4: Tested Limits are guaranteed and 100% tested in production.

Note 5: Design Limits are guaranteed (but not 100% production tested) over the indicated temperature and supply voltage ranges. These limits are not used to calculate outgoing quality levels.

Note 6: Specifications in boldface apply over the full rated temperature range.

Note 7: Accuracy is defined as the error between the output voltage and 10mv/°C times the device's case temperature, at specified conditions of voltage, current, and temperature (expressed in °C).

Note 8: Nonlinearity is defined as the deviation of the output-voltage-versus-temperature curve from the best-fit straight line, over the device's rated temperature range.

Note 9: Quiescent current is defined in the circuit of *Figure 1*.

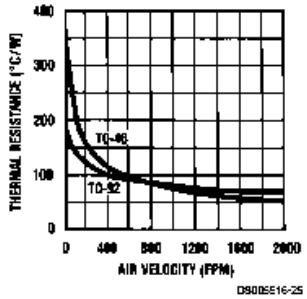
Note 10: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. DC and AC electrical specifications do not apply when operating the device beyond its rated operating conditions. See Note 1.

Note 11: Human body model, 100 pF discharged through a 1.5 kΩ resistor.

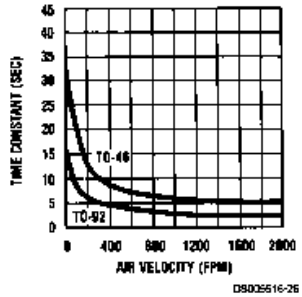
Note 12: See AN-450 "Surface Mounting Methods and Their Effect on Product Reliability" or the section titled "Surface Mount" found in a current National Semiconductor Linear Data Book for other methods of soldering surface mount devices.

Typical Performance Characteristics

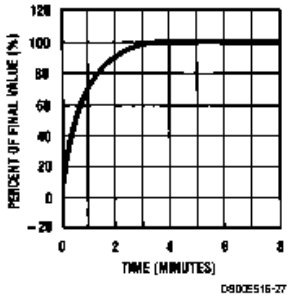
Thermal Resistance
Junction to Air



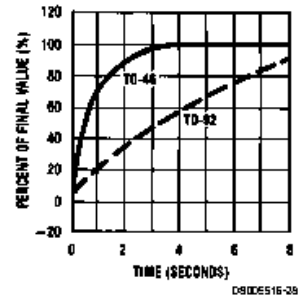
Thermal Time Constant



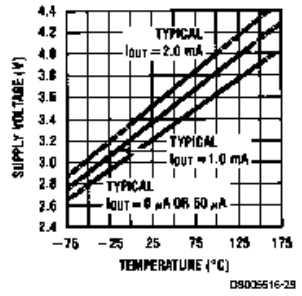
Thermal Response
in Still Air



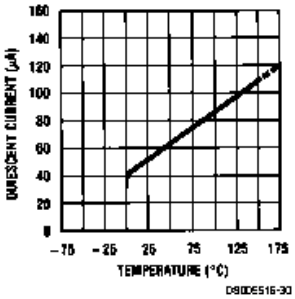
Thermal Response in
Stirred Oil Bath



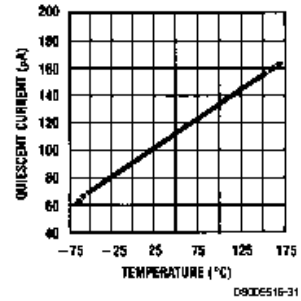
Minimum Supply
Voltage vs. Temperature



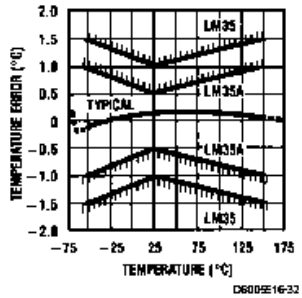
Quiescent Current
vs. Temperature
(In Circuit of *Figure 1*.)



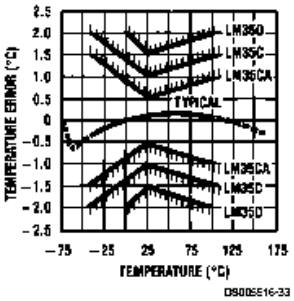
Quiescent Current
vs. Temperature
(In Circuit of *Figure 2*.)



Accuracy vs. Temperature
(Guaranteed)

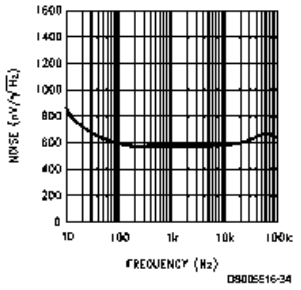


Accuracy vs. Temperature
(Guaranteed)

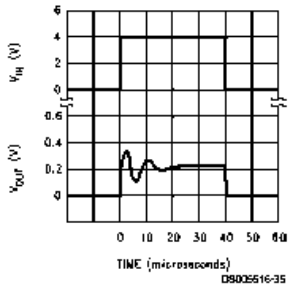


Typical Performance Characteristics (Continued)

Noise Voltage



Start-Up Response



Applications

The LM35 can be applied easily in the same way as other integrated-circuit temperature sensors. It can be glued or cemented to a surface and its temperature will be within about 0.01°C of the surface temperature.

This presumes that the ambient air temperature is almost the same as the surface temperature; if the air temperature were much higher or lower than the surface temperature, the actual temperature of the LM35 die would be at an intermediate temperature between the surface temperature and the air temperature. This is especially true for the TO-92 plastic package, where the copper leads are the principal thermal path to carry heat into the device, so its temperature might be closer to the air temperature than to the surface temperature.

To minimize this problem, be sure that the wiring to the LM35, as it leaves the device, is held at the same temperature as the surface of interest. The easiest way to do this is to cover up these wires with a bead of epoxy which will insure that the leads and wires are all at the same temperature as the surface, and that the LM35 die's temperature will not be affected by the air temperature.

The TO-46 metal package can also be soldered to a metal surface or pipe without damage. Of course, in that case the V− terminal of the circuit will be grounded to that metal. Alternatively, the LM35 can be mounted inside a sealed-end metal tube, and can then be dipped into a bath or screwed into a threaded hole in a tank. As with any IC, the LM35 and accompanying wiring and circuits must be kept insulated and dry, to avoid leakage and corrosion. This is especially true if the circuit may operate at cold temperatures where condensation can occur. Printed-circuit coatings and varnishes such as Humiseal and epoxy paints or dips are often used to insure that moisture cannot corrode the LM35 or its connections.

These devices are sometimes soldered to a small light-weight heat fin, to decrease the thermal time constant and speed up the response in slowly-moving air. On the other hand, a small thermal mass may be added to the sensor, to give the steadiest reading despite small deviations in the air temperature.

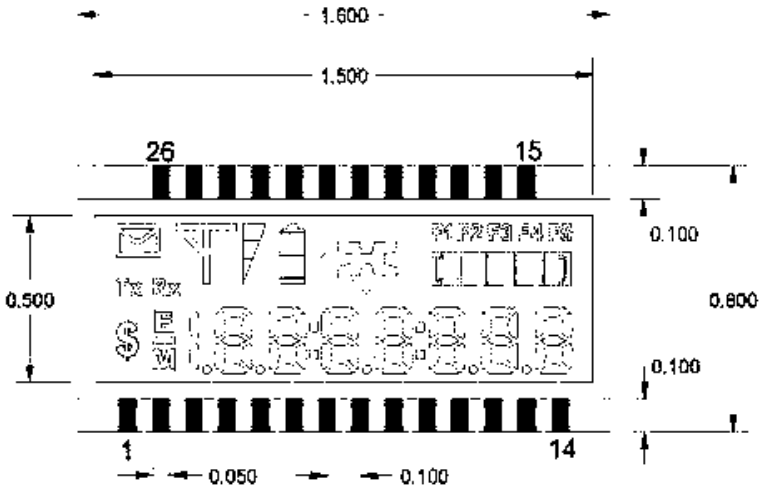
Temperature Rise of LM35 Due To Self-heating (Thermal Resistance, θ_{JA})

	TO-46, no heat sink	TO-46*, small heat fin	TO-92, no heat sink	TO-92**, small heat fin	SO-8 no heat sink	SO-8** small heat fin	TO-220 no heat sink
Still air	400°C/W	100°C/W	180°C/W	140°C/W	220°C/W	110°C/W	90°C/W
Moving air	100°C/W	40°C/W	90°C/W	70°C/W	105°C/W	90°C/W	26°C/W
Still oil	100°C/W	40°C/W	90°C/W	70°C/W			
Stirred oil	50°C/W	30°C/W	45°C/W	40°C/W			
(Clamped to metal, infinite heat sink)	(24°C/W)				(55°C/W)		

*Wakefield type 201, or 1" disc of 0.020" sheet brass, soldered to case, or similar.
**TO-92 and SO-8 packages glued and leads soldered to 1" square of 1/16" printed circuit board with 2 oz. foil or similar.

SoftBaugh
4080 McGinnis Ferry Road
Suite 604
Alpharetta, GA 30005
www.softbaugh.com

SBLCDA4



Overview

Our advanced SBLCDA4 is compatible with all MSP430s featuring LCD drive. The SBLCDA4 offers the following features:

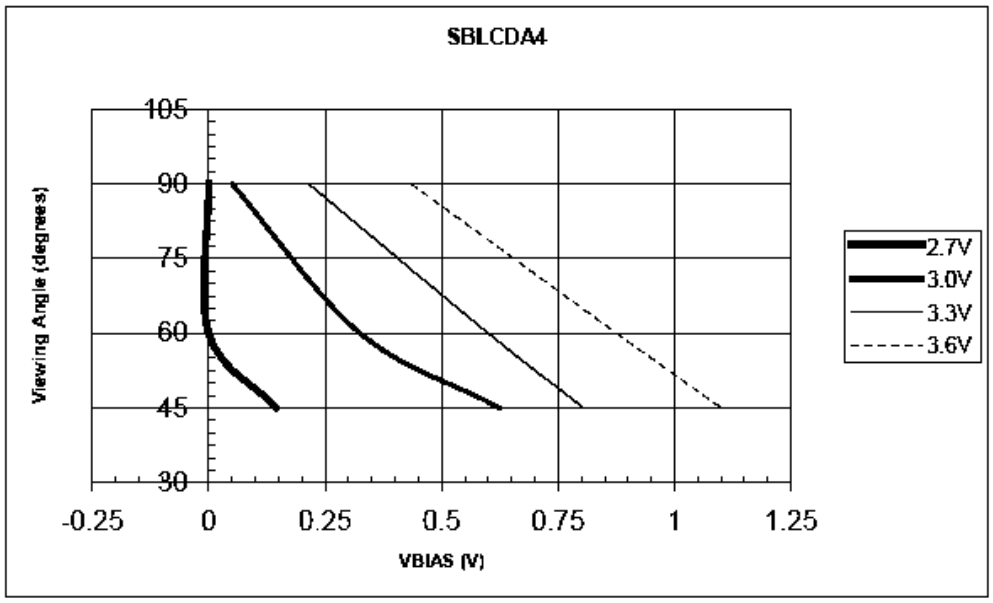
- 2.7v to 3.6v operation directly connected to the MSP430 LCD drive
- 4-mux operation
- 7.1 seven-segment with minus symbology allows versatile text
- Envelope, $, Error, and Memory symbols
- Arrows left, up, right, and down
- Battery with two-segment meter
- Antenna with three-segment meter, plus Tx/Rx symbols
- Colons for HH:MM:SS operation, plus five decimal options
- Progress bar for convenient user feedback
- Five function symbols
- 6 o'clock viewing angle
- Low-cost bias circuit allows adjustment for viewing angle, contrast, and temperature
- Operating temperature –20C/50C

Typical Operation

The following graphs illustrate the SBLCDA4 viewing angles for the range of V_{CC} from 2.7v to 3.6v. The test was performed on an MSP430F413. The voltage measured was on pin 40 (R03) of the MSP430F413. This is the input port of the lowest analog LCD level (V5). The voltage was controlled by a 200kΩ potentiometer.

Due to the wide viewing angle of the SBLCDA4 your bias voltage may vary from those shown in Figure 1.

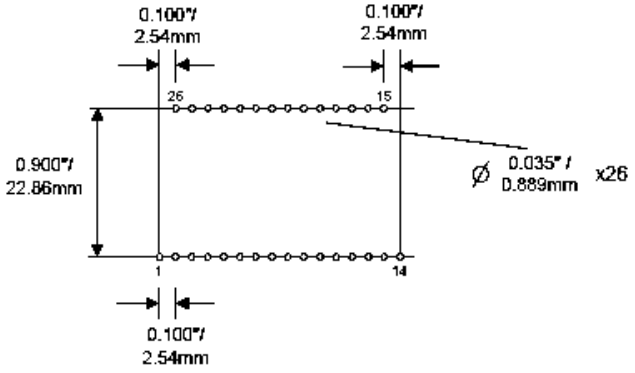
Figure 1—SBLCDA4 Viewing Angle

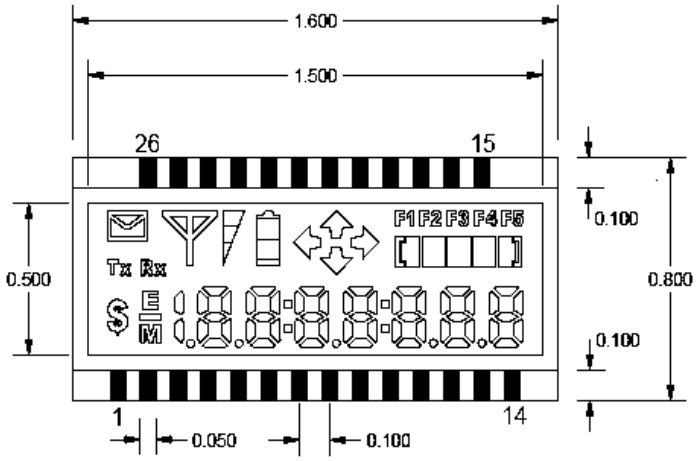


SBLCDA4 Segment Mapping

PIN	COM3	COM2	COM1	COM0		COM3	COM2	COM1	COM0	PIN
1	7F	7G	7E	DP7						
2	7A	7B	7C	7D		DOL	ERR	MINUS	MEM	26
3	6F	6G	6E	DP6		ENV	TX	RX	8BC	25
4	6A	6B	6C	6D		ANT	A2	A1	A0	24
5	5F	5G	5E	COL5		BT	B1	B0	BB	23
6	5A	5B	5C	5D		AU	AR	AD	AL	22
7	4F	4G	4E	DP4		PL	P0	P1	P2	21
8	4A	4B	4C	4D		F1	F2	F3	F4	20
9	3F	3G	3E	COL3		F5	PR	P4	P3	19
10	3A	3B	3C	3D					COM0	18
11	2F	2G	2E	DP2				COM1		17
12	2A	2B	2C	2D			COM2			16
13	1F	1G	1E	DP1		COM3				15
14	1A	1B	1C	1D						

Recommended SBLCDA4 Footprint



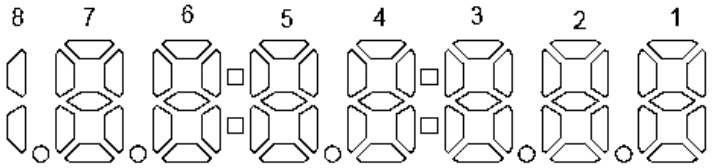


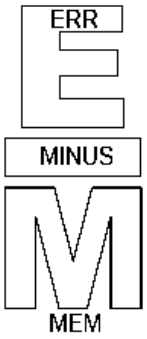
SBLCDA4

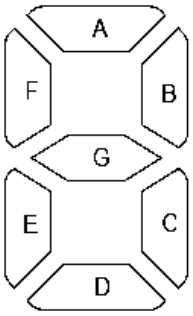
Rev 1.0.0 2 June 2003

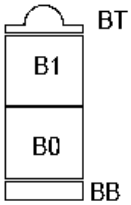
SoftBaugh, Inc.
4080 McGinnis Ferry Road
Suite 604
Alpharetta, GA 30005
www.softbaugh.com
(770) 772-8111
(770) 772-9030 (fax)

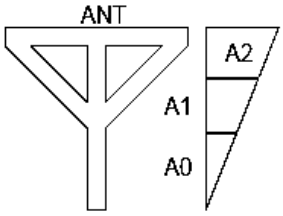
Pins both sides, 0.100" pitch, approx. 20mil wide at PCB, approx 0.250" overall
Glass is 0.8" wide, pins are 0.9" apart at PCB (0.050" knee on each side)

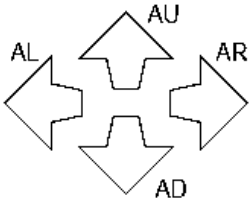


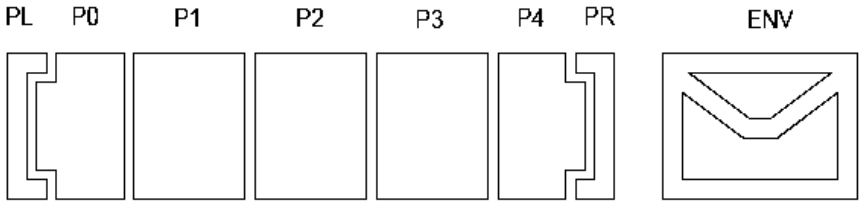












National Semiconductor

May 2000

LF155/LF156/LF355/LF356/LF357 JFET Input Operational Amplifiers

General Description

These are the first monolithic JFET input operational amplifiers to incorporate well matched, high voltage JFETs on the same chip with standard bipolar transistors (BI-FET™ Technology). These amplifiers feature low input bias and offset currents/low offset voltage and offset voltage drift, coupled with offset adjust which does not degrade drift or common-mode rejection. The devices are also designed for high slew rate, wide bandwidth, extremely fast settling time, low voltage and current noise and a low 1/f noise corner.

Advantages

- Replace expensive hybrid and module FET op amps
- Rugged JFETs allow blow-out free handling compared with MOSFET input devices
- Excellent for low noise applications using either high or low source impedance—very low 1/f corner
- Offset adjust does not degrade drift or common-mode rejection as in most monolithic amplifiers
- New output stage allows use of large capacitive loads (5,000 pF) without stability problems
- Internal compensation and large differential input voltage capability

Applications

- Precision high speed integrators
- Fast D/A and A/D converters
- High impedance buffers
- Wideband, low noise, low drift amplifiers
- Logarithmic amplifiers
- Photocell amplifiers
- Sample and Hold circuits

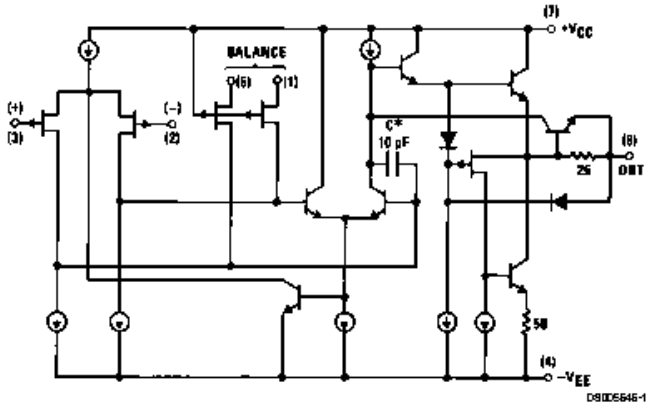
Common Features

- Low input bias current: 30pA
- Low Input Offset Current: 3pA
- High input impedance: $10^{12}\Omega$
- Low input noise current: $0.01\ pA/\sqrt{Hz}$
- High common-mode rejection ratio: 100 dB
- Large dc voltage gain: 106 dB

Uncommon Features

	LF155/ LF355	LF156/ LF356	LF357 (A_V=5)	Units
■ Extremely fast settling time to 0.01%	4	1.5	1.5	µs
■ Fast slew rate	5	12	50	V/µs
■ Wide gain bandwidth	2.5	5	20	MHz
■ Low input noise voltage	20	12	12	$nV/\sqrt{Hz}$

Simplified Schematic



*3 pF in LF357 series.

BI-FET™, BI-FET II™ are trademarks of National Semiconductor Corporation.

Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, contact the National Semiconductor Sales Office/Distributors for availability and specifications.

	LF155/6	LF356B	LF355/6/7
Supply Voltage	±22V	±22V	±18V
Differential Input Voltage	±40V	±40V	±30V
Input Voltage Range (Note 2)	±20V	±20V	±16V
Output Short Circuit Duration	Continuous	Continuous	Continuous
T_{JMAX}			
H-Package	150°C	115°C	115°C
N-Package		100°C	100°C
M-Package		100°C	100°C
Power Dissipation at T_A = 25°C (Notes 1, 8)			
H-Package (Still Air)	560 mW	400 mW	400 mW
H-Package (400 LF/Min Air Flow)	1200 mW	1000 mW	1000 mW
N-Package		670 mW	670 mW
M-Package		380 mW	380 mW
Thermal Resistance (Typical) θ_{JA}			
H-Package (Still Air)	160°C/W	160°C/W	160°C/W
H-Package (400 LF/Min Air Flow)	65°C/W	65°C/W	65°C/W
N-Package		130°C/W	130°C/W
M-Package		195°C/W	195°C/W
(Typical) θ_{JC}			
H-Package	23°C/W	23°C/W	23°C/W
Storage Temperature Range	−65°C to +150°C	−65°C to +150°C	−65°C to +150°C
Soldering Information (Lead Temp.)			
Metal Can Package			
Soldering (10 sec.)	300°C	300°C	300°C
Dual-In-Line Package			
Soldering (10 sec.)	260°C	260°C	260°C
Small Outline Package			
Vapor Phase (60 sec.)		215°C	215°C
Infrared (15 sec.)		220°C	220°C
See AN-450 "Surface Mounting Methods and Their Effect on Product Reliability" for other methods of soldering surface mount devices.			
ESD tolerance			
(100 pF discharged through 1.5 kΩ)	1000V	1000V	1000V

DC Electrical Characteristics

(Note 3)

Symbol	Parameter	Conditions	LF155/6			LF356B			LF355/6/7			Units
			Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
V_{OS}	Input Offset Voltage	R_S=50Ω, T_A=25°C		3	5		3	5		3	10	mV
		Over Temperature			7			6.5			13	mV
$\Delta V_{OS}/\Delta T$	Average TC of Input Offset Voltage	R_S=50Ω		5			5			5		µV/°C
$\Delta TC/\Delta V_{OS}$	Change in Average TC with V_{OS} Adjust	R_S=50Ω, (Note 4)		0.5			0.5			0.5		µV/°C per mV
I_{OS}	Input Offset Current	T_J=25°C, (Notes 3, 5)		3	20		3	20		3	50	pA
		$T_J \leq T_{HIGH}$			20			1			2	nA
I_B	Input Bias Current	T_J=25°C, (Notes 3, 5)		30	100		30	100		30	200	pA
		$T_J \leq T_{HIGH}$			50			5			8	nA
R_{IN}	Input Resistance	T_J=25°C		10^{12}			10^{12}			10^{12}		Ω

DC Electrical Characteristics (Continued)

(Note 3)

Symbol	Parameter	Conditions	LF155/6			LF356B			LF355/6/7			Units
			Min	Typ	Max	Min	Typ	Max	Min	Typ	Max	
A_{VOL}	Large Signal Voltage Gain	V_S=±15V, T_A=25°C V_O=±10V, R_L=2k Over Temperature	50 25	200		50 25	200		25 15	200		V/mV V/mV
V_O	Output Voltage Swing	V_S=±15V, R_L=10k V_S=±15V, R_L=2k	±12 ±10	±13 ±12		±12 ±10	±13 ±12		±12 ±10	±13 ±12		V V
V_{CM}	Input Common-Mode Voltage Range	V_S=±15V	±11	+15.1 −12		±11	±15.1 −12		+10	+15.1 −12		V V
CMRR	Common-Mode Rejection Ratio		85	100		85	100		80	100		dB
PSRR	Supply Voltage Rejection Ratio	(Note 6)	85	100		85	100		80	100		dB

DC Electrical Characteristics

$T_A = T_J = 25°C$, $V_S = ±15V$

Parameter	LF155		LF355		LF156/356B		LF356		LF357		Units
	Typ	Max	Typ	Max	Typ	Max	Typ	Max	Typ	Max	
Supply Current	2	4	2	4	5	7	5	10	5	10	mA

AC Electrical Characteristics

$T_A = T_J = 25°C$, $V_S = ±15V$

Symbol	Parameter	Conditions	LF155/355	LF156/356B	LF156/356/ LF356B	LF357	Units
			Typ	Min	Typ	Typ	
SR	Slew Rate	LF155/6: A_V=1, LF357: A_V=5	5	7.5	12	 50	V/µs V/µs
GBW	Gain Bandwidth Product		2.5		5	20	MHz
t_s	Settling Time to 0.01%	(Note 7)	4		1.5	1.5	µs
e_n	Equivalent Input Noise Voltage	R_S=100Ω f=100 Hz f=1000 Hz	 25 20		 15 12	 15 12	 nV/$\sqrt{Hz}$ nV/$\sqrt{Hz}$
i_n	Equivalent Input Current Noise	f=100 Hz f=1000 Hz	0.01 0.01		0.01 0.01	0.01 0.01	pA/$\sqrt{Hz}$ pA/$\sqrt{Hz}$
C_{IN}	Input Capacitance		3		3	3	pF

Notes for Electrical Characteristics

Note 1: The maximum power dissipation for these devices must be derated at elevated temperatures and is dictated by T_{JMAX}, θ_{JA}, and the ambient temperature, T_A. The maximum available power dissipation at any temperature is $P_d=(T_{JMAX}-T_A)/\theta_{JA}$ or the 25°C P_{dMAX}, whichever is less.

Note 2: Unless otherwise specified the absolute maximum negative input voltage is equal to the negative power supply voltage.

Note 3: Unless otherwise stated, these test conditions apply:

	LF155/156	LF356B	LF355/6/7
Supply Voltage, V_S	±15V≤V_S≤±20V	±15V≤V_S≤±20V	V_S=±15V
T_A	−55°C≤T_A≤+125°C	0°C≤T_A≤+70°C	0°C≤T_A≤+70°C
T_{HIGH}	+125°C	+70°C	+70°C

and V_{OS}, I_B and I_{OS} are measured at V_{CM}=0.

Note 4: The Temperature Coefficient of the adjusted input offset voltage changes only a small amount (0.5µV/°C typically) for each mV of adjustment from its original unadjusted value. Common-mode rejection and open loop voltage gain are also unaffected by offset adjustment.

Notes for Electrical Characteristics (Continued)

Note 5: The input bias currents are junction leakage currents which approximately double for every 10°C increase in the junction temperature, T_J. Due to limited production test time, the input bias currents measured are correlated to junction temperature. In normal operation the junction temperature rises above the ambient temperature as a result of internal power dissipation, Pd. $T_J = T_A + \theta_{JA}$ Pd where θ_{JA} is the thermal resistance from junction to ambient. Use of a heat sink is recommended if input bias current is to be kept to a minimum.

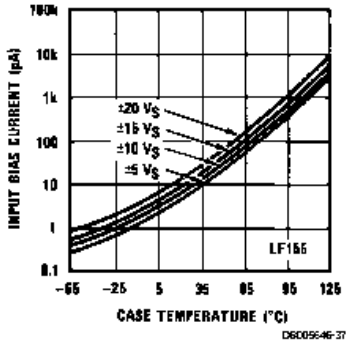
Note 6: Supply Voltage Rejection is measured for both supply magnitudes increasing or decreasing simultaneously, in accordance with common practice.

Note 7: Settling time is defined here, for a unity gain inverter connection using 2 kΩ resistors for the LF155/6. It is the time required for the error voltage (the voltage at the inverting input pin on the amplifier) to settle to within 0.01% of its final value from the time a 10V step input is applied to the inverter. For the LF357, $A_V = -5$, the feedback resistor from output to input is 2 kΩ and the output step is 10V (See Settling Time Test Circuit).

Note 8: Max. Power Dissipation is defined by the package characteristics. Operating the part near the Max. Power Dissipation may cause the part to operate outside guaranteed limits.

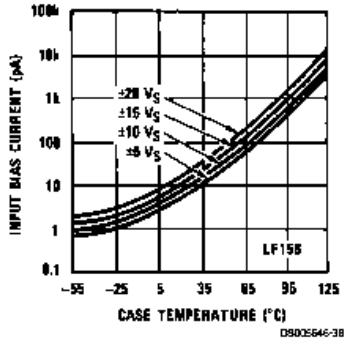
Typical DC Performance Characteristics Curves are for LF155 and LF156 unless otherwise specified.

Input Bias Current



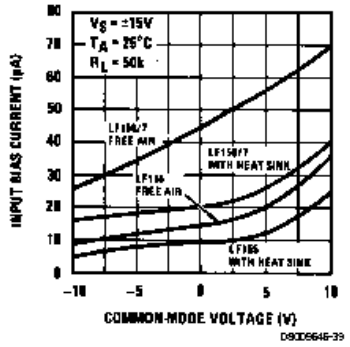
D6005646-37

Input Bias Current



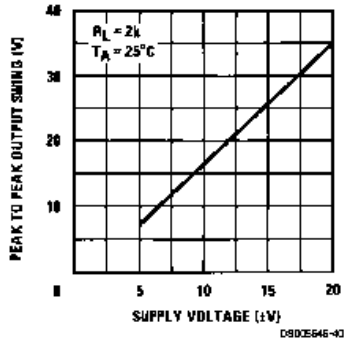
D6005646-38

Input Bias Current



D6005646-39

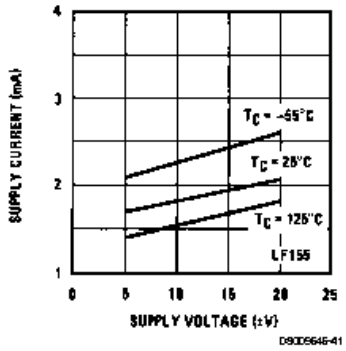
Voltage Swing



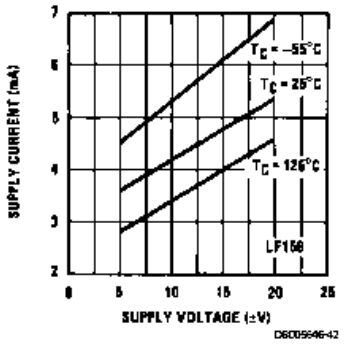
D6005646-40

Typical DC Performance Characteristics Curves are for LF155 and LF156 unless otherwise specified. (Continued)

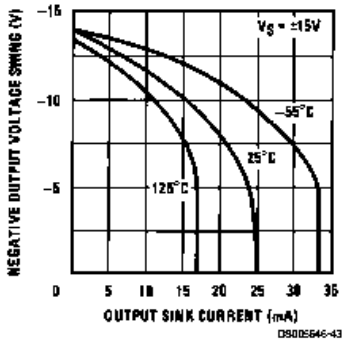
Supply Current



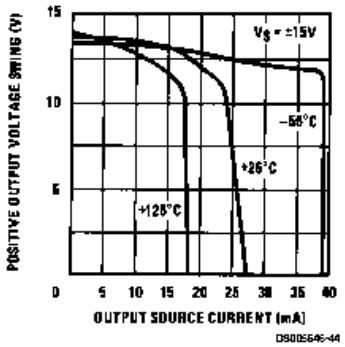
Supply Current



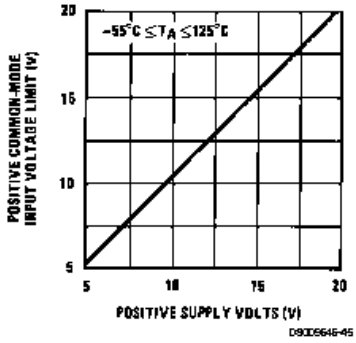
Negative Current Limit



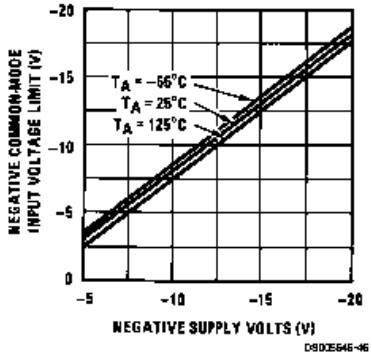
Positive Current Limit



Positive Common-Mode Input Voltage Limit

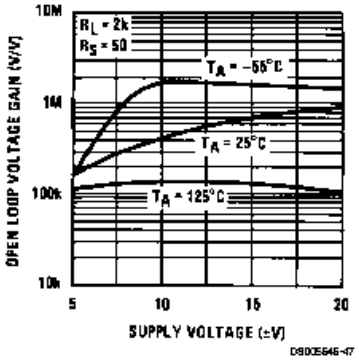


Negative Common-Mode Input Voltage Limit

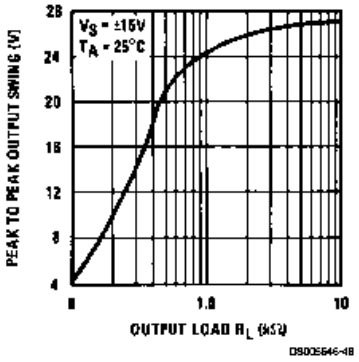


Typical DC Performance Characteristics Curves are for LF155 and LF156 unless otherwise specified. (Continued)

Open Loop Voltage Gain

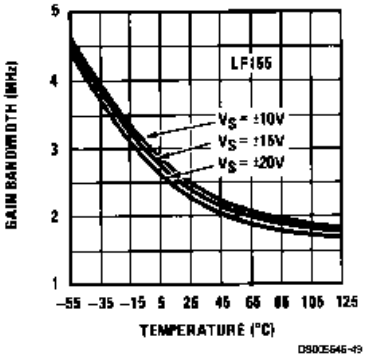


Output Voltage Swing

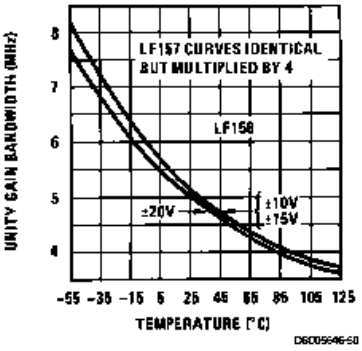


Typical AC Performance Characteristics

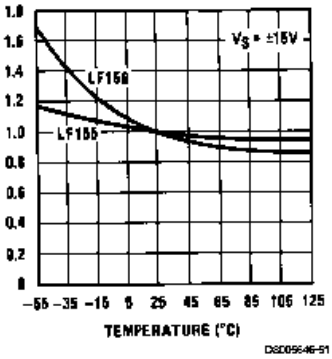
Gain Bandwidth



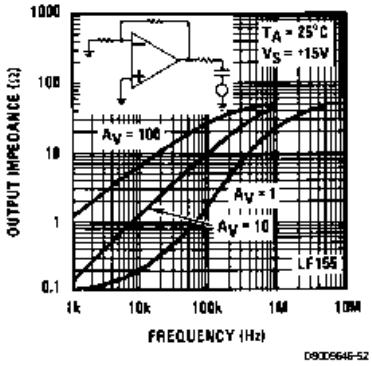
Gain Bandwidth



Normalized Slew Rate

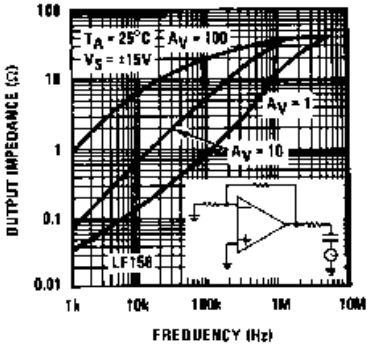


Output Impedance

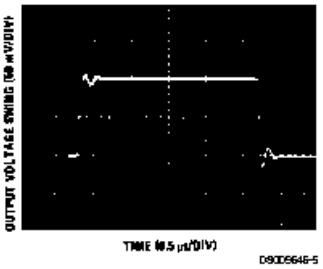


Typical AC Performance Characteristics (Continued)

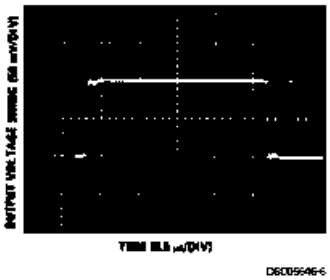
Output Impedance



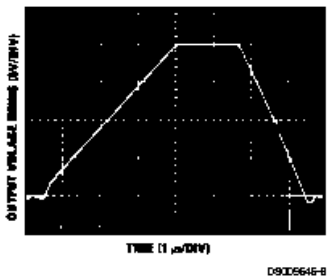
LF155 Small Signal Pulse Response, A_V = +1



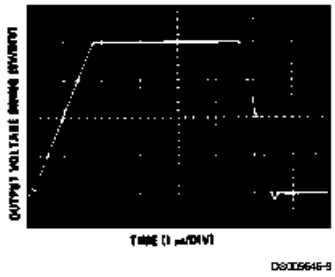
LF156 Small Signal Pulse Response, A_V = +1



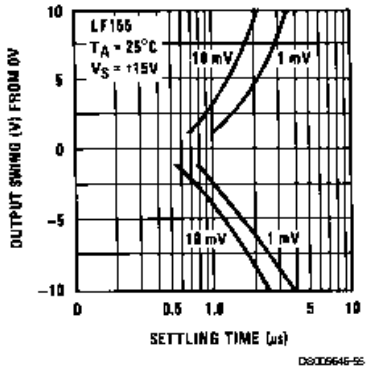
LF155 Large Signal Pulse Response, A_V = +1



LF156 Large Signal Puls Response, A_V = +1

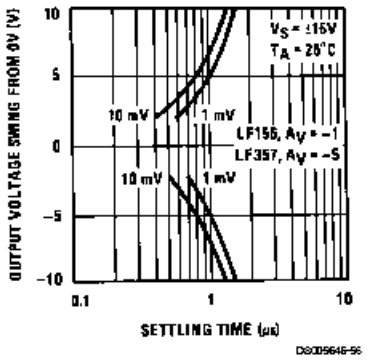


Inverter Settling Time



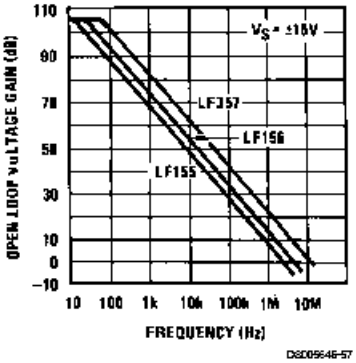
Typical AC Performance Characteristics (Continued)

Inverter Settling Time



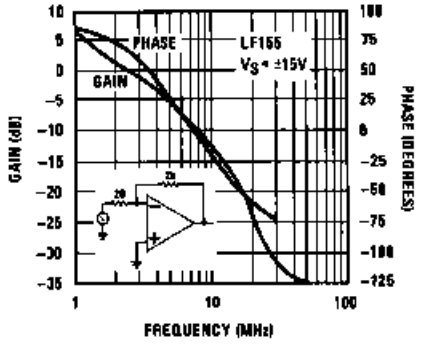
DS005646-56

Open Loop Frequency Response



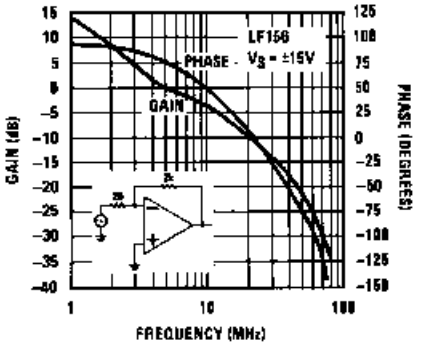
DS005646-57

Bode Plot



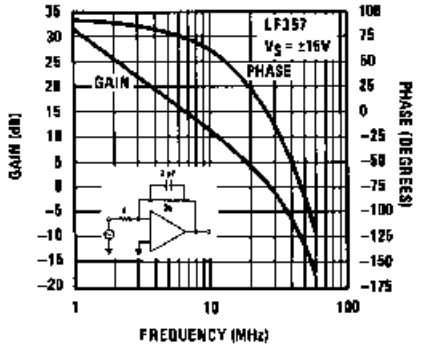
DS005646-58

Bode Plot



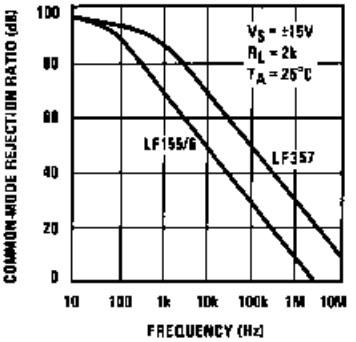
DS005646-59

Bode Plot



DS005646-60

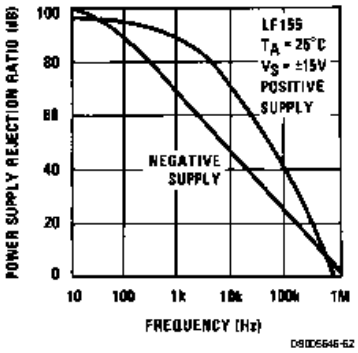
Common-Mode Rejection Ratio



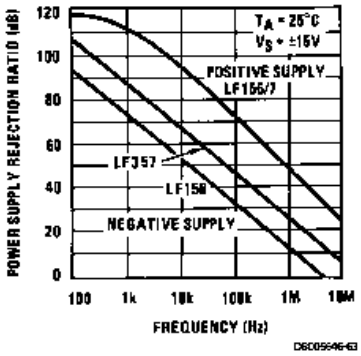
DS005646-61

Typical AC Performance Characteristics (Continued)

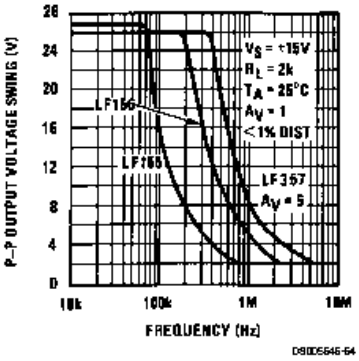
Power Supply Rejection Ratio



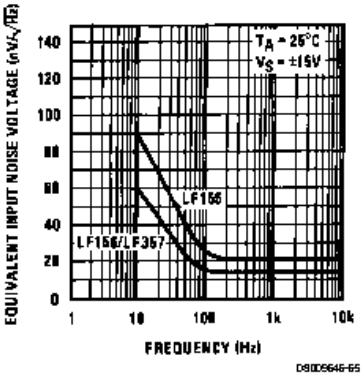
Power Supply Rejection Ratio



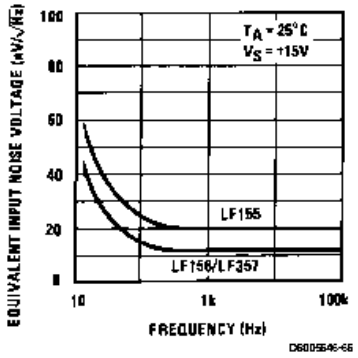
Undistorted Output Voltage Swing



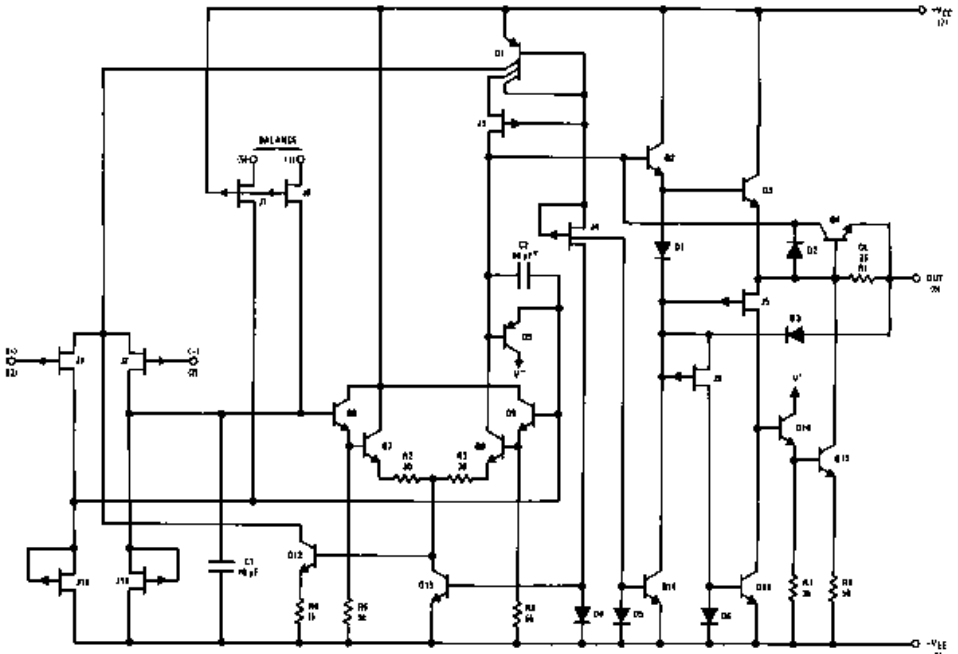
Equivalent Input Noise Voltage



Equivalent Input Noise Voltage (Expanded Scale)



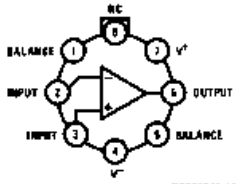
Detailed Schematic



*C = 3 pF in LF357 series.

Connection Diagrams (Top Views)

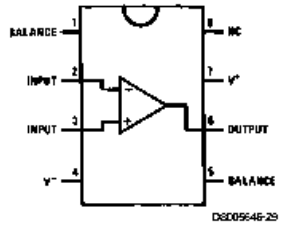
Metal Can Package (H)



*Available per JM38510/11401 or JM38510/11402

Order Number LF155H, LF156H, LF356BH, LF356H, or LF357H
See NS Package Number H08C

Dual-In-Line Package (M and N)



Order Number LF356M, LF356MX, LF355N, or LF356N
See NS Package Number M08A or N08E