COMPUTADOR DE FLUJO: DISEÑO E IMPLEMENTACIÓN DEL SOFTWARE PARA MEDICIÓN DE GAS

LUIS CARLOS GÓMEZ MORALES

UNIVERSIDAD INDUSTRIAL DE SANTANDER
FACULTAD DE INGENIERÍAS FÍSICO – MECÁNICAS
ESCUELA DE INGENIERÍA DE SISTEMAS
BUCARAMANGA
2004

COMPUTADOR DE FLUJO: DISEÑO E IMPLEMENTACIÓN DEL SOFTWARE PARA MEDICIÓN DE GAS

Proyecto de Grado para optar el título de

INGENIERO DE SISTEMAS

Autor

LUIS CARLOS GÓMEZ MORALES 941127

Director de la Investigación

_____

Ing. LUIS IGNACIO GONZALEZ

Codirector

_____

Ing. EDGAR JAVIER BARAJAS HERRERA

Candidato a *Magíster en Potencia Eléctrica UIS*

Bucaramanga 2004

DEDICATORIA

A mi madre.

AGRADECIMIENTOS

El autor expresa sus agradecimientos a:

LUIS IGNACIO GONZALEZ, Ingeniero de Sistemas y Director de la investigación.

NELFOR SAMAEL CASTELBLANCO, por su constante motivación y apoyo en este trabajo.

EDGAR JAVIER BARAJAS HERRERA, Ingeniero Eléctrico y Codirector de la investigación.

CONTENIDO

	Pág.
INTRODUCCIÓN	
1. PRINCIPIOS DE MEDICIÓN DE GAS E INSTRUMENTACION PRIMARIA.	3
1.1 MEDIDORES QUE PRODUCEN PRESIONES DIFERENCIALES	4
1.1.1 Medidor de orificio.	4
1.1.2 Medidor de codo.	6
1.1.3 El tubo Venturi.	6
1.1.4 El tubo Pitot.	7
1.1.5 La tobera de flujo.	8
1.2 MEDIDORES DE VELOCIDAD	9
1.2.1 Medidores de turbina.	9
1.2.2 Medidor de flujo vórtice.	10
1.3 MEDIDORES DE DESPLAZAMIENTO	11
1.3.1 Medidor Rotativo.	11
1.3.2 Medidor de diafragma.	12
1.4 OTROS MEDIDORES	14
1.4.1 Medidores ultrasónicos.	14
1.4.2 Medidores de Flujo Coriolis.	15
1.4.3 Medidor de flujo magnético.	16
1.4.4 Medidores termodinámicos.	17
2. Evolución Tecnológica de los Computadores de Flujo.	18
2.1 COMPUTADOR DE FLUJO ANÁLOGO.	18
2.2 COMPUTADOR DE FLUJO DIGITAL.	18
2.2.1 Basados en circuitos digitales.	19
2.2.2 Basados en microprocesador.	19

2.2.3 Software y memoria. 20
2.2.4 Consumo de potencia. 21
2.3. ESTADO DEL ARTE DE LOS COMPUTADORES DE FLUJO. 21
2.3.1 Computadores de flujo de alta tecnología. 22
2.3.2 Capacidad de CÓMPUTO. 23
2.3.3 Opciones de Entrada-Salida (E/S). 23
2.3.4 Comunicaciones Digitales. 24
2.3.5 Adquisición y Almacenamiento de Datos. 25
2.3.6 Descripción de un Daniels. 25
3. DISEÑO PRELIMINAR. 27
3.1 Consideraciones Preliminares. 27
3.2 Unidad de Medición. 28
3.2.1 Sobre los Transmisores. 28
3.2.1.1 Transmisores de Presión. 29
3.2.1.2 Transductores de Temperatura. 29
3.2.2 Sobre la Conversión Análogo-Digital. 30
3.2.2.1 Muestreo. 30
3.2.2.2 Temperatura en los ADCs. 31
3.2.3 Sobre la Incertidumbre. 31
3.3 UNIDAD DE CONTROL Y PROCESAMIENTO. 31
3.3.1 Sistema Embebido TDS2020F. 32
3.3.2 Memoria flash 33
3.3.3 Lenguaje Forth. 33
3.3.4 Hardware. 34
3.3.5 Usando Forth. 35
3.3.6 Velocidad. 35

3.3.7 Memoria. 35
3.3.8 Entrada salida paralela 36
3.3.9 Bus I2C . 36
3.3.10 Entrada salida serial 37
3.3.11 Microprocesador. 37
3.3.12 Soporte de teclado. 37
3.3.13 Displays de cristal líquido. (LCD) 38
3.3.14 Periféricos externos. 38
3.3.15 Conversor análogo a digital. 38
3.3.16 Conversor digital a análogo. 38
3.3.17 Tiempo y fecha no volátiles. 38
3.3.18 Temporizador-contadores. 39
3.3.19 Temporizadores de “watchdog”. 39
3.3.20 Modo de baja potencia. 39
3.3.21 Fuente de alimentación. 39
3.3.22 Interfase de bus. 39
3.3.23 Dimensiones. 40
3.3.24 Sistema Forth ANS. 40
3.3.25 Extensiones ROM Forth. 41
3.3.26 Multitarea. 41
3.3.27 Software de desarrollo (firmware). 41
3.4 UNIDAD DE INTERFASE AL USUARIO 42
3.4.1 Visualizadores 42
3.4.2 Teclados 42
4. DISEÑO DEL HARDWARE. 44
4.1 SISTEMA EMBEBIDO TDS2020F– CPU. 44

4.2 CONVERSOR ANÁLOGO-DIGITAL. 47
4.3 ACONDICIONAMIENTO DE SEÑAL. 51
4.4 CONEXIÓN DEL ADS7825 AL TDS2020F. 54
4.5 CONEXIÓN DE SENSORES DE TEMPERATURA. 56
4.6 CONEXIÓN DE SENSORES PRESION. 58
4.7 CONEXIÓN DE ENTRADA DE PULSOS. 60
5. GENERALIDADES DE FORTH. 61
5.1 ARQUITECTURA. 61
5.2 La Pila. 62
5.3 EJEMPLOS. 63
5.4 FORTH Y VELOCIDAD. 65
5.5 PROCESADORES FORTH. 67
5.6 Forth en el Mercado 68
6. Transferencia de Custodia. 69
6.1 INFLUENCIAS DEL CAPÍTULO 21 DE API. 70
6.2 Seguridad y Normativa. 70
7. DISEÑO DEL FIRMWARE. 72
7.1 DESARROLLO DE LA NORMA AGA8-GROSS2 72
7.1.1 Teoría del Método 72
7.1.2 Implementación del Método 77
7.2 DESARROLLO DE LA NORMA AGA7 83
7.3 CARACTERISTICAS DEL SOFTWARE EMBEBIDO 85
7.3.1 Conversión Análogo-Digital 85
7.3.2 Relojes y Contadores 85
7.3.3 Fecha y Hora 86
7.3.4 Soporte de Teclado 88
7.3.5 Manejo de LCD 90
7.4 DESARROLLO DEL FIRMWARE EJECUTABLE 92

8. PRUEBA EN CAMPO 97
8.1 Actividades realizadas 97
8.2 Descripcion de la Prueba 97
8.3 RESULTADOS DE LA PRUEBA 97
8.4 Observaciones y conclusiones de la prueba 106
9. CONCLUSIONES 108
10. Recomendaciones 110
BIBLIOGRAFÍA 111
ANEXOS 113

LISTA DE FIGURAS

	Pág.
Figura 1. Medidor de platina de orificio	4
Figura 2. Medidor de codo	6
Figura 3. Medidor tipo Venturi	7
Figura 4. Medidor tipo Pitot	8
Figura 5. Medidor tipo tobera	8
Figura 6. Medidor de turbina	9
Figura 7. Medidor tipo Vórtice	10
Figura 8. Medidor Rotativo	12
Figura 9. Medidor tipo diafragma	13
Figura 10. Medidor Ultrasónico	14
Figura 11. Medidor tipo Coriolis	16
Figura 12. Medidor de tipo magnético	16
Figura 13. Medidor termodinámico	17
Figura 14. Paquete de Desarrollo TDS2020F	33
Figura 15. Tarjeta CPU TDS2020F	34
Figura 16. Partes de la CPU TDS2020F	44
Figura 17. Ubicación de pines comunicación y alimentación	45
Figura 18. Entorno de Programación TDS-PC	47
Figura 19. Distribución de Pines TDS2020F	49
Figura 20. Conversor Análogo-Digital ADS7825	50
Figura 21. Diagrama de Bloques del ADS7825	50
Figura 22. Conexión Tí pica del ADS7825	51
Figura 23. Encapsulados del INA118	52

Figura 24. Diagrama de Bloques del INA118 53
Figura 25. Conexión Típica del INA118 54
Figura 26. Conexiones del ADS7825 a la CPU 55
Figura 27. Acondicionador de Señal para una RTD PT100. 57
Figura 28. Bornes Típicos de una RTD 57
Figura 29. RTD PT-100 58
Figura 30. Amplificación de los Transductores de Presión. 58
Figura 31. Alimentación para los sensores de presión. 59
Figura 32. Sensor de Presión Barksdale 59
Figura 33. Sensor de Presión Diferencial 60
Figura 34. Circuito para entrada de pulsos 60
Figura 35. Diagrama de Flujo para hallar HCH 79
Figura 36. Diagrama de Flujo AGA8 Gross-2 80
Figura 37. Diagrama de Flujo Rutinas AGA8 Gross-2 81
Figura 38. Diagrama de Flujo del Firmware Ejecutable 96
Figura 39. Presión Promedio 98
Figura 40. Temperatura Promedio 99
Figura 41. Volumen corregido día_1 99
Figura 42. Volumen no corregido día_1 100
Figura 43. Volumen corregido día_2. 101
Figura 44. Volumen no corregido día_2. 101
Figura 45. Volumen corregido día_3. 102
Figura 46. Volumen no corregido día_3. 103
Figura 47. Volumen corregido día_4. 103
Figura 48. Volumen no corregido día_4. 104

Figura 49. Volumen corregido día_5. 104
Figura 50. Volumen no corregido día_5. 105
Figura 51. Volumen Corregido 105

LISTA DE TABLAS

Pág.

Tabla 1. Opciones de Entrada-Salida del TDS2020F 36
Tabla 2. Configuración del Cable de Comunicaciones 46
Tabla 3. Secuencia de Operaciones de un programa en Forth 65
Tabla 4. Siglas de Entidades Normativas 70
Tabla 5. Rango de Aplicación del Método AGA8 Gross 2 73
Tabla 6. Coeficientes viriales iterativos para Nitrógeno y Dióxido de Carbono 75
Tabla 7. Coeficientes viriales Iterativos para el equivalente hidrocarbonado, CH 75
Tabla 8. Coeficientes viriales Iterativos para el equivalente hidrocarbonado, CH 76
Tabla 9. Tiempos de Conversión en A-D, TDS. 85
Tabla 10. Relojes y contadores. 86
Tabla 11. Fecha y hora. 86
Tabla 12. Mecanismos de reloj en el TDS2020*F* 87
Tabla 13. Comandos usados para soporte de teclado. 88
Tabla 14. Librerías usadas para el manejo de LCD 90
Tabla 15. Presión Promedio 98
Tabla 16. Temperatura Promedio 98
Tabla 17. Volumen día_1. 99
Tabla 18. Volumen día_2. 100

Tabla 19. Volumen día_3. 102
Tabla 20. Volumen día_4. 103
Tabla 21. Volumen día_5. 104
Tabla 22. Volumen corregido diario. 105
Tabla 23. Comparación del ECOFLOW2020 vs AE5000 107

LISTA DE ANEXOS

		Pág.
ANEXO 1	CODIGO FUENTE de AGA8 GROSS-2	113
ANEXO 2	CODIGO FUENTE DEL FIRMWARE FINAL	123

RESUMEN

TÍTULO: Computador de Flujo: Diseño e implementación del software para medición de gas. [*]

AUTOR: Luis Carlos Gómez Morales [**]

PALABRAS CLAVE: Medición de Gas Natural, Computador de flujo, *software*, *Forth*, interrupción *hardware*, interrupción *software*, sistema embebido.

DESCRIPCIÓN:
Este proyecto hace parte de un macroproyecto para la construcción de un prototipo de computador de flujo para medición de gas natural en transferencia de custodia. El proyecto está financiado por el Centro de Investigación del Gas de la Universidad Industrial de Santander, la Empresa Colombiana de Gas y Colciencias.

El objetivo del proyecto es diseñar e implementar el software para la medición de flujo de gas natural con turbina; usando un *hardware* de soporte existente basado en microprocesador y en lenguaje Forth. Las variables a medir son temperatura, presión y velocidad del flujo. El proyecto pretende que se puedan sustituir las importaciones de computadores de flujo en Colombia por medidores fabricados en el país, contribuyendo así con el desarrollo tecnológico, la generación de empleo y la formación de investigadores.

En el presente documento se realiza una investigación sobre el estado de arte de los computadores de flujo y sus especificaciones actuales. Se establecen los principios y fundamentos del lenguajes *Forth* , en el cual se implementa el *software* de desarrollo. Se realiza también un estudio de las normas técnicas AGA 8 y AGA 7 para su implementación en los cálculos y corrección del volumen de gas; se desarrollan los algoritmos de adquisición de señales y la programación de una pantalla LCD y teclado de interfase al usuario.

[*] Trabajo de Grado
[**] Facultad Físico-Mecánicas. Ing. Sistemas. Luis Ignacio Gonzalez.

ABSTRACT

TITLE: Flow Computer: Design and implementation of the software for the measurement of gas. [*]

AUTHOR: Luis Carlos Gómez Morales [**]

KEY WORDS: Natural Gas Measurement, Flow Computer, Hardware, Primary Instrumentation, Secondary Instrumentation, Signal Conditioning.

DESCRIPTION:

This project belongs to a macroproject to design and to build a flow computer prototype for natural gas measurement in custody transfer. The funds for this project come from the "Centro de Investigación del Gas" of the "Universidad Industrial de Santander", the "Empresa Colombiana de Gas" and Colciencias.

The objective of the project is to design and to implement the software for the measurement of natural gas flow with turbine-meter, using an existent support hardware based on microprocessor and on Forth language . The measured variables will be temperature, absolute pressure and speed of the flow. The project goal is to replace the importations in Colombia with flow computers made in the country, thus contributing with the technological development, the employment generation and the formation of researchers.

In the present document is made a investigation about the state-of-the-art the flow computers and its present day specifications. Are established the principles and foundations of the Forth language which is used to implement the development software. Besides is made a research of the norms American Gas Association (AGA 8 and AGA 7) for its implementation in the calculation and correction of the gas volume in a pipe; is developed the algorithm of acquisition signals and the programming of a LCD screen and the user interface keyboard.

[*] Degree Project

[**] Facultad Físico-Mecánicas. Ing. Sistemas. Luis Ignacio Gonzalez..

INTRODUCCIÓN

En la industria del gas es fundamental disponer de sistemas confiables y eficientes que permitan realizar la medición de los volúmenes de gas que se reciben y se entregan. Para realizar estas labores se dispone de medidores primarios como los medidores rotativos, tipo diafragma, platina de orificio, turbina y de ultrasonido. La medida primaria del flujo requiere correcciones, ya que ésta se ve influenciada por la temperatura, la presión y las características del gas; de modo que se hace necesario emplear un dispositivo, usualmente electrónico, que permita obtener datos reales y datos históricos de la presión del flujo y el volumen del gas que está circulando en la tubería. Este dispositivo se conoce como Computador de Flujo.

Los computadores electrónicos de flujo han sido usados por la industria de transmisión de gas por más de veinte años. Durante este tiempo se han desarrollado desde burdos indicadores de flujo hasta sistemas exactos de medida de energía.

Hoy en día, el aumento de la aceptación del usuario, la nueva tecnología electrónica y el alto costo de la energía parecen converger para crear condiciones que hacen que el uso de computadores de flujo tome lugar como el primer elemento para la corrección y registro del flujo de gas.

Además de tener la responsabilidad de transferir con integridad fiscal el flujo, el computador de flujo actúa mas como el "cerebro" de un sistema mejorado que integra numerosas funciones de medición, secuenciación, almacenamiento de datos y comunicaciones dentro de un único dispositivo autónomo.

Con todo esto, la industria exige un dispositivo de medida altamente seguro, confiable y exacto, que reduzca el riesgo de errores del sistema y que garantice la no alteración, por medios externos, de los datos registrados. La experiencia con algunos dispositivos de campo programables e inseguros que están controlando bombas, motores y compresores, los cuales han perdido datos críticos o señales medidas como resultado de un *reset* accidental o de la reprogramación del equipo, hace que se tomen medidas muy exigentes en el diseño y selección de estos dispositivos.

El presente proyecto tiene como objetivo realizar una revisión bibliográfica sobre los medidores de gas natural empleados en la industria, el estado del arte de los computadores de flujo empleados en transferencia de custodia y las normas AGA (*American Gas Asociation*) que rigen las mediciones. Toda esta documentación, con el fin de llevar a cabo el diseño e implementación del

software para la medición de flujo de gas natural con turbina. Este proyecto hace parte del macroproyecto para la construcción e implementación del *hardware*, *software* y *firmware* de un computador de flujo, desarrollado como trabajo de investigación para la Maestría en Ingeniería, el cual ha sido financiado en su totalidad por el Centro de Investigación del Gas, Ecogás y Colciencias.

La primera etapa del proyecto consistió en una recopilación bibliográfica sobre los medidores primarios empleados en la industria del gas y especialmente los usados en Colombia por Ecogás. Posteriormente se buscó información relacionada con computadores de flujo y su estado del arte con el fin de adquirir criterios para el diseño. Estos temas se encuentran en los capítulos uno y dos, respectivamente. El capítulo tres presenta consideraciones a tener en cuenta en el diseño preliminar del computador de flujo. En el capítulo cuatro se presenta la selección de la instrumentación de medida y el diseño del *hardware;* que fue elaborado por un proyecto paralelo de la escuela de ingeniería electrónica y es reproducido acá para ampliación del hardware que sirve de soporte para la realización de este proyecto.

En el capitulo 5 se presentan generalidades del lenguaje utilizado en el desarrollo del firmware y algunos ejemplos. El capitulo seis presenta algunas consideraciones en la transferencia de custodia. El capitulo 7 expone el diseño del firmware; el desarrollo de las normas AGA 8 Gross 2, y AGA 7; las características del software embebido y el desarrollo del firmware ejecutable.
El capitulo 8 muestra los resultados de la prueba en campo. Por último en los capítulos 9 y 10 se presentan las conclusiones del proyecto y algunas recomendaciones.

En los anexos se presentan los códigos fuentes del método AGA 8 Gross 2 en matlab y del firmware final en Forth.

1. PRINCIPIOS DE MEDICION DE GAS E INSTRUMENTACIÓN PRIMARIA

Los beneficios que obtiene una compañía de transmisión de gas resultan parcialmente del transporte de cantidades medidas de gas natural. Una compañía de transmisión se encarga de proveer este servicio a las compañías de productores y consumidores de gas natural. A su vez, las compañías productoras y consumidoras requieren hacer medición para determinar con exactitud los pagos a recibir o a desembolsar por el gas comercializado.

La medición de gas natural se debe realizar siempre que se presente un cambio de responsabilidad o propiedad del producto. El gas se mide cuando se compra, se vende, se almacena, o se intercambia entre propietarios, transportadores o procesadores. También se requieren mediciones cuando el gas se transporta a través de fronteras nacionales o internacionales. Además, las medidas son necesarias para monitorear el desempeño del sistema de tuberías.

El método ideal para medir gas natural consiste en medir la energía real entregada. Sin embargo, no existen actualmente métodos convenientes para medir directamente la energía del gas natural. La energía entregada se puede calcular multiplicando el volumen entregado por el valor de calentamiento del gas. El método mas común para medir gas natural consiste en determinar el volumen o la velocidad del flujo que pasa a través de algún tipo de medidor.

Los gases comerciales se comportan de manera diferente que los gases ideales, estos varían en su composición química y en sus propiedades físicas. Esta desviación hace que la medición práctica sea mas compleja que la teoría básica. La desviación de la temperatura y presión base implica la necesidad de dispositivos con mediciones compensadas o compensación calculada. Para elegir el dispositivo de medida y el tipo de instalación a usar se deben tener en cuenta factores como la contaminación, la acumulación de líquidos y las pulsaciones.

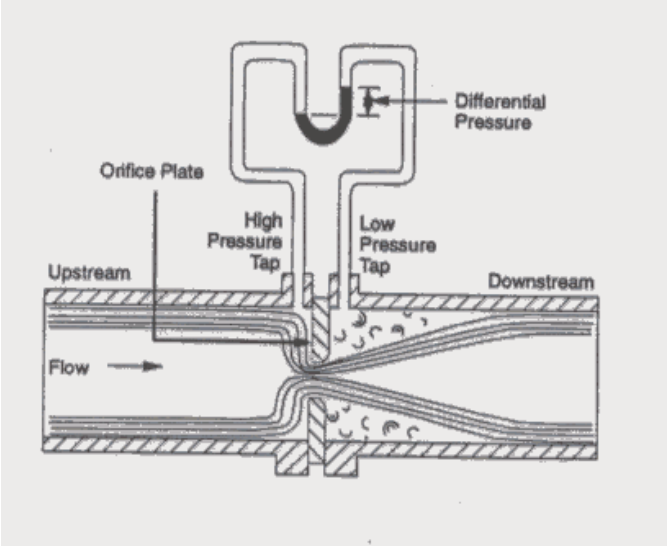
Actualmente existen diversos dispositivos que emplean principios de medición diferentes, entre estos, medición de presión, de velocidad, de desplazamiento, etc. A continuación se esbozará el principio y el funcionamiento básico de cada uno de estos dispositivos.

1.1 MEDIDORES QUE PRODUCEN PRESIONES DIFERENCIALES

Los medidores de flujo que producen presiones diferenciales, se llaman medidores de tipo cabeza y se seleccionan frecuentemente por su larga historia de uso en muchas aplicaciones. Un número de elementos primarios pertenece a esta clase: El Orificio Concéntrico, El *Venturi*, La Tobera de Flujo, La Cuña, El Codo y el Tubo *Pitot*, entre otros.

1.1.1 Medidor de orificio. La platina de orificio es uno de los dispositivos medidores de flujo mas usado en la industria donde se requieren medir, con buena exactitud, grandes volúmenes a presión alta.

Figura 1. Medidor de platina de orificio



Memorias Seminario de Medición de Gas, CentrOriente S.A.

Consiste básicamente de una placa circular perforada (figura 1) que se inserta en la tubería causando una restricción al flujo, lo que genera una presión diferencial. Cuando el fluido interacciona con la restricción se produce un cambio de energía potencial (presión) a energía cinética (velocidad). La forma descrita por el flujo al pasar por el orifico se conoce como vena contractada. El teorema de *Bernoulli* describe el funcionamiento de la placa de orificio relacionando la energía potencial, la energía cinética y las pérdidas por fricción del fluido con la tubería. La presión diferencial generada y el flujo tienen proporcionalidad tipo cuadratura, es decir, el flujo es proporcional a la raíz cuadrada de la presión diferencial. Esta relación se observa en la siguiente ecuación:

$Q_h = C' (h_W P_f)^{1/2}$ donde,
Q_h = Tasa de flujo volumétrico en condiciones base (pies cúbicos/hora).
C' = Constante de flujo del orificio, depende de las características del gas.
h_W = Presión diferencial (pulgadas de agua a 60 °F).

P_f = Presión estática absoluta (libras por pulgada cuadrada absoluta- psia).
C' = F_b F_r Y F_{pb} F_{tb} F_{tf} F_g F_{pv} F_a F_m F_l

De la ecuación para C' se tiene,

F_b = Factor básico del orificio.
F_r = Factor del número de Reynolds.
Y = Factor de expansión.
F_{pb} = Factor de presión base.
F_{tb} = Factor de temperatura base.
F_{tf} = Factor de temperatura de flujo.
F_g = Factor de gravedad específica.
F_{pv} = Factor de supercompresibilidad.
F_a = Factor de expansión térmica del orificio.
F_m = Factor de manómetro (solamente para medidores de mercurio).
F_l = Factor de localización del medidor.

Los constituyentes básicos del elemento primario son el ejecutor de la medida, la platina de orificio y su adaptador, y las tomas de presión aguas arriba y aguas abajo. Usualmente se usan tomas con brida puesto que proveen mayor exactitud y fidelidad que otras tomas. La relación Beta es el cociente entre el diámetro del orificio y el diámetro interno de la tubería; su valor recomendado debe estar entre 0.2 y 0.6.

La presión diferencial obtenida a través de la platina de orificio por las tomas con brida, se transfiere al elemento secundario, que puede ser un registrador de carta o un transmisor de presión diferencial en conjunto con un computador de flujo para lograr medición electrónica de gas en tiempo real.

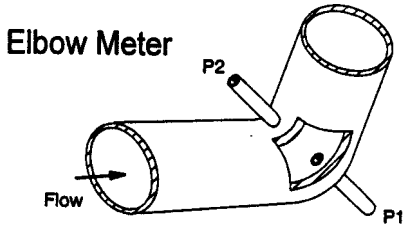
Como todos los medidores, la platina de orificio tiene algunas ventajas y desventajas que son elementos determinantes a la hora de seleccionarla para algún tipo de aplicación. Entre las ventajas se encuentran:

- Normas y estándares bien documentados.
- Compatibilidad con sistemas electrónicos de medición de gas.
- Amplio uso y gran aceptación a nivel industrial.
- Fidelidad y exactitud (el orificio puede operar a 0.75-1.25% de error).
- Bajo costo en la inversión y su instalación.
- Dispositivo sencillo que no posee partes móviles en la línea de flujo.
- Mantenimiento mínimo.
- No tiene limitaciones en cuanto a temperatura y presión.
- Sistema de lectura electrónico disponible para el cálculo de flujo.

Algunas desventajas del medidor tipo orificio son:

- Bajo rango de operación que impone la necesidad de cambios frecuentes de platina para ajustarse a flujos variantes.
- Caídas de presión altas para flujos determinados, particularmente a bajos β. Estas caídas son mayores que las presentadas por los medidores de turbina.
- Muy sensible a perfiles no uniformes de velocidad.
- Susceptibilidad a errores en la medida por causa de la presencia de líquidos en la corriente de gas.
- Los transmisores de presión y los registradores de carta requieren calibración regular en campo.

Figura 2. Medidor de codo

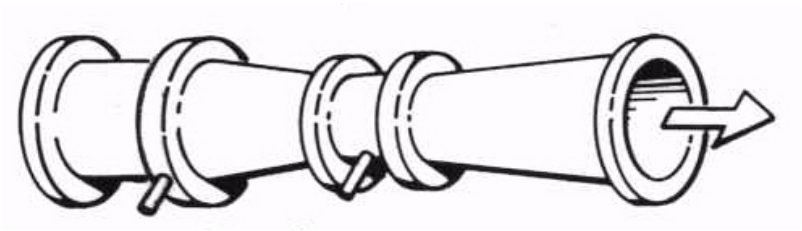


Memorias Seminario de Medición de Gas, CentrOriente S.A.

1.1.2 Medidor de codo. Este dispositivo (figura 2) tiene una toma de presión en la parte interna y en la externa de un tubo estándar en forma de codo. La velocidad de flujo a través de éste es aproximadamente proporcional a la raíz cuadrada del diferencial entre estas tomas. Los medidores para las tomas deben ser extremadamente sensibles a diferencias de presión muy pequeñas.

1.1.3 El tubo *Venturi*. La entrada ahusada y la salida divergente del *Venturi* (figura3) reduce sustancialmente la pérdida de presión permanente. Este elemento diferencial se puede usar para aplicaciones de flujo sucio, puesto que la suciedad no se depositará sobre las secciones contorneadas como ocurre en la lámina de orificio. Se diseñó inicialmente para tamaños de línea de 6 pulgadas y mayores, en aplicaciones de flujo de agua y desechos.

Figura 3. Medidor tipo Venturi



Medición y Regulación, Hernando Galvis Barrera.

Algunas ventajas del medidor tipo *Venturi* son:

- Baja caída de presión comparado con otros medidores diferenciales.
- Excelente desempeño para fluidos con sólidos.

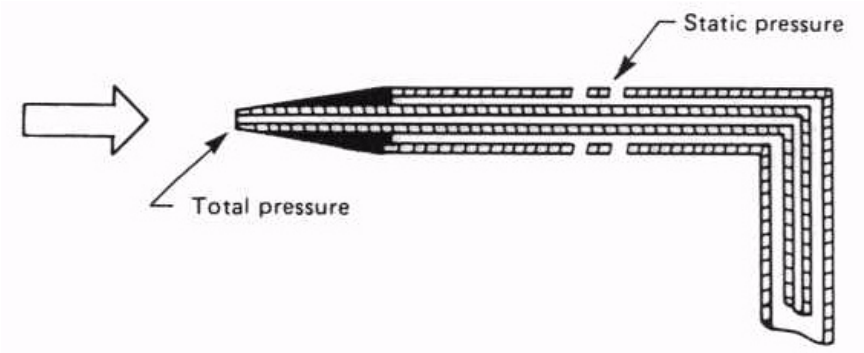
Entre las desventajas del medidor tipo *Venturi* se encuentran:

- Rango de operación limitado, solo es utilizado en aquellos sistemas donde el rango de flujo es bien conocido.
- Costos altos.
- Unidades grandes y de gran peso lo que dificulta su manejo e instalación.

1.1.4 El tubo *Pitot*. Se usa para tubos de gran tamaño, cuando el fluido es un líquido limpio o un gas (vapor) y se requiere una medición poco costosa. Para este instrumento (figura 4), la diferencia entre la presión total (de estancamiento) y la presión estática sigue la relación de la raíz cuadrada, registrando únicamente la velocidad en la profundidad de inserción.

El tubo *Pitot* de pasos múltiples se extiende en la sección transversal de todo el tubo y ha reemplazado al tubo *Pitot* tradicional. Su fácil instalación, bajo costo y buen funcionamiento hacen a estos instrumentos competitivos con otros instrumentos que causan presión diferencial en muchas aplicaciones industriales. Existen varios diseños disponibles en este tubo, desde un cilindro hasta una punta en forma de diamante.

Figura 4. Medidor tipo Pitot

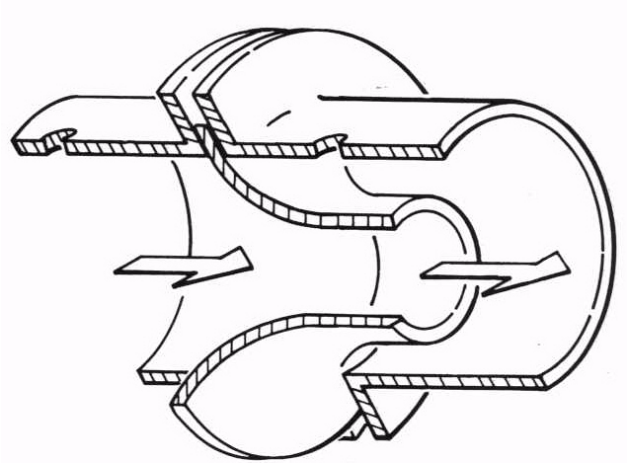


Medición y Regulación, Hernando Galvis Barrera.

1.1.5 La tobera de flujo. Es un dispositivo con características intermedias entre el *Venturi* y el medidor de orificio.

Como se muestra en la figura 5, consiste en una sección corta y recta, seguida de un conducto bien redondeado, el cual se centra en la tubería por medio de bridas. En éste no se usa la sección divergente del *Venturi*. En la tobera, al igual que en el *Venturi*, la velocidad se incrementa y la presión disminuye, y el chorro emerge libremente en la sección aguas abajo, como en el orificio. Durante la contracción la corriente se confina y se controla por medio de unos límites sólidos, no se forma vena contracta y no existen incertidumbres en lo relativo a la velocidad del chorro. La toma aguas arriba está aproximadamente a un diámetro de tubería antes de la tobera. La toma corriente abajo está inmediatamente detrás de la salida de la tobera. El coeficiente de la tobera es aproximadamente igual al de una platina de orificio. El coeficiente de una tobera usualmente es suministrado por el fabricante.

Figura 5. Medidor tipo tobera



Medición y Regulación, Hernando Galvis Barrera.

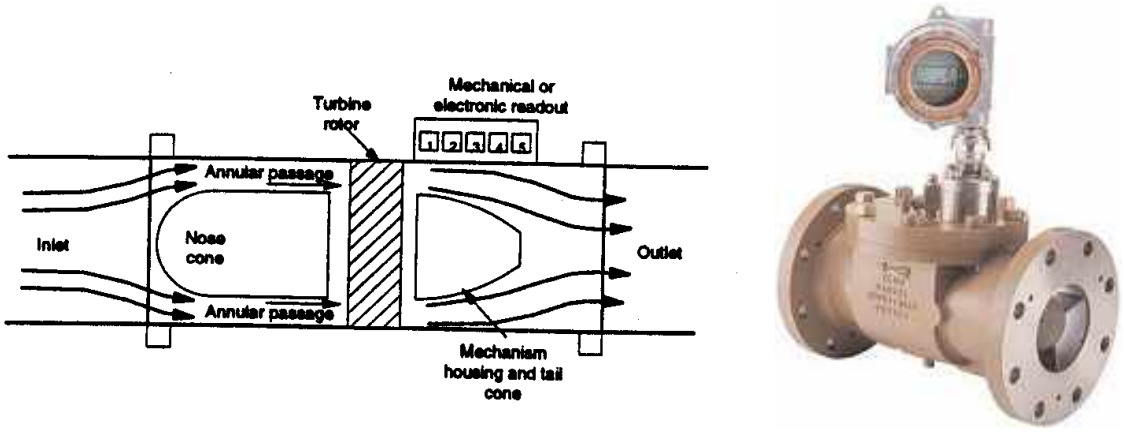
1.2 MEDIDORES DE VELOCIDAD

Estos medidores también se conocen como medidores lineales. Son dispositivos que miden la velocidad de una corriente de gas que está relacionada con el volumen a las condiciones de la línea. La salida de estos dispositivos es directamente proporcional a la velocidad de la corriente de gas. Este tipo de medidores producen sólo pequeñas caídas de presión a través del elemento primario.

1.2.1 Medidores de turbina. La velocidad del rotor de un medidor de flujo de turbina se incrementa linealmente con la velocidad del flujo. Así, la rotación de las turbinas es una medida de la velocidad y se detecta por medio de sensores magnéticos externos o por medio de engranajes, apareciendo la lectura, en pies cúbicos o en metros cúbicos, en un contador montado en la parte externa del medidor (figura 6). La relación entre la velocidad del fluido y la velocidad del rotor es lineal (dentro de un ± 5%), sobre un amplio margen de 10:1 a 20:1. El funcionamiento a baja velocidad se ve afectado por el perfil de velocidad, la fricción a lo largo de las turbinas, fricción en los cojinetes y otros torques retardantes. El medidor de flujo de turbina se utiliza para medición de flujos de gases y líquidos limpios, con un amplio margen de flujo.

Debido a las grandes diferencias de densidad entre gases y líquidos se dispone de dos diseños diferentes de medidores de flujo tipo turbina. Ambos diseños se utilizan en aplicaciones de transferencia de custodia que requieren muy buena exactitud.

Figura 6. Medidor de turbina



Memorias Seminario de Medición de Gas, CentrOriente S.A y Medición y Regulación, Hernando Galvis Barrera.

A continuación se van a mencionar las ventajas del medidor tipo turbina:

- Buena exactitud dentro del rango de operación del medidor.

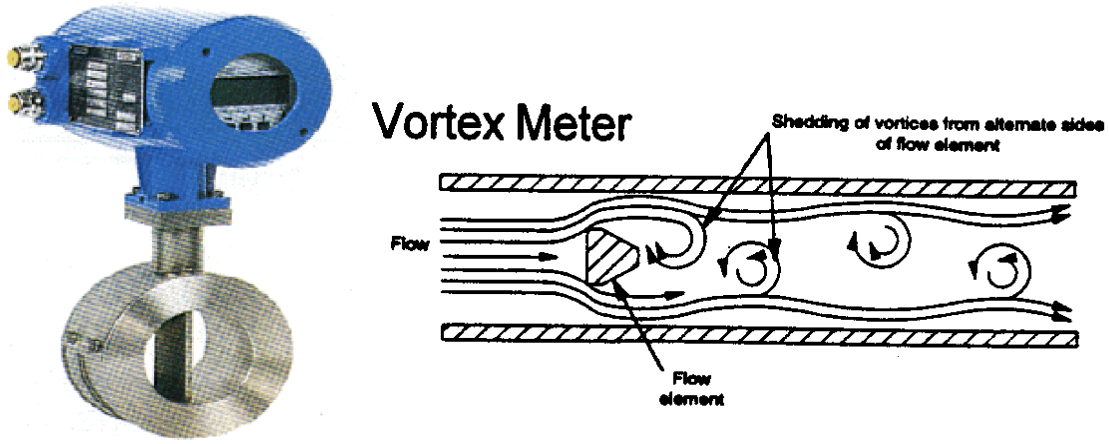
- Equipos electrónicos disponibles para lectores de flujo en corto tiempo y alta resolución.
- Costos medios de inversión comparados con otros tipos de medidores.
- Excelente rango de operación a altas presiones.

También se presentan algunas desventajas de este dispositivo:

- Necesidad de revisión periódica de todos sus componentes para garantizar una buena exactitud.
- Requiere perfil de velocidad uniforme.

1.2.2 Medidor de flujo vórtice. Los medidores vórtice o *vortex* pertenecen a una clase de dispositivos llamados osciladores de flujo. El principio de este medidor se fundamenta en la inestabilidad de un flujo cuando éste se divide en dos caminos alrededor de un obstáculo, lo cual hace que se formen remolinos (vórtices) discretos a los lados del objeto, en forma alterna y a intervalos regulares. Con la ayuda de los sensores apropiados, los remolinos se cuentan y aparecen desplegados en unidades de volumen. Si el flujo es estable, la formación del segundo torbellino tardará el mismo tiempo que la del primero y ésta será proporcional a la velocidad de flujo.

Figura 7. Medidor tipo Vórtice



Medición y Regulación, Hernando Galvis Barrera y Memorias Seminario de Medición de Gas, CentrOriente S.A.

La diferencia básica entre los medidores tipo Vórtice está en la geometría del obstáculo; la frecuencia del desprendimiento es función del ancho de la barrera, la longitud y la relación de radios respecto a la tubería.

Las ventajas del medidor tipo vórtice son:

- Amplio margen de operación con señal de salida lineal.
- Para fluidos limpios su desempeño es estable.
- La señal de salida puede ser leída por instrumentos electrónicos.
- Instalación simple, costos moderados de inversión.
- Los efectos de viscosidad, presión y temperatura del fluido son mínimos.
- No posee partes móviles en contacto con el fluido.

La principal desventaja del medidor tipo vórtice es la necesidad de perfil de flujo uniforme, libre de remolinos y chorros.

Varios fabricantes ofrecen medidores de flujo de vórtice para tamaños de línea de ½ a 16 pulgadas (12 a 400 mm). Estos medidores se están utilizando ampliamente, debido a su exactitud, margen, fidelidad y relativa insensibilidad a las propiedades de los fluidos. Sin embargo, requieren condiciones de flujo sin alteraciones, por lo que se necesitan largas secciones de tuberías rectas tanto aguas arriba como aguas abajo.

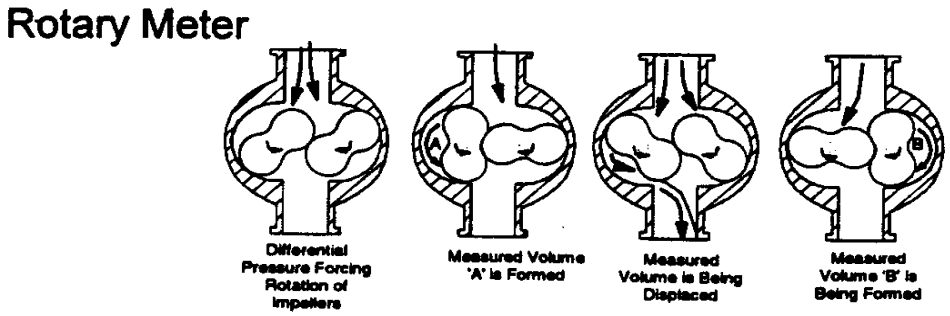
1.3 MEDIDORES DE DESPLAZAMIENTO

Estos dispositivos segregan o dividen el flujo en volúmenes discretos y luego suman el volumen total, contando las unidades de volumen que pasan a través del medidor. Cuando el fluido entra a una cámara, un impulsor, pistón o diafragma, o un disco, rota o se mueve para acomodar el flujo que entra y se descarga un volumen conocido. Los medidores de desplazamiento positivo para gas y líquido miden volumen en línea a condiciones de presión y temperatura.

Su mayor uso es para medición de gas natural a baja presión, petróleo crudo y derivados del petróleo.

1.3.1 Medidor Rotativo. Este tipo de instrumento tiene válvulas rotativas que giran excéntricamente rozando con las paredes de una cámara circular y transportan el líquido en forma incremental de la entrada a la salida (figura 8). Se emplean mucho en la industria petroquímica para la medida de crudos y de gasolina con intervalos de medida que van de unos pocos litros por minuto, de líquidos limpios de baja viscosidad, y hasta 64000 litros/min, de crudos viscosos.

Figura 8. Medidor Rotativo



Memorias Seminario de Medición de Gas, CentrOriente S.A.

Hay varios tipos de medidores rotativos, siendo los mas empleados los cicloidales, los de dos rotores (birrotor) y los ovales. Los cicloidales contienen dos lóbulos engranados entre si que giran en direcciones opuestas manteniendo una posición relativa fija y desplazando un volumen fijo de fluido líquido o gas en cada revolución. El birrotor consiste de dos rotores sin contacto mecánico entre si que giran como únicos elementos móviles en la cámara de medida.

Las ventajas para este dispositivo son las siguientes:

- La calibración requerida es mínima.
- Exactitud (puede operar a menos de 1.5% de error).
- No hay pérdidas de gas.
- Está probado y es ampliamente usado en la industria del gas natural.

A continuación se presentan algunas desventajas de este medidor:

- Tiene mucho ruido y vibraciones.
- Si el medidor falla el flujo de gas estará completamente interrumpido.
- Velocidades de flujo altas y continuas pueden causar daño debido a la naturaleza de las partes móviles.

1.3.2 Medidor de diafragma. En el medidor de desplazamiento positivo tipo diafragma operan cuatro compartimentos de medición simultáneamente (figura 9); algunos se llenan mientras otros se descargan. La rotación se transmite a partir de un engranaje adecuado a un contador que lee el volumen total. Se requieren sellos para separar los volúmenes. La pérdida de presión a través del medidor provee la energía necesaria para accionar las partes móviles.

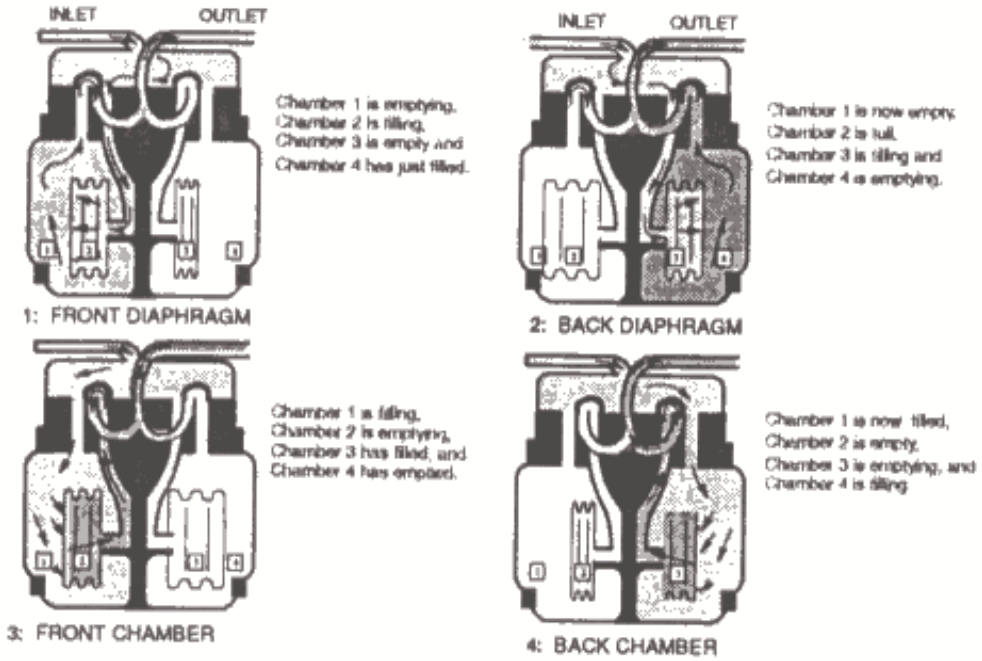
A continuación se presentan algunas ventajas de este medidor:

- Amplio margen.
- La exactitud de la medida no se ve afectada por cargas que fluctúen rápidamente.
- Caída de presión pequeña a través del medidor.
- Se requiere calibración y servicio mínimos.
- Se puede montar en un encapsulado pequeño.

Las desventajas para este dispositivo son las siguientes:

- No es práctico para grandes volúmenes con altas presiones debido a la naturaleza de diseño que tiene partes móviles susceptibles al uso.
- La acumulación de líquidos puede implicar medidas falsas.
- Tamaño complicado en relación a otros medidores con capacidad de medida equivalente.

Figura 9. Medidor tipo diafragma



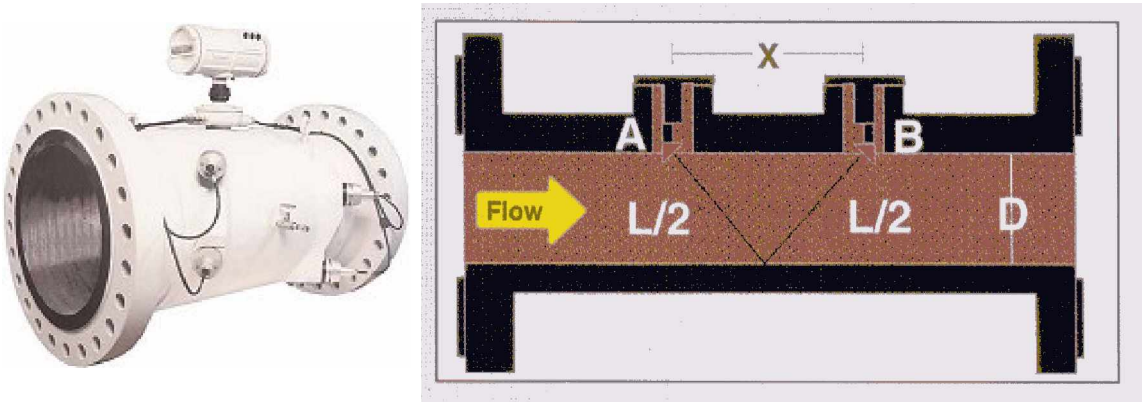
Memorias Seminario de Medición de Gas, CentrOriente S.A.

1.4 OTROS MEDIDORES

1.4.1 Medidores ultrasónicos. El principio de un medidor de flujo ultrasónico se fundamenta en enviar una onda de sonido de alta frecuencia (aproximadamente 1 MHz) en un ángulo agudo a través del tubo. Un transmisor envía señales ultrasónicas a un receptor (figura 10). La diferencia de tiempo entre la transmisión y la recepción de una señal depende de la velocidad del flujo. A partir de este principio, y conociendo las dimensiones del medidor, se puede determinar el volumen de flujo por unidad de tiempo. Es más ventajoso usar dos canales sónicos directos, de manera opuesta (T_1 à R_1 y T_2 à R_2), de esta forma la medida se hace independiente de la velocidad del sonido en el medio que se va a medir.

Los medidores ultrasónicos se pueden usar en tuberías desde 4 pulgadas. Dependen mucho del perfil del flujo, por lo que requieren largos tramos de tubería recta antes de la entrada y después de la salida.

Figura 10. Medidor Ultrasónico



Daniel Measurement and Control.

La determinación de la velocidad real del volumen de flujo para el medidor esquematizado en la figura 10, es la siguiente:

$$t_{AB} = L / (C + VX/L),\ t_{BA} = L / (C - VX/L)$$

$$V = (L^2 / 2X)((t_{BA} - t_{AB})/ t_{BA}\ t_{AB})$$

$$Q = V (\pi D^2 / 4) = K ((t_{BA} - t_{AB})/ t_{BA}\ t_{AB})$$

donde,

t_{AB} = tiempo de tránsito del transductor A al B.
t_{BA} = tiempo de tránsito del transductor B al A.
L = distancia entre los transductores.
C = velocidad del sonido en el gas.
X = distancia axial entre transductores.
D = diámetro del soporte del medidor.
K = constante para cada medidor específico.

Las ventajas del medidor ultrasónico son:

- No causa caídas de presión.
- Alto margen de operación.
- No posee partes móviles en contacto con el fluido a medir.
- Calibración mecánica simple, mediante chequeo en *software* de prueba.
- Requiere poco mantenimiento.
- No hay obstrucción de la línea.
- Tolerante a gases húmedos o sucios.

Entre las desventajas de este dispositivo se encuentran:

- Requieren potencia para su operación.
- Requieren tramos de tubería recta a la entrada y a la salida o enderezadores de flujo.
- Costos de inversión altos.

1.4.2 Medidores de Flujo *Coriolis*. Las fuerzas de *Coriolis* ocurren en sistemas que rotan. Si una persona se encuentra de pie en el centro de un disco que gira y se mueve radialmente hacia el borde del disco, experimenta una fuerza lateral que intenta desviarlo de la ruta más corta. Ésta es la fuerza de *Coriolis*. Desde el punto de vista de la medición, se usa cuando el medio que va a medirse fluye a través de un tubo que vibra (figura 11). La fuerza de *Coriolis* deforma el tubo, en adición a la vibración causada por la oscilación. La deformación es proporcional al flujo másico.

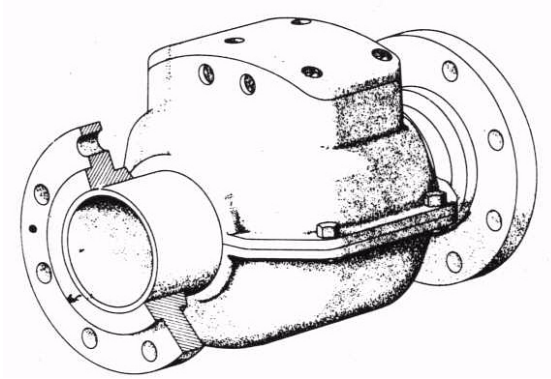
Figura 11. Medidor tipo Coriolis

Micro Motion Inc.

Los medidores de flujo que usan el principio *Coriolis*, registran el flujo másico directamente, sin la influencia de parámetros como la temperatura, la presión y la densidad. Para sustancias gaseosas, la medición de flujo másico requiere altas presiones.

1.4.3 Medidor de flujo magnético. El medidor de flujo magnético (figura 12) es del tipo no obstruccional y se basa en principios de inducción magnética. Promedia la velocidad sobre el área del tubo. El medidor opera generando un voltaje que es perpendicular a la dirección del flujo y el campo magnético se detecta por dos electrodos montados sobre un tubo no conductor. La señal en miliVolts es proporcional a la velocidad promedio del fluido en la tubería y, por tal razón, los medidores de flujo magnético son apropiados para todos los fluidos conductivos que operan en regímenes de flujo laminar y turbulento.

Figura 12. Medidor de tipo magnético



Medición y Regulación, Hernando Galvis Barrera.

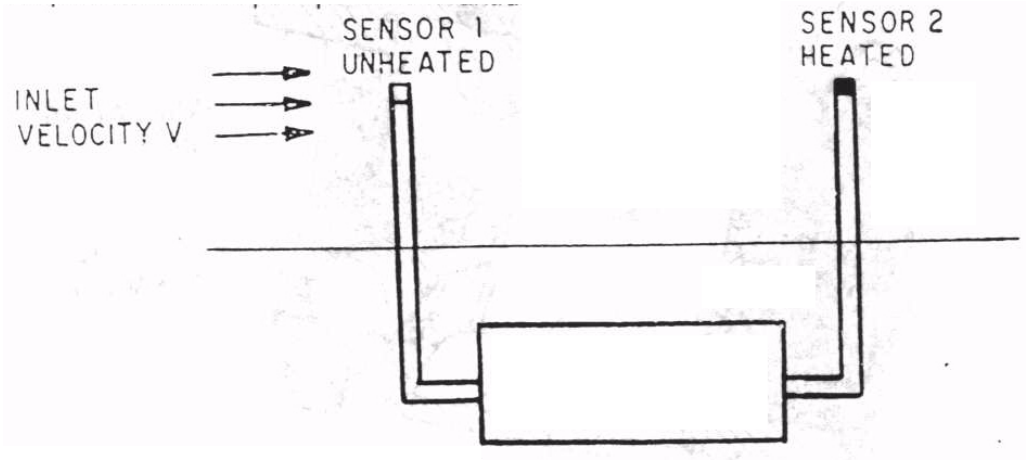
Usualmente un material no magnético (acero inoxidable) provee la integridad estructural necesaria, mientras que un recubrimiento no conductor de teflón, caucho o cerámica aísla eléctricamente el voltaje generado de la estructura del soporte. Algunos medidores están hechos de fibra de vidrio, pero éstos tienen características nominales de baja presión. Las cubiertas del medidor están hechas de materiales magnéticos con bobinas instaladas en su interior y recubiertas con un revestimiento aislante delgado. Si se seleccionan adecuadamente los revestimientos, los medidores de flujo magnético son adecuados para medición de lodos, flujos sucios, líquidos corrosivos y otros líquidos difíciles de manejar.

Estos medidores son de alto costo de inversión y operación. Debido a que deben usarse para medir fluidos conductivos estos medidores no son apropiados para el sector de hidrocarburos y gases.

1.4.4 Medidores termodinámicos. Estos medidores se basan en el mantenimiento de una temperatura (o una diferencia de temperatura) constante entre dos puntos separados, dentro del flujo de gas. El calor que se extrae (por medio del gas que fluye) es compensado agregándole calor producido eléctricamente (figura 13).

Este sistema de medida reacciona rápidamente a los cambios en el flujo de gas (10 a 200 milisegundos), pero es susceptible a formación de sedimentos en el sensor y a los cambios en el perfil del flujo. Para cada gas se debe calibrar el medidor individualmente. Se usan preferiblemente para pequeñas cantidades de gas.

Figura 13. Medidor termodinámico



Medición y Regulación, Hernando Galvis Barrera.

2. EVOLUCIÓN TECNOLÓGICA DE LOS COMPUTADORES DE FLUJO

El interés original en un sistema de cálculo electrónico de flujo apareció del requisito de proveer datos instantáneos en tiempo real para la industria del gas.

Inicialmente, el objetivo de la industria del gas natural era la eliminación de los problemas asociados con el uso de cartas para medidores de orificio. Estos problemas se debían básicamente a la demora involucrada en envío postal y procesamiento de la carta y en cálculos de volumen. Se requería un programa de computador sofisticado para desarrollar estos cálculos. Esto usualmente consumía entre cinco y ocho días para devolver el volumen de la carta debido a los requisitos de envío, procesamiento y tiempo de computación.

2.1 COMPUTADOR DE FLUJO ANÁLOGO

Los primeros computadores de flujo fueron dispositivos análogos, en los cuales, los voltajes proporcionales al flujo y las constantes se colocaban a la entrada de amplificadores operacionales. La salida resultante era un voltaje proporcional a la velocidad del flujo. El voltaje que representaba el flujo podía ser leído en un medidor o registro, o enviado por telemetría a otro lugar.
Los computadores de flujo análogos no deben ser interpretados como dispositivos computacionales inferiores. Los circuitos análogos modernos dan excelente exactitud en cómputo y ofrecen estabilidad en un ambiente de temperatura variable. La mayoría de las unidades son de tamaño pequeño y de costo competitivo. Estos equipos constan de un sistema de acondicionamiento de señal de entrada, un circuito multiplicador/divisor y un extractor de raíz cuadrada para proveer el cálculo del volumen y normalmente, están disponibles para platina de orificio con transductores DP (Diferential Pressure). Algunas unidades tienen una entrada de un medidor de densidad para medida de masa, en lugar de las entradas usuales de presión y temperatura.

2.2 COMPUTADOR DE FLUJO DIGITAL

Combinar circuitos con transistores en un simple dispositivo llamado circuito integrado, abrió la mayor brecha en el campo de la electrónica. Muchos circuitos sencillos de transistores como las puertas lógicas pueden ser localizados en una única oblea de silicio. Estas obleas son referidas con frecuencia como “chips”. Los circuitos más complejos, tales como amplificadores operacionales, también se montan en un único “chip”. Las

puertas lógicas se combinan para formar los llamados "flip-flops" y estos pueden ser usados como dispositivos de memoria o contadores. Varias combinaciones de "flip-flops" se han puesto en un único circuito integrado para desarrollar funciones específicas y a través del uso de estos circuitos se desarrollaron los computadores de propósito especial.

2.2.1 Basados en circuitos digitales. De esta forma, mientras los computadores de flujo análogos empleaban circuitos integrados lineales como los amplificadores operacionales, los computadores de flujo digitales usan circuitos integrados digitales tales como: puertas lógicas, "flip-flops", contadores, etc., para controlar el funcionamiento de su calculadora interna.

El advenimiento de esta electrónica digital, mejoró la exactitud de los cálculos de flujo "in situ". Con los computadores digitales, las señales de corriente o voltaje que representan los datos de flujo, se colocan a la entrada de convertidores análogo/digitales. Estas entradas ahora digitalizadas, se convierten a unidades de ingeniería y los cálculos se hacen de forma digital. Los primeros calculadores digitales llevaban a cabo los cálculos como los haría un individuo que usa un portátil. La secuencia de operaciones se controlaba por la forma en la cual, los componentes electrónicos eran diseñados en los circuitos. La salida de la calculadora era la tasa de flujo expresada en unidades de ingeniería y una salida de pulso que representaba unidades volumétricas. Esta tasa podía ser convertida a un valor análogo por un conversor digital/análogo y presentada como una corriente o voltaje.

2.2.2 Basados en microprocesador. La era de los microprocesadores no solo ha introducido muchos cambios en las aplicaciones de medición de gas, sino que ha afectado la vida del día a día. Pero, qué es un microprocesador?. Los microprocesadores han sido referidos frecuentemente como "un computador en un chip".

El advenimiento del microprocesador (µP) mejoró todo el sistema, ya que el µP es controlado por un programa o grupo de instrucciones más que por el diseño del circuito, como sucedía en las primeras calculadores digitales. Esta característica de los computadores de flujo basados en microprocesador, los hace más flexibles que los primeros computadores digitales y los dota de una potencia única para manejar procesos y cálculos más complejos.

Hoy en día, la mayoría de fabricantes de computadores de flujo utilizan microprocesadores y el método de visualización de resultados varía. Ya que, todos los computadores de flujo de gas están resolviendo las ecuaciones básicas de flujo, la diferencia está en el empaque, las técnicas de programación y la forma como los datos de entrada y salida se manejan. Estas características se deben observar cuando se quiera comprar un computador de flujo de gas con el fin de seleccionar la unidad más adecuada para las aplicaciones.

Como el microprocesador es programable, su software puede ser expandido para incluir otras caracterí sticas además del cálculo del flujo, como incluir rutinas para monitorear entradas análogas. En caso de falla del transductor, se puede iniciar alguna acción, tal como desplegar el estado o aproximar la entrada hasta que el dispositivo sea reparado. El microprocesador se puede usar para controlar varios dispositivos en una estación de medida. La salida del microprocesador se puede ligar a un posicionador en una válvula de control para controlar la velocidad del flujo, diferencial o presión.

2.2.3 Software y memoria. Tal como los computadores análogos se controlan por el diseño del circuito, el microprocesador debe ser controlado por su programa. El microprocesador debe ser visto como la unidad central de proceso (CPU). El programa o software es una secuencia de instrucciones ejecutadas con el fin de desarrollar una tarea especí fica. Los computadores análogos y los primeros computadores digitales fueron controlados por hardware. Es obvio que es más fácil cambiar la secuencia de ejecución de instrucciones que modificar el diseño del circuito de un computador.

Los computadores digitales de flujo pueden ser programados por el fabricante para efectuar muchos programas, todos en milésimas de segundo. Ya que el microprocesador es controlado por software, cada variable puede ser manipulada con su propia ecuación y luego puesta de vuelta en la ecuación de flujo. Los microprocesadores digitales usados en campo, pueden tener programas fijos diseñados solo para una ecuación especí fica. Esto permite al computador de flujo basado en microprocesador actuar como un instrumento dedicado al propósito de la medición de flujo.

Como el microprocesador debe tener un programa para controlarlo, también debe tener un medio para almacenarlo. El frecuentemente usado es una memoria de solo lectura ROM (Read-Only Memory). Como su nombre lo indica, esta memoria es fija y solo puede ser leí da. Hay muchos tipos de ROM's disponibles. Un tipo es la ROM hecha a la medida, en la cual el programa es incorporado dentro de la estructura de la ROM; éste no puede ser cambiado o alterado de ninguna forma. Otro tipo es la ROM programable o PROM. Otro tipo de ROM es la ROM borrable o EPROM. Este tipo se programa eléctricamente y puede ser borrado exponiendo el "chip" de silicio a luz ultravioleta. Después de ser borrada la EPROM puede ser reprogramada. Actualmente se usa también la memoria EEPROM o E^2PROM que puede ser borrable eléctricamente con todas las ventajas que esto conlleva. Otro medio de almacenar el programa es usar una memoria de acceso aleatorio o RAM. Además de leer de la RAM, el microprocesador puede escribir en ella. Los dispositivos RAM no mantienen su memoria cuando se les retira la energí a; por esta razón, no son adecuados para el almacenamiento permanente del programa. Normalmente son usados para almacenamiento de datos dinámicos usados por el microprocesador durante la ejecución del programa.

2.2.4 Consumo de potencia. Un factor en la resistencia al uso de los instrumentos basados en el microprocesador fue la falta de confiabilidad, provocada por muchas fuentes. Una de ellas fue que, típicamente, el dispositivo disipaba significativas cantidades de potencia. Como resultado de esto, cuando los usuarios lo ubicaban en ambientes no controlados, la alta disipación de potencia acoplada con la alta temperatura del ambiente causaban fallas.

Actualmente, los desarrollos tecnológicos en este aspecto han hecho que un instrumento que formalmente requería 40 Watts para la operación, requiera solo 10 Watts. Esta disminución de potencia viene acompañada con un aumento significativo en la capacidad. Además, usando chips CMOS (circuitos integrados de bajo consumo de corriente), ahora es posible producir instrumentos basados en microprocesador que requieren mucho menos de un watt para operar. De esta forma, se han removido muchas objeciones por los avances en la tecnología electrónica. En particular, por el uso de tecnología electrónica de potencia extremadamente baja. La circuitería CMOS se puede emplear en tecnología avanzada de microprocesador para medición de flujo de gas natural en sitios donde no hay potencia disponible.

2.3 ESTADO DE ARTE DE LOS COMPUTADORES DE FLUJO

Actualmente, en aplicaciones de transferencia de custodia se está logrando tasas de flujo en tiempo real, alta exactitud y precisión.

Como el computador de flujo es considerado como la "caja registradora", es imperativo que mantenga o adquiera la capacidad de interconectarse a diversos dispositivos.

La funcionalidad básica de los computadores de flujo tiende a permanecer constante de generación en generación. Sin embargo, se han hecho avances significativos en los últimos años en las siguientes áreas:

- Tecnología de visualizadores.
- Flexibilidad de las aplicaciones.
- Facilidad de uso (mejoras en la interfase humano-máquina).
- Arquitectura.
- Configuración.
- Velocidad y complejidad de los cálculos realizados.
- Opciones de entrada-salida (E/S).
- Opciones de comunicación.

- Opciones de almacenamiento de datos.

2.3.1 Computadores de flujo de alta tecnología. A continuación se describirá las principales características de un computador de flujo de la última generación creado por la empresa Controles de Flujo Hoffer en Carolina del Norte (USA), quien ha introducido el "Nova Flow" como el primer computador "plug-and-flow". Éste fue diseñado por Hoffer y Chronotek de Houston, Texas. Nova Flow es configurable como un totalizador de velocidad, o computador de flujo a través del uso de una arquitectura del sistema que solo había sido vista antes en dispositivos de control muy sofisticados y costosos.

Para aplicaciones y opciones más avanzadas, Nova Flow tiene ocho ranuras de expansión para colocar módulos electrónicos, los cuales proveen opciones de Entrada-Salida (E/S) y de comunicaciones. Se puede escoger entre muchos módulos opcionales para ajustarse de la mejor manera a una aplicación particular. La configuración de la unidad básica y de los módulos se implementa usando un software basado en Windows, el cual se incluye con la unidad.

Uno de los módulos de expansión permite configurar la unidad para entradas de presión y temperatura o directamente de densidad para masas de fluido corregidas en líquidos y gases. La entrada del medidor de flujo puede ser una bobina magnética, TTL (Logica Transistor-Transistor), colector abierto o contacto seco. El tipo de entrada del medidor de flujo se selecciona por medio de puentes sin cambiar las conexiones cableadas. La unidad básica incluye ocho líneas adicionales para E/S (entrada-salida) digital que se configuran por medio de software. Una salida digital aislada ópticamente viene estándar con el computador. Nova Flow soporta muchos formatos digitales, incluyendo 0–5, 0–10, o de colector abierto hasta 30 VDC y 250 mA.

Los módulos opcionales actualmente disponibles incluyen:

- El módulo de entrada análoga de la RTD (Dispostivo para medición de temperatura) para más de dos RTDs y dos entradas análogas que se usan en aplicaciones de volúmenes y masas de fluidos corregidas.
- El módulo cuádruple de entradas análogas, con 4 entradas de 12 bits configurables por el usuario y protegidas contra sobrecorriente y sobretensión.
- El módulo de salida análoga, con dos salidas análogas configurables por el usuario, éstas tienen 12 bits de resolución y soportan 4–20 mA y 1–5 VDC
- El módulo de relé, con dos relés de lanzamiento sencillo o doble, configurables por el usuario a 175 V_{max} y 5 A I_{max}
- Los módulos RS-232 y RS-485. Los módulos RS-485 también soportan aplicaciones con Modbus (protocolo de comunicaciones).

- El módulo digital de E/S, con siete líneas adicionales para E/S digital configurable por el usuario.
- El módulo de medidor de flujo, el cual soporta una segunda entrada de medidor de flujo con el mismo rango de señales de entrada, como la unidad básica. Este módulo soporta dos corrientes de flujo independientes para aplicaciones volumétricas no compensadas. La compensación total de presión y temperatura en aplicaciones de masa de fluido se logra con este módulo combinado con el de entrada análoga de RTD. Este módulo también provee cuatro entradas en cuadratura, usadas para aplicaciones bidireccionales y chequeo de pulsos en aplicaciones de transferencia de custodia.
- El módulo de almacenamiento de datos, con capacidad de memoria de 256 KB o 512 KB. Todas las funciones de almacenamiento pueden ser configuradas por el usuario. Se puede acceder al módulo para transferencia de datos por medio de un puerto serie.
- El módulo de red industrial, que incluye el circuito de interfase y el software para varias redes y protocolos "fieldbus".

Con la selección de módulos se puede especificar lo que se requiera y mantener flexibilidad para actualizaciones o mejoras futuras. Hoffer estima que con la unidad básica y los módulos se puede configurar el computador en más de 5,000 formas diferentes.

2.3.2 Capacidad de cómputo. Como muchos productos de alta tecnología, los computadores de flujo industriales deben la mayoría de sus mejoras y crecimiento de mercado a la disponibilidad de microprocesadores más rápidos y componentes de memoria más grandes a precios siempre decrecientes. Actualmente las CPUs de 16 bits son más, que una excepción, la norma para estos dispositivos. Los relojes y calendarios en tiempo real son estándar y los computadores se adaptan al Y2K (año 2000). El incremento en la potencia de cómputo, la velocidad y su capacidad de memoria son los factores que guían al desarrollo del computador de flujo multipropósito.

Estos avances permiten expandir el diseño básico del computador y extender su vida. Seleccionando los microprocesadores y sus periféricos, fácilmente se puede adaptar el computador de flujo básico a nuevas aplicaciones, a medida que ellas aparecen. Ahora es factible pensar en términos de computadores de flujo hechos al gusto del comprador, producidos en pequeñas cantidades para un usuario o mercado específico.

2.3.3 Opciones de entrada-salida (E/S). En los años noventa se podían contar con los dedos de la mano el número y tipo de señales de E/S que soportaba un computador de flujo. Hoy en día, aún hay computadores con opciones de E/S limitadas, pero la tendencia va hacia más opciones. La mayoría de los nuevos computadores de flujo soportan ahora una variedad de entradas directas de sensores de flujo, incluyendo pulsos de DC (corriente directa), TTL (Lógica Transistor-Transistor), colector abierto, captación

magnética, contacto seco y análogas. Las entradas directas de RTD son características estándar para unidades que compensan temperatura, así como lo son las señales análogas para entradas de presión y densidad. En muchos casos, el software de instalación permite asignar y configurar las opciones de E/S sin tener que cambiar el hardware.

Algunos computadores de flujo pueden ahora calcular y tomar caudales de más de un medidor de flujo. Por ejemplo, puede aceptar y procesar dos entradas de medidor de flujo y dos grupos de entradas de presión y temperatura para determinar el flujo de masa compensado de dos corrientes de diferentes. Esta capacidad estaba disponible únicamente en los computadores de flujo más sofisticados y costaba dos o cuatro veces más que los nuevos sistemas.

2.3.4 Comunicaciones digitales. Históricamente las comunicaciones digitales para computadores de flujo han estado limitadas a RS-232 o RS-485. Los nuevos computadores de flujo pueden tener también interfases con redes industriales estándar. Hasta la fecha, la red más común con la que se hace interfase es Modbus, pero muchos fabricantes están desarrollando soporte para Profibus, DeviceNet, y Fieldbus. También se ha empezado un trabajo preliminar para soportar sistemas Ethernet. Estas nuevas opciones representan un gran salto en la capacidad e interoperabilidad de los computadores de flujo.

Además de las opciones de comunicación cableadas, una tendencia creciente es la disponibilidad de enlaces para comunicaciones IR (infrarrojo) para operación y configuración remota de computadores de flujo. Los enlaces están disponibles en una base limitada pero estarán ampliamente disponibles en los próximos años.
El protocolo usado comúnmente para transmisión IR es el IrDA, desarrollado por la Asociación de Dispositivos Infrarrojos. Este es el mismo protocolo usado en la mayoría a de productos electrónicos. La aplicación típica del IR en el mercado industrial, ha sido su uso en áreas inflamables o peligrosas. Así, un comunicador IR portátil es usado para establecer enlaces con computadores de flujo montados en encapsulados a prueba de explosión. Estos computadores de flujo tiene una ventana IR en su panel frontal a través de la cual recibe y transmite al dispositivo portátil.
Otra opción que aparece en los nuevos computadores de flujo es el módem telefónico para conexiones a través de líneas en tierra, celulares o enlaces de radio. Típicamente estos módems se ofrecen como unidades externas. Esta configuración es debida a la parte económica, ya que es más barato proveer un módem estándar disponible que construir uno que se ajuste a la parte interna del computador de flujo. La necesidad de cumplir los límites EMI y RFI para ciertos estándares, así como las clasificaciones de seguridad intrínseca están manejando esta tendencia.

2.3.5 Adquisiciòn y almacenamiento de datos. Gracias al reducido costo de las memorias de computador, muchos fabricantes de computadores de flujo están ofreciendo almacenamiento básico de datos como una característica estándar u opcional. La memoria para almacenamiento de datos usualmente cae en el rango de 256–516 KB. Aunque es pequeña en comparación con los almacenadores de datos industriales, esta capacidad es adecuada para la mayoría de aplicaciones de computadores de flujo. Los datos pueden ser descargados cada cierto tiempo en un sistema de control y memoria más grande mediante enlaces por algunas de las opciones de comunicación.

La configuración de la función de almacenamiento de datos se lleva a cabo por medio del software de instalación provisto con el computador. La mayoría de las entradas, salidas o parámetros calculados pueden ser almacenados, y la frecuencia de almacenamiento es determinada por el usuario. Las funciones especiales (ej: almacenamiento de valores pico), están disponibles en la mayoría de sistemas de almacenamiento.

2.3.6 Descripción de un Daniels. A continuación se describirá otro computador de flujo de la empresa Daniel Industries, Inc. llamado el Solarflow, lo cual ayudará a dar sencillas pautas durante el proceso de diseño.

El computador de flujo SolarFlow es un instrumento basado en microprocesador que se alimenta con baterías, las cuales se mantienen y se cargan con un panel solar. Este computador viene en dos versiones – una para uso con medidores tipo pulso (desplazamiento positivo y turbina) y otra para uso con medidores de orificio. Aunque el paquete de hardware es el mismo y las funciones y operaciones son muy similares, la versión descrita será la del computador para medidor tipo orificio.

El SolarFlow es un computador de flujo con microprocesador capaz de operar en ambientes hostiles, con total independencia de la alimentación externa. El computador de flujo se suministra como un paquete completo que incluye los transductores requeridos. La potencia de operación deriva de una combinación entre un panel solar y un paquete de baterías recargables.

El sistema deriva considerablemente menos potencia de la batería totalmente cargada, y provee ocho días de operación sin luz disponible. El ajuste en el panel solar se provee para el ángulo azimutal y el ángulo de elevación. El rango operacional de temperatura del instrumento es de -20° a +160°F.

El programa de control que gobierna el instrumento está en una memoria de solo lectura (ROM) y los parámetros que introduce el usuario se almacenan en una memoria programable (RAM). Provee comunicación con un terminal de datos portátil usado para toda la programación y para las funciones de calibración y recuperación de información. Las entradas del proceso provienen de transductores especiales de baja potencia y están multiplexadas en un conversor análogo/digital de alta precisión. Se miden tres tipos de variables de proceso. Presión estática, presión diferencial y temperatura. El instrumento es

capaz de usar dos transductores de presión diferencial para ampliar el rango. Como elemento de temperatura se usa una RTD de platino de 500-ohm.

Dadas las entradas de presión, presión diferencial y temperatura, el SolarFlow resuelve la ecuación de flujo de gas según el AGA3 y AGA8. La exactitud es menor que 1% de fondo de escala con un cálculo cada dos segundos (43,200 por día).

En primera instancia la comunicación con el computador se hace a través del terminal de datos portátil, que indicará al operador y en unidades de ingeniería, las entradas requeridas en la secuencia correcta. Solo la información requerida se preguntará, y el computador no permitirá que los parámetros necesarios se omitan. Se solicitará la información requerida hasta que se reciba. El "display" del terminal de datos se puede usar para indicar flujo, flujo totalizado, presión, presión diferencial, y temperatura en unidades de SCFH, SCF, PSIG, pulgadas de agua, y grados F, respectivamente. También se pueden visualizar todos los factores. El computador también indica presión, presión diferencial y temperatura en su propio "display" LCD (pantalla de cristal liquido) de 8 dígitos. La velocidad del fluido y el flujo totalizado también se pueden visualizar. El "display" LCD sirve para indicar condiciones de alarma, tales como transductores fuera de rango. Un número único de clave de acceso por usuario, da protección contra acceso no autorizado a los datos, y el computador es a prueba de las condiciones ambiente.

Cada SolarFlow almacenará 35 días de totales, presión promedio, temperatura, presión diferencial, en una base de contrato diario.

Cada terminal portátil cargará y almacenará todos los 35 días de datos de más de 16 computadores o un número mayor de computadores con 10 días. El terminal de datos se puede conectar a una impresora. Cada reporte de la compañía se puede imprimir individualmente o todos a la vez, secuencialmente por el número de ID.

En este dispositivo, no se requieren ajustes mecánicos o eléctricos. La presión diferencial se compensa por cero a presiones elevadas y el transmisor de presión estática se debe linealizar sobre tres puntos. La calibración requiere solo una persona. El computador alimentado solarmente fue diseñado en primera instancia para medición de gas por orificio, pero se puede usar con cualquier dispositivo primario y virtualmente con cualquier gas. También hay modelos similares disponibles para medidores tipo pulso. El microprocesador de baja potencia con una fuente solar/batería, provee un medio robusto y de costo efectivo para emplear el poder de los computadores digitales en las estaciones de medición remotas, aún en áreas donde hay días en los cuales no prevalece el sol.

3. DISEÑO PRELIMINAR

El diseño de un computador de flujo está sujeto a diversos estándares e involucra gran desarrollo electrónico y de software. No obstante, sus partes están bien definidas y se requiere implementarlas con el mayor rigor electrónico posible, buscando la mejor exactitud y teniendo en cuenta todos los detalles que exige la creación de un sistema de medida totalmente autónomo.

A continuación se darán algunas pautas preliminares y conocimientos consultados al respecto de algunas unidades. Luego, en el siguiente capitulo y con base en estos conceptos, se expondrá con más detalle el diseño del hardware implementado.

3.1 CONSIDERACIONES PRELIMINARES

Este computador como dispositivo de medición y registro del flujo de gas, deberá constar de las siguientes partes:

- Unidad de Medición: Consta de los sensores y transmisores pertinentes según el instrumento primario de medida, así como de la adecuación de las señales que de estos surjan. Esta unidad deberá ser de gran resolución y exactitud pues ella contiene un gran porcentaje del error de incertidumbre total del sistema.

- Unidad de Procesamiento y Control: A esta unidad se le atribuye la tarea de coordinar todo el sistema para que funcione como un dispositivo autónomo de campo. Así mismo, procesará todas las variables de medición y efectuará todos los cálculos siguiendo los parámetros dados por los estándares de medición del AGA. Consta tanto de hardware como de software, a este último se le llamará firmware ya que es un software de bajo nivel, y a su vez para distinguirlo del software de visualización y presentación que se expondrá más adelante y que se desarrolla en un PC o laptop convencional.

- Unidad de Interfase: Ofrece al usuario la posibilidad de leer las variables de interés e introducir o modificar datos o configuración del sistema. Consta de un visualizador y de un teclado. Estas partes deberán garantizar facilidad y comodidad al operador, así como seguridad y larga vida al dispositivo.

- Unidad de Almacenamiento: Tiene la función de registrar y guardar todas las variables de interés por el tiempo que se estime conveniente. Deberá evitar la pérdida o corrupción de datos, así como garantizar el suficiente tamaño de memoria para albergarlos por el tiempo asignado.

- Unidad de Software: Esta unidad es externa al computador de flujo pero complementaria. Es la que permite mediante un software de comunicación y presentación, descargar a un PC los datos corregidos por el computador de flujo y desplegarlos amena e interactivamente al usuario. Adicionalmente, puede involucrar una base de datos y un manejo estadístico de la información.

3.2 UNIDAD DE MEDICIÓN

Ampliando el contenido de esta unidad se definen tres partes:

- Los transmisores: quienes por medio de un transductor, convierten una señal física en una eléctrica y la transmiten al computador de flujo.

- El acondicionador de señal: encargado de adecuar esta señal eléctrica, depurarla, y aislarla o amplificarla si es necesario, y así dejarla en un nivel óptimo para ser digitalizada.

- El conversor: Encargado de procesar la señal análoga acondicionada y convertirla en una señal digital para ser procesada matemáticamente.

3.2.1 Sobre los Transmisores. Los transmisores electrónicos son necesarios para convertir las señales de los medidores primarios a señales de corriente o voltaje para el computador de flujo. Estas señales, para el caso de una medición con platina de orificio, vienen de la presión diferencial a través del orificio de la platina, de la presión estática de la línea y de la temperatura del fluido.
La mejor exactitud en las mediciones y la facilidad de calibración confiable, son objetivos constantes en un mundo con retos tecnológicos que ha estado usando señales estándar de 4-20mA. La mayoría de los transmisores de campo, que usan capacitancia y resistencia, se comunican con el computador de flujo por una señal de 4-20mA. Sin embargo, solo una cantidad limitada de información (la variable medida) puede ser representada con esta señal.
El último concepto en tecnología son los transmisores digitales o inteligentes (Smart Transmitters), que tienen la capacidad de proveer más de una variable de medida en el tiempo, conllevando a reducción de costos y mejor exactitud.
Un beneficio importante del transmisor digital, es la capacidad de proveer más de una variable medida en el tiempo. La comunicación digital puede reducir significativamente los costos de conectar múltiples dispositivos de medida en un sistema de control de procesos distribuido. Un ejemplo de estos

transmisores son la serie ST3000 de Honeywell que usan un protocolo llamado Honeywell´s DE.
Los transmisores inteligentes mejoran la exactitud al eliminar las conversiones análogo/digitales, ya que se comunica con una señal digital directa. Sin embargo, los protocolos digitales pueden gastar tiempo en enviar toda la información del proceso al sistema maestro. Esta demora puede ser de más de un segundo. Esto puede causar una gran incertidumbre en la medida y pueden así solo dar una pequeña o nula ventaja sobre el transmisor de señal de 4-20mA.
Los mayores obstáculos del mercado para aceptar los transmisores digitales y sus protocolos usados por fabricantes individuales, son la ausencia de estándares comunes y de técnicos de campo suficientemente capacitados en varias marcas mundiales, que puedan adaptarse a los cambiantes requisitos de calibración y demandas de nuevas tecnologías.
Un estándar común aseguraría el desarrollo de los sistemas con arquitectura abierta. Un estándar común en la industria, como el de los transmisores análogos, aseguraría que cualquier producto de cualquier fabricante pueda trabajar con confiabilidad y seguridad junto a otro dispositivo. El nuevo estándar digital que está siendo desarrollado se conoce como Fieldbus.
Por esta falta de compatibilidad y estandarización de protocolos se recomienda usar transmisores análogos de señal de 4-20mA para el desarrollo del presente proyecto; los cuales continúan ofreciendo la exactitud requerida por los estándares americanos de medición (AGA y API). En un futuro cuando ya se tenga más soporte para los transmisores digitales se pudiese incursionar en este campo si fuera necesario.
Los transmisores más comunes para la medición del flujo de gas son el de presión, presión diferencial y temperatura.

3.2.1.1 Transmisores de Presión. Un transmisor consta principalmente de un transductor el cual convierte un tipo de señal en otro. En este caso la presión en un voltaje. Este transductor consta de una galga extensométrica, la cual es un dispositivo resistivo delgado que varía su resistencia de forma directa con un cambio en su longitud. Son comúnmente usados en grupos de a cuatro conocidos como puentes de "Wheatstone". El puente de galgas extensométricas da un pequeño voltaje de salida por lo que se requiere un amplificador para obtener una señal utilizable. Los transductores modernos de presión, que incluyen los transductores de presión diferencial, requieren un puente de galgas extensométricas, un amplificador y un "driver" de corriente en un estuche a prueba de explosión.

3.2.1.2 Transductores de Temperatura. El transductor de temperatura más lineal y estable, es el dispositivo termómetro de resistencia de platino, conocido como una RTD.

Esta unidad consiste de una sonda acoplada a la corriente de gas por medio de un termopozo. Esta resistencia de platino es instalada de forma similar al transductor de presión, ya que ella es un "brazo" de un puente activo de cuatro "brazos" (como el de Wheatstone) y como tal, requiere un amplificador y un "driver" de corriente. El sensor y el amplificador son abastecidos dentro de la misma unidad.

Teniendo en cuenta que estos transmisores son dispositivos análogos, enunciar especificaciones de exactitud a escala total, tal como $\pm$0.25%, puede causar un problema. Por ejemplo, un transductor de presión de 0 a 1000 PSIG con la anterior exactitud a escala total puede tener una incertidumbre estimada de $\pm$2.5% al 10% de su escala total o 100 PSIG.

Para mayor exactitud con estos dispositivos, se deben operar siempre cerca de su escala total o al menos a la tercera parte más alta.

3.2.2 Sobre la Conversión Análogo-Digital. Las señales de los transmisores son acondicionadas y deben digitalizarse para ser procesadas matemáticamente; esta es la tarea del conversor análogo-digital de la unidad de medición. Por tanto, es necesario tener en cuenta una serie de aspectos en el uso del conversor análogo-digital, algunos de los cuales se expondrán a continuación.

Un frecuente equívoco que se encuentra al tratar los conversores análogo-digitales (ADCs), es que la selección de un conversor AD con alta resolución de bits no necesariamente garantiza un incremento proporcional en la exactitud. Alta resolución no es lo mismo que alta exactitud.

La alta exactitud del ADC se determina por su estabilidad, linealidad y repetibilidad. Naturalmente se espera una mejoría usando un conversor AD de 16 bits en vez de uno de 12 bits. Sin embargo, ninguna solución mejorará la exactitud del resultado medido, si el periodo de muestreo es bajo o si hay una pobre compensación de los cambios por temperatura en el margen del ADC.

3.2.2.1 Muestreo. Cuando las condiciones del proceso están cambiando rápidamente, el paso de cambio entre lecturas tomadas por un sistema de muestreo A/D lento, puede ser mucho más grande que la precisión del transmisor análogo o del ADC del computador de flujo. La temporización asociada con el muestreo de la señal y su método para promediar es tan significativo como la resolución del ADC.

Para efectos de este proyecto, las tasas de muestreo serán guiadas por el estándar API 21, el cual está dedicado a este respecto y del cual se tratará en posteriores informes.

3.2.2.2 Temperatura en los ADCs. El supuesto beneficio de un ADC de alta resolución desaparecerá si ocurren derivas significativas cuando el ADC se expone a variaciones de temperatura ambiente. Si el ADC no está adecuadamente compensado para minimizar los errores de "offset de cero" y "span", la mejora de incrementar la resolución del ADC se anulará debido a cambios en la temperatura ambiente. Para una mejora significativa en exactitud, una resolución más alta debe estar acompañada de una mejora congruente en estabilidad de temperatura y del circuito análogo en general.

3.2.3 Sobre la Incertidumbre. Incertidumbre del bit menos significativo del ADC. Un ADC convertirá una señal análoga en '2n' pasos discretos o valores digitales (donde 'n' es igual a la resolución de bits del ADC). Un ADC de 14 bits provee 16,384 valores digitales discretos, mientras que el de 16 bits produce 65,536 valores digitales discretos. No todos estos valores están disponibles para medir rangos de señales análogas de entrada tales como 1-5 volts. En muchos casos, el ADC debe ser capaz de medir señales que tienen 5% o más de sobrerango (por ejemplo, 5.25 volts en lugar de 5 volts). Algunos de los rangos de valores del ADC se usan también para medir el cero elevado (1 volt), el cual es común en medición de procesos. El resultado es que solo el 76% del número total de valores discretos del ADC puede ser usado para medir la señal de 1-5 voltios.

Al calcular, los ADCs de 14 y 16 bits tendrán aproximadamente 12,500 y 49,800 valores digitales discretos respectivamente. Sin embargo, el porcentaje de diferencia en la lectura, producida por un cambio en el bit menos significativo 'LSB', de uno de 14-bit es 0.008% [(1/12500)*100], y uno de 16-bit es 0.002%[(1/49800)*100].

- *Incertidumbre de un Transmisor.* La incertidumbre en la lectura de un transmisor de proceso (+/-0.075%), no es impactada significativamente por la incertidumbre adicional dada por el cambio del LSB usando un A/D de 14 bits (0.008%) o uno de 16 bits (0.002%). La incertidumbre añadida a la variable del proceso medida, causada por el equipo de calibración y procedimientos operacionales también debe ser considerada.

Se asume que el equipo usado para calibrar transmisores de proceso no es sensible a efectos de temperatura y será calibrado contra un estándar que se puede trazar. Los supuestos adicionales incluyen que el equipo será exacto en una magnitud más grande que el transmisor de proceso.

3.3 UNIDAD DE CONTROL Y PROCESAMIENTO

Siendo esta unidad el corazón del sistema, fue necesario dejar establecido desde el inicio de este proyecto los criterios de selección y escoger definitivamente la unidad a utilizar.

Los criterios establecidos para la selección de esta unidad fue:

- Sistema basado en 16 bits o superior.
- Manejo de punto flotante para cálculos matemáticos.
- Fácil adaptación de periféricos, sensores, conversores.
- Bajo consumo de potencia.
- Rango de temperatura dentro de los límites de trabajo.
- Reducido tamaño que se ajuste a los promedios del mercado estándar.
- Suficientes entradas y salidas para el rango necesario de instrumentación.
- Capacidad suficiente de manejo de memoria y almacenamiento de datos.
- Versátil soporte para displays y teclados.
- Mínimo de dos puertos de comunicaciones con posibilidad de expansión.
- Buen posicionamiento en el mercado que asegure su continuidad de producción.
- Interactivo sistema de desarrollo que facilite la rápida implementación del código o firmware.

Con estas pautas, se escogió de la empresa inglesa Triangle Digital Services (TDS) un sistema embebido referenciado como TDS2020F. Este sistema embebido es un conjunto de periféricos básicos controlados por un microprocesador, que a fin de cuentas, actúa como un microcomputador de propósito especial. Este conjunto ofrece los requisitos básicos para fabricar un sistema o dispositivo autónomo. No tiene aplicación específica, pues es un sistema de arquitectura abierta y se ajusta de acuerdo al usuario y a su aplicación de control, medición o almacenamiento de datos. Adicionalmente, se observó el historial de 20 años de la compañía así como una serie de equipos y dispositivos que diversas empresas mundiales fabricaron con base en esta tarjeta.

De esta forma, a continuación se detallan las características de este sistema embebido escogido y seguidamente se exponen los principios de su lenguaje de programación llamado Forth. El cual, siendo óptimo para sistemas embebidos basados en microcontrolador o microprocesador, solo es conocido por dedicados programadores de hardware y no es tan popular en el ámbito académico.

3.3.1 Sistema Embebido TDS2020F. El TDS2020*F* es un poderoso sistema embebido o microcomputador de control de 16 bits. A pesar de su pequeño tamaño y su bajo consumo de potencia, está equipado con importantes características que facilitan su uso para resolver problemas de control y recopilación de datos.

Figura 14. Paquete de Desarrollo TDS2020F

La rápida implementación de aplicaciones se facilita gracias a su naturaleza interactiva de desarrollo y depuración (encontrar y remover errores del programa). Una librería de software provee soluciones instantáneas a muchos problemas de aplicación. El código fuente, escrito en el lenguaje de alto nivel Forth, se compila directamente en una memoria no volátil Flash-EEPROM.

3.3.2 Memoria "Flash". La compilación directa en una memoria Flash-EEPROM es conveniente y evita el costo y el tiempo de desarrollo en un programador de memoria PROM. Hay muchas otras ventajas, como que se puede borrar el programa remotamente por medio de un módem y recompilarlo. Por ejemplo, si el TDS2020*F* estuviera embebido en un vehículo o en tubo presurizado, se podría actualizar el programa sin necesidad de extraer la tarjeta.

3.3.3 Lenguaje Forth. La razón de programar en Forth es que para sistemas embebidos y microprocesadores, se obtienen resultados más rápidos y eficientes (menos líneas de código) con este lenguaje. Para la máquina es un tanto de más bajo nivel que 'C' pero es un lenguaje de más alto nivel que assembler. A diferencia de otros, es interactivo, ofreciendo un rápido desarrollo. El Forth de la tarjeta es el mismo Forth Estándar Nacional

Americano con muchas extensiones para explotar las características de hardware del TDS2020*F*. El Forth de 16 kbytes incluye controladores de teclado y LCD, junto con muchas otras utilidades y un assembler totalmente simbólico. De esta forma, se pueden escribir los programas en un lenguaje de alto nivel, mezclándolo con assembler si se requiere.

3.3.4 Hardware. El TDS2020*F* está basado en el microprocesador Hitachi H8/532. Tiene 45k bytes de espacio para el programa compilado y más de 512k bytes de memoria "Flash" o RAM respaldada con batería para mantener los datos vitales mientras la tarjeta no está trabajando. Esto se puede expandir a más de un gigabyte con tarjetas PCMCIA o tarjetas "flash".

Un adaptador de red de área controlada (bus CAN) permite la interconexión rápida de los computadores TDS2020*F*, con un PC en la red si se requiere. También está provisto con los protocolos CAN de alto y bajo nivel ampliamente usados.

La tarjeta tiene entre 26 y 41 E/Ss paralelas dependiendo de la configuración y tiene dos puertos seriales RS232. Posee un conversor análogo/digital de 8 canales y 10 bits de resolución y contiene tres canales de conversión digital/análogo de 8 bits. Naturalmente se pueden conectar conversores externos de mayor resolución.
Las características adicionales incluyen cuatro contador-temporizadores en hardware (tres son de 16 bits), dos temporizadores separados de "watch-dog" (perro-guardián), reloj de tiempo del día, no volátil y multitarea. Necesita de una fuente de alimentación sencilla con 32mA de salida y además consume solo 155µA en un modo operacional de baja potencia, para aplicaciones como almacenamiento de datos.

Figura 15. Tarjeta CPU TDS2020F

El TDS2020*F* mide solo 10 x 8 cm, tiene pines para conectar con cable plano; o simplemente se puede usar como un componente insertado en una tarjeta más grande. Otra versión tiene un conector DIN41612 para usarlo en un "rack".

3.3.5 Usando Forth. Cuando se alcanza finalmente la etapa de manufactura de un producto basado en TDS2020*F* no es el final de Forth. Se puede usar para pruebas finales, reparación y mantenimiento, ya que el lenguaje está en la tarjeta.

Construyendo un conector que de acceso serial al computador Forth en el instrumento, se gana acceso al sistema de lenguaje interno con un PC o un terminal portátil. Se pueden tener disponibles todas las funciones que constituyen la aplicación.

Por ejemplo, en un termómetro electrónico, la rutina análogo/digital se puede ejercitar separadamente para ver si la falla está en la entrada del transductor o en el amplificador. Si no, se puede seguir con una prueba al LCD escribiendo unos pocos caracteres en el. Asimismo, se puede probar el teclado presionando una tecla para ver si genera el código esperado.
Si es necesario, la totalidad del código compilado se puede borrar de la Flash-EEPROM. Entonces se puede recompilar un nuevo código fuente, aún remotamente sobre un módem (o en una localización inaccesible).
Forth en la tarjeta es muy útil durante el diseño, pero la capacidad de acceder a un procedimiento de software individual en un producto terminado es única e invaluable.

3.3.6 Velocidad. La frecuencia de reloj del TDS2020*F* es 19.6608MHz y el tiempo de un ciclo es 102ns (nanosegundos). Las instrucciones se procesan en un promedio de alrededor de 3 millones por segundo (MIPS). La latencia de interrupción es solo de 2.3µs (microsegundos), más el tiempo para completar la instrucción actual.

3.3.7 Memoria.

- Dentro del Microprocesador H8/532: 32kbyte de PROM OTP (programable una vez), que lleva 16k de lenguaje de alto nivel Forth y 16k de espacio en blanco para programas adicionales de aplicación para el usuario. Además, contiene 1kbyte de RAM, de la cual la mitad se necesita para el sistema dejando 512 bytes libres para ser usados como variables para el programa de aplicación.

- "Socket" de 28 pines: En este "socket" de la tarjeta se debe insertar una memoria Flash-EEPROM de 32kbyte no volátil para el programa final del aplicación del usuario. Debido a que alguna parte del mapa de memoria se usa para los registros en el chip y E/S (entrada-salida), solo

29kbytes están realmente disponibles. Una vez se ha finalizado el software, el "socket" puede soportar una EPROM 27C256 si se desea.

- "Socket" de 32 pines: En este "socket" de la tarjeta se pueden insertar 512k, 128k o 32k de RAM, EEPROM o memoria Flash para variables adicionales, memoria auxiliar o almacenamiento de datos. Las memorias EEPROM y Flash conservan los datos en ausencia de potencia; mientras las RAM pueden hacerse no volátiles añadiendo una pequeña tarjeta de batería o una celda externa de litio. Las señales necesarias para expansión están disponibles en la tarjeta para extender indefinidamente la memoria fuera de la tarjeta.

- Chip de reloj: El chip de reloj en la tarjeta tiene 239 bytes de RAM libre que son accesibles a través del bus I^2C. Con batería externa, esta RAM (también como el reloj de tiempo real en cualquier RAM en el "socket" de 32 pines) se mantendrá en ausencia de potencia.

3.3.8 Entrada salida paralela. La capacidad del puerto paralelo del TDS2020*F* depende si usan o no el conversor AD y otras facilidades. La tabla 1 muestra solo algunos de los posibles esquemas de entrada salida E/S:

Plan	Inputs	I/O	Clock/I^2C	A/D	D/A	Interrupciones Externas
0		26	yes	8	3	3
1		29	yes	8		3
2		31	yes	8		1
3	8	29	yes			3
4	8	31	yes			1
5		28	no	8	3	3
6		31	no	8		3
7		33	no	8		1
8	8	31	no			3
9	8	33	no			1

Tabla 1. Opciones de Entrada-Salida del TDS2020F

Los pines se pueden agrupar para diferentes funciones, así que el número de entradas y salidas paralelas libres depende de si se necesita conversión analógo/digital, reloj del tiempo del día no volátil, etc. A partir de esta tabla se puede verificar cuantos puertos de reserva están disponibles. En resumen, el número de bits de entrada o salida paralela está entre 26 y 41 dependiendo de las facilidades usadas.

3.3.9 Bus I^2C. Este es un sistema de dos cables que se usa con periféricos de bajo costo tales como conversores A/D, relojes, E/S, RAMs, EEPROMs, etc

comercializados por Philips, Signetics, Xicor, Microchip Technology y otros. La tarjeta tiene un chip de reloj PCF8583 y RAM conectados a este bus. Otros chips I^2C se pueden añadir externamente.

3.3.10 Entrada salida serial. En la tarjeta hay dos controladores y receptores serial que usan formato y voltajes reales RS232. En el Puerto 1 se soportan todas las tasas de baudios de 75 a 38.4k y hay un MIDI adicional de 31.25k baudios. El Puerto 2 se puede usar a todas las tasas por encima de 4800 baudios. Aunque el sistema se alimenta con una fuente de alimentación sencilla de +6 a +16V, los puertos serie usan ±8V generados internamente en la tarjeta para dar los niveles lógicos de salida que cumplan con las especificaciones RS232.

3.3.11 Microprocesador. El microprocesador es H8/532, un dispositivo de 16 bits con ocho registros de propósito general de 16 bits, hardware de multiplicación y división de 16 bits e instrucciones de 8 y 16 bits. Las manipulaciones directas a bit operan con registros, toda la memoria y entrada-salida. La frecuencia del cristal es de 19.6608MHz y el dispositivo corre en el modo expandido máximo con un espacio de memoria de 1024k bytes.

El H8/532 posee mucho hardware en el chip que incluye tres contadores o temporizadores de 16 bits, un contador/temporizador de 8 bits, puerto serial síncrono y asíncrono y un sistema de interrupciones muy versátil (65 interrupciones y vectores DTC, 8 niveles de prioridad). El Controlador de Transferencia de Datos (DTC) habilita la transferencia entre E/S y memoria sin necesidad de software.

Todas las capacidades del procesador están disponibles al usuario del TDS2020*F*, pero a través del lenguaje Forth de alto nivel, que las hace más fáciles de utilizar. El acceso a algunas facilidades de hardware del microprocesador ya está construido en el sistema Forth pero el usuario es libre de usarlos a su manera por medio de Forth o assembler.

3.3.12 Soporte de teclado. Su sistema de software y hardware habilita más de 64 teclas para ser conectadas al sistema embebido y solo se requieren ocho de las líneas de entrada/salida paralelas. Se conecta un puerto particular de salida paralelo a un lado de una matriz de teclas 8 x 8 por medio de diodos. El otro lado regresa por medio de otros ocho diodos al bus de datos.

3.3.13 Displays de cristal líquido. (LCD) Cualquier LCD alfanumérico basado en el chip HD44780 se conecta directamente al TDS2020*F*; el software para controlarlo está construido dentro de la tarjeta. Un chip externo habilita más de 8 LCDs para ser conectados y el software los sirve a todos ellos simultáneamente.

Los LCDs gráficos también se pueden conectar sin ningún hardware extra; el software controlador para displays basados en HD61830, T6963C y otros controladores gráficos, se da en el CD adjunto con la compra de la tarjeta.

3.3.14 Periféricos externos. Los buses de direcciones y datos están provistos en su totalidad, así como cinco direcciones decodificadas libres para su uso en la selección de periféricos del chip. Se pueden por ejemplo añadir LCDs, E/S extras paralelas, un chip de cuatro puertos seriales o dispositivos similares sin ningún chip de interfase.

Cada decodificador cubre un intervalo de memoria de 16 posiciones. Tres no están sincronizados con la señal de reloj y son capaces de manejar muchos periféricos como "latches" octales (sujetadores de ocho canales) o LCDs gráficos. Los otros dos están sincronizados con la señal de reloj. Uno es activo negativo, el otro es activo positivo. Este último es exactamente apropiado para entradas a displays LCDs alfanuméricos.

3.3.15 Conversor análogo a digital. El TDS2020*F* tiene un conversor A/D de ocho canales y 10 bits que da una parte en 1024 de resolución. El tiempo de conversión es de 21µs usando un programa abastecido en una librería. Un voltaje de entrada de 0V da 0 y uno de +5V da 1023. La referencia en la tarjeta es 5.00±0.05V y hay facilidad para conexión de una referencia externa. Si no se usa el A/D los ocho pines se convierten automáticamente en puertos de entrada paralelos.

3.3.16 Conversor digital a análogo. Hay tres canales de conversión D/A modulada con ancho de pulso (PWM) que actúan como tres bits extra de E/S paralela cuando no se usan. Se debe añadir un filtro externo en la salida (tal como un resistor de 10k y un capacitor de 1µF) para obtener el nivel análogo deseado.

3.3.17 Tiempo y fecha no volátiles. El TDS2020*F* tiene un chip de reloj PCF8583 que mantiene la fecha y el tiempo aún con el computador apagado si se provee con batería externa. La batería también se usa para mantener más de 512k bytes de RAM estática en la tarjeta. El consumo típico es de 3.5µA, dando a la batería un tiempo de vida limitado más por los 5 años de vida de fábrica, que por la capacidad de 950mA-hora propia de la batería.

La fecha y el tiempo son accesibles a una resolución de 813 nanosegundos debido a que otro reloj que corre bajo interrupciones refleja el reloj del chip. Se soportan los formatos de fecha Europeo y Americano.

El chip de reloj genera una interrupción no enmascarable (NMI) de un segundo. Por defecto no hace nada, pero se puede redireccionar para ejecutar una rutina de usuario en Forth de alto nivel o Assembler. El tiempo de repetición se puede cambiar en pasos muy pequeños para dar intervalos de 30ms (milisegundos) a un año.

3.3.18 Temporizador-contadores. Hay cuatro temporizadores de hardware (más el chip de reloj); uno es de 8 bits y los otros tres de 16 bits. Cada uno puede ser cronometrado internamente a cualquiera de tres tasas, o externamente para conteo de eventos. Todos ellos tienen dos registros de captura de salida y los tres temporizadores de 16 bits tienen un modo de captura de entrada. Múltiples interrupciones están asociadas con los temporizadores para procesar eventos asíncronos en Forth de alto nivel o assembler.

3.3.19 Temporizadores de "watchdog". Si el microprocesador se cae por un pico o rizo de potencia o cualquier otra situación, uno de los temporizadores de "watchdog" (perro guardián) restaurará el sistema. El programa de aplicación se iniciará de nuevo. Hay un "watchdog" interno (por defecto 106ms) y un contador externo al microprocesador (426ms). Ambos se restauran por muchas palabras de Forth y normalmente no se notará su acción, pero cronometrarán y reiniciarán el sistema si algo va mal. Al reiniciar, se puede entrar una palabra de Forth definida por el usuario o aceptar el reset por defecto.

3.3.20 Modo de baja potencia. Se puede retirar la alimentación al controlador RS232 y poner el microprocesador en el modo de software "standby". El consumo de corriente baja típicamente a 155µA más algunas cargas externas. Cuando una interrupción no enmascarada ocurra el computador despertará, encendiendo el puerto serie y continuando la ejecución del programa. Esto hace al TDS2020*F* ideal para aplicaciones portátiles y de recolección de datos.

3.3.21 Fuente de alimentación. Hace uso de una fuente sencilla de +6V a 16V y consume una corriente típica de 32mA. El modo operacional de baja potencia consume solo 155µA y no necesita soporte externo de hardware. La tarjeta tiene un generador de ±8V para ser usado por los puertos serie y éste se puede apagar por software para ahorrar potencia. La alimentación negativa también es útil para algunos periféricos externos. El regulador en la tarjeta puede dar más de 100mA, así que hay corriente extra disponible para alimentar cualquier circuitería externa particular a la aplicación.

3.3.22 Interfase de bus. El bus TDS2020*F* es compatible con casi cualquier chip periférico tales como conversores A/D y múltiples dispositivos de puerto

serie. El bus de datos no está multiplexado. En la práctica, se limita la longitud de los buses de direcciones y datos fuera de la tarjeta a 300mm, a menos que se extienda con la circuitería sugerida.

3.3.23 Dimensiones. El tamaño de la tarjeta es 100mm x 80mm con huecos de montaje para tornillos de 2.5mm. La máxima altura, excluyendo los conectores para pines es de 15mm. Si se instala una tarjeta de aplicación sobre los pines (modo sándwich), la distancia entre el sistema embebido y la tarjeta de aplicación es de 11±1mm. Esta debería ser la longitud de los espaciadores si se usan. La temperatura de operación del sistema es de –10 a +70°C. Todos los pines de conexiones coinciden con una matriz de 0.1 pulgadas, de modo que la tarjeta se puede montar fácilmente en una tarjeta prototipo con una matriz de huecos si se requiere. También hay un botón azul de reset en la esquina de la tarjeta.

3.3.24 Sistema Forth ANS. El lenguaje es una completa implementación del Forth Estándar Nacional Americano que incluye punto flotante. Hay muchas extensiones útiles para creadores de sistemas con única tarjeta. Por ejemplo, el número de microsegundos tomados por cualquier palabra Forth se pueden medir con exactitud en tiempo real. El lenguaje ha sido implementado específicamente para el procesador H8/532. Este usa las facilidades del chip donde sea posible; por ejemplo, las multiplicaciones y divisiones de Forth están construidas a partir de las instrucciones de hardware multiplicar/dividir de 16 bits. Sus instrucciones de manipulación de bit también se emplean, ellas son útiles especialmente para detectar y conmutar líneas sencillas de E/S.

Todos los registros Forth están implementados en registros de microprocesador y la pila de parámetros es la misma pila de hardware de las máquinas. Las dos pilas de Forth y las variables de usuario están dentro de la memoria del chip del microprocesador para una velocidad de acceso máxima. Dos registros de máquina se dejan libres (usualmente), de modo que puedan ser usados por interrupciones rápidas sin necesidad de almacenarlos.

El código de aplicación se compila en una Flash-EEPROM no volátil para crear instantáneamente un sistema autónomo. Aparte de Forth, el sistema ROM tiene un assembler simbólico (lenguaje de máquina) que habilita al usuario para escribir código de máquina directamente en la tarjeta. No se requiere tanto trabajo con assembler, ya que una rutina en assembler se puede probar inmediatamente sin necesidad de descargar un archivo binario. La depuración interactiva del código de assembler es muy poderosa.

3.3.25 Extensiones ROM Forth. Entre las extensiones incluidas en la ROM Forth están:

Exploración de teclado.
Displays de cristal líquido alfanuméricos.
Soporte de periféricos de bus I^2C que incluye RAM y reloj.
Interrupciones escritas en código assembler o Forth.
Multitarea.
Reloj del tiempo del día.
Servicio de temporizador "watchdog".
Operación a baja potencia.
Assembler simbólico completo.
E/S serial vectorizada y muchas otras palabras vectorizadas.
Caracteres Hex, ASCII y de control.
Aritmética de número doble (32-bit) y manipulación de pila.
Medida de tiempo de ejecución exacta a un 1µs.

3.3.26 Multitarea. El desarrollo de impresiones, programas de visualización y comunicaciones, como entidades separadas que usan el sistema multitarea, pueden correr todas al mismo tiempo en el TDS2020*F*.

La depuración de cada tarea se puede hacer independientemente de las otras y así se permite la corrida de un subprograma solo tecleando su nombre. La etapa final es combinar los programas individuales y esto es esencialmente un paso mecánico.

3.3.27 Software de desarrollo (firmware). El ambiente de desarrollo *TDS-PC* para Windows, es el software de PC con facilidades para emulación del terminal y edición de múltiples ventanas. El código fuente se almacena en el disco del PC, aunque la compilación toma lugar dentro del TDS2020*F*. Una tecla inicia el proceso automático que envía el programa fuente del puerto serie del PC al TDS2020*F* para su compilación allí. Luego se depura el código interactivamente en el TDS2020*F*. *TDS-PC* para Windows también corre en emuladores de PC Macintosh.

El código fuente puede ser escrito con muchos comentarios y estructuras correctamente separadas del margen (con sangría). Teclas especiales invocan la compilación y otras características. Hay una facilidad de incluir (INCLUDE), de modo que los archivos puedan ser anidados. Hay una interfase al editor usual de texto del código fuente, de modo que la escritura sea más fácil.
Una vez el programa está desarrollado y trabajando, no se necesita nada más para la configuración de desarrollo estándar. No se necesita un programador PROM porque el código se ha compilado en una Flash-EEPROM no volátil.

3.4 UNIDAD DE INTERFASE AL USUARIO

3.4.1 Visualizadores. Una de las primeras cosas que se notan en un computador de flujo, es el visualizador. Tradicionalmente los visualizadores han sido conjuntos de LEDs (diodos emisores de luz) o LCDs (displays de cristal líquido), y los LCDs más avanzados que tienen iluminación de fondo como característica estándar o como una opción. Hoy en día, la selección de visualizadores se ha incrementado significativamente. Los LEDs aún están disponibles pero han sido sobrepasados por los LCDs con múltiples líneas y cadenas de caracteres más grandes, y estos últimos han sido superados por los VFDs (Vacuum Fluorescent Displays) y los LCDs totalmente gráficos.

Los VFDs superan a los LCDs en los siguientes aspectos: Alta luminosidad y brillo, mayor ángulo de visión y amplio rango de temperatura. Además son compatibles con la interfase y con el montaje de un LCD, tienen modo de baja potencia, interfaces de 4 o 8 bits y control del brillo por código.

En general, se deben considerar los LEDs para las aplicaciones más básicas o para situaciones donde la luz del entorno no contribuye a los LCDs. Por ejemplo, la luz directa y brillante del sol es, con frecuencia, un problema para las unidades LCD y VFD. Sin embargo, para la mayoría de aplicaciones los LCD y VFD se prefieren para computadores de flujo, ya que estos visualizadores proveen mayor flexibilidad en el tipo, variedad y volumen de información visualizada.

3.4.2 Teclados. Los fabricantes están incorporando pantallas activas y teclas suaves para programación y configuración dentro de la nueva generación de computadores de flujo. Los primeros computadores de flujo empleaban teclas de panel de control dedicadas para estas tareas. La tendencia hacia computadores de flujo capaces de manejar una gran variedad de aplicaciones, ha llevado a la necesidad de mantener el empaque de las unidades dentro de un tamaño manejable y a hacerlo ergonómico.

Los primeros modelos de computadores de flujo usaban botones pulsadores mecánicos similares a los del teclado de un PC. Aunque estos dispositivos proveían excelente realimentación táctil, eran menos confiables en ambientes industriales hostiles donde los computadores de flujo son instalados. Una de las primeras mejoras fue incorporar un panel plano con tecnología de microinterruptores. Aunque esto ayudó a reducir las fallas mecánicas de los teclados, no proveía realimentación táctil al operador.

La solución a este problema ha sido el desarrollo de botones de panel con microinterruptores que usan varios materiales de memoria mecánica para dar la sensación de botón pulsador al botón de control. Las cubiertas del panel frontal están hechas de materiales poliméricos que proveen buena protección contra salpicaduras, polvo y suciedad. Debajo de la cubierta hay capas adicionales de material polimérico y un componente que provee resistencia al

movimiento del botón por el operador. Con frecuencia, este elemento está conformado por una arandela elástica que puede estar hecha de un material polimérico o de acero inoxidable. Típicamente los microinterruptores están incorporados dentro de una de las capas de polímero. El resultado es un botón que provee buena sensibilidad al operador, aún cuando usa guantes, y a su vez ofrece la superior confiabilidad, durabilidad y economía de los microinterruptores.

Finalmente, la última generación de computadores de flujo permite visualizar pantallas en un idioma extranjero. Esta opción refleja el incremento de la demanda internacional por computadores de flujo.

4. DISEÑO DEL HARDWARE

En este capitulo se encuentran detalles del diseño del hardware existente; este diseño fue elaborado por un proyecto paralelo de la escuela de ing. electrónica y es reproducido aquí para mayor ampliación del hardware que sirve de soporte para la realización de este proyecto.

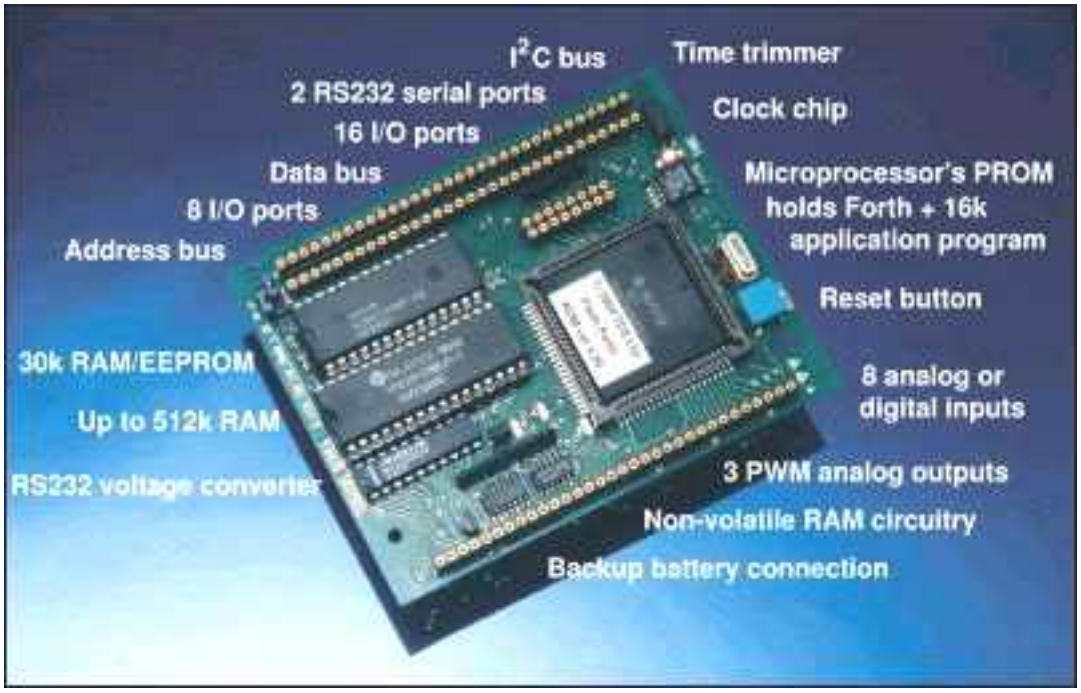
El hardware del computador de flujo está basado en el sistema embebido o CPU (unidad central de proceso), descrita cualitativamente en el capítulo anterior.

En este capítulo se detalla más acerca de las conexiones de esta tarjeta (que de aquí en adelante se llamará CPU) y cómo será acoplada con los periféricos necesarios para el desarrollo del proyecto.

4.1 SISTEMA EMBEBIDO TDS2020F– CPU

La figura 16 muestra la CPU con los nombres de sus principales componentes y su correspondiente ubicación.

Figura 16. Partes de la CPU TDS2020F



Para este proyecto todos los componentes descritos son materia de estudio excepto dos:

- Ø "8 analog or digital inputs": Debido a que no se usará el conversor AD interno de la tarjeta dada su escasa resolución.
- Ø "3 PWM analog outputs": No hay elementos análogos de salida para controlar.

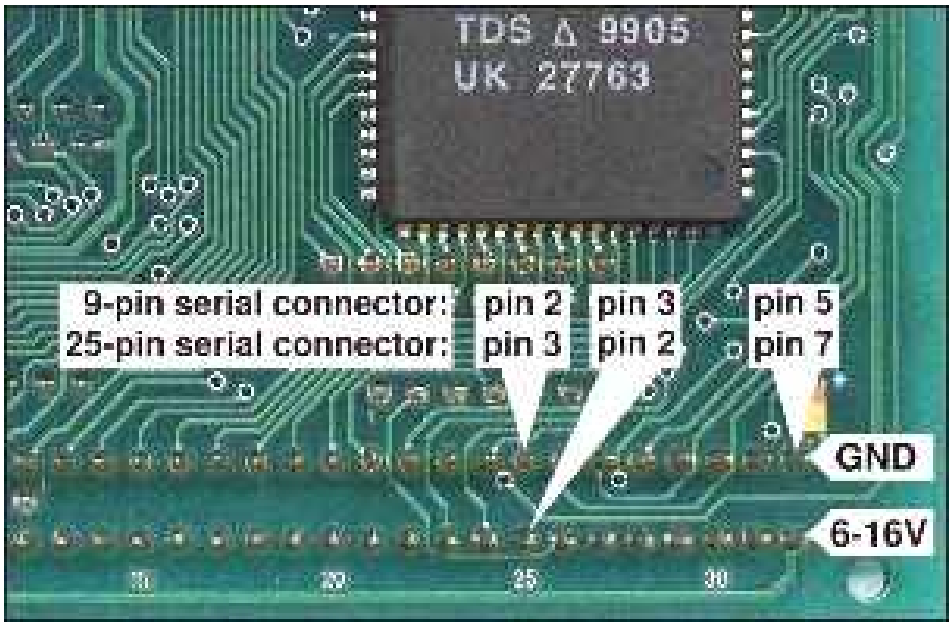
Los pasos a seguir para el encendido de la CPU y el comienzo del proceso de desarrollo del firmware (software embebido) de la CPU son:

1. Colocar el chip llamado "RAM32K" (comprado con el paquete) en el único socket vacío de 28 pines existente. Esta RAM servirá para las pruebas durante el proceso de desarrollo. En las etapas finales se cambiara por el chip EEPROM32k que almacena el programa indefinidamente.

2. Conectar una fuente de DC con un voltaje estable cualquiera dentro del rango de +6 a +16V en los pines a32 (positivo) y c32 (tierra). Ver figura 4.

3. Realizar un cable serial para conectar la tarjeta con un PC y de esta forma establecer la comunicación necesaria para el software de desarrollo del sistema embebido (Forth).

Figura 17. Ubicación de pines para comunicación y alimentación



El cable se realiza siguiendo la correspondencia mostrada en la Tabla 2.

CONEXIÓN EN LOS BORNES DE LA TARJETA	CONECTOR SERIAL DEL PC (COM1 o COM2)
TDS2020*F* Receive (a25)	PC pin 2 (para DB25) pin 3 (para DB9)
TDS2020*F* Transmit (c25)	PC pin 3 (para DB25) pin 2 (para DB9)
TDS2020*F* Ground (c32)	PC pin 7 (para DB25) pin 5 (para DB9)

Tabla 2. Configuración del Cable de Comunicaciones

4. Instalar el software TDS-Windows en el PC, y con el cable conectado encender la fuente de la CPU y arrancar el software.

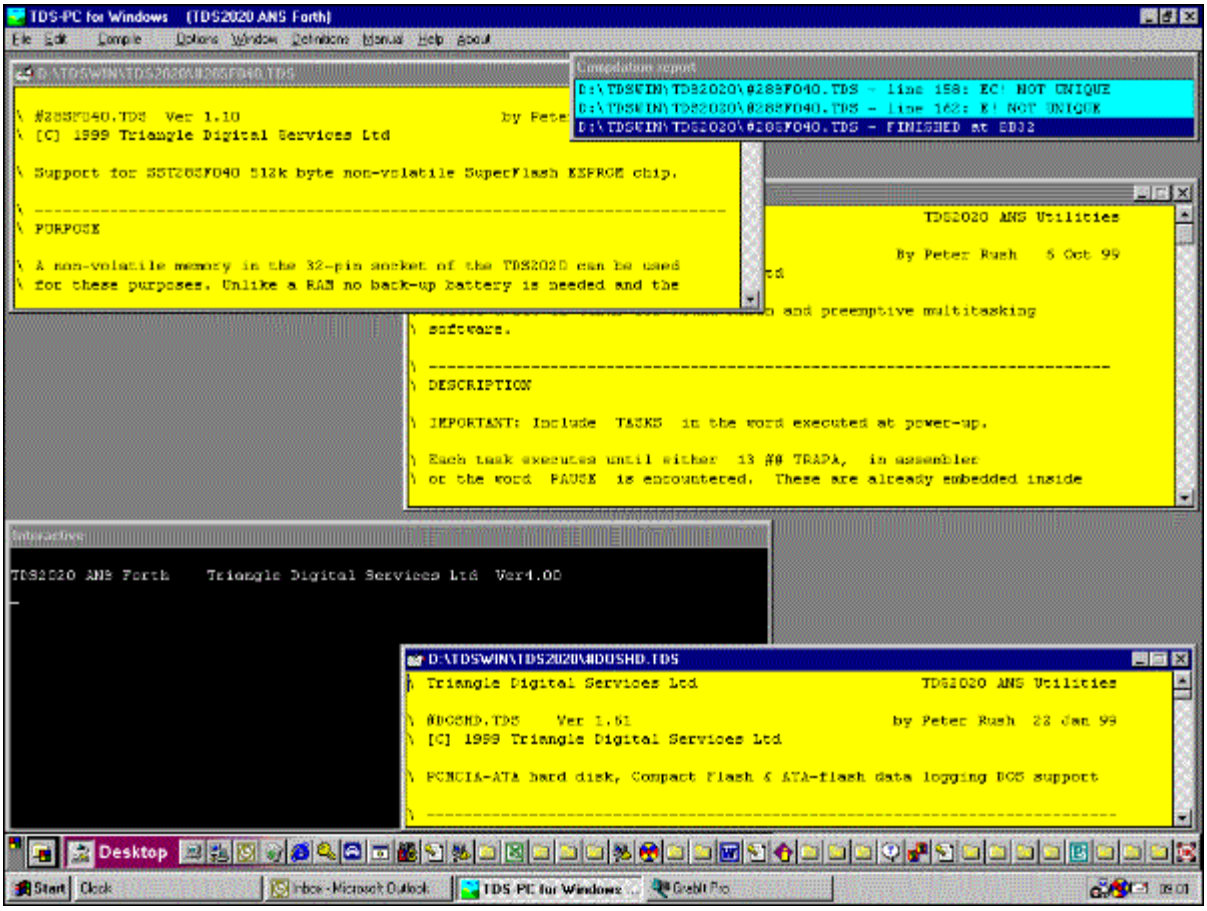
5. Este software automáticamente establece comunicación con la CPU y abre una ventana interactiva, donde se teclean directamente los comandos en Forth para que se ejecuten en la *tarjeta CPU.*

6. Adicionalmente, se abre otra ventana que es un editor de texto donde se redacta el código de software del usuario para luego compilarse y probarse en línea o en vivo con la CPU. (Nada se puede compilar si no hay comunicación con la CPU)

En la figura 17 se observa un entorno típico de trabajo. Las ventanas amarillas son editores de texto donde se visualizan o editan programas en código fuente (Forth). La única ventana negra es el terminal interactivo donde se pueden escribir comandos y son directamente ejecutados en la CPU y en la misma se pueden visualizar los resultados. La pequeña ventana azul claro es donde se despliegan los errores cometidos en código después de un proceso de compilación.

En la figura 18 se detallan todos los pines de la tarjeta CPU y su correspondiente señal; está figura servirá de referencia para futuras ilustraciones de conexión de periféricos.

Figura 18. Entorno de Programación TDS-PC



4.2 CONVERSOR ANÁLOGO-DIGITAL

Este dispositivo es el empleado para convertir las señales análogas de los sensores de presión y temperatura, en señales digitales para ser almacenadas y computadas dentro de la CPU.

Toda esta función la realiza un solo chip y mí nimos componentes adicionales son necesarios para mantener al chip en condiciones estables e inmune al ruido.

Del mercado se escogió al fabricante Texas Instruments y su división la marca Burr-Brown, este fabricante es uno de los lideres mundiales en elaboración de semiconductores.

Dados los exigentes requisitos necesarios para un sistema de medida, se escogió un conversor con los siguientes criterios:

- Ø Alta resolución: 16 bits, para minimizar el porcentaje de error final del computador de flujo.

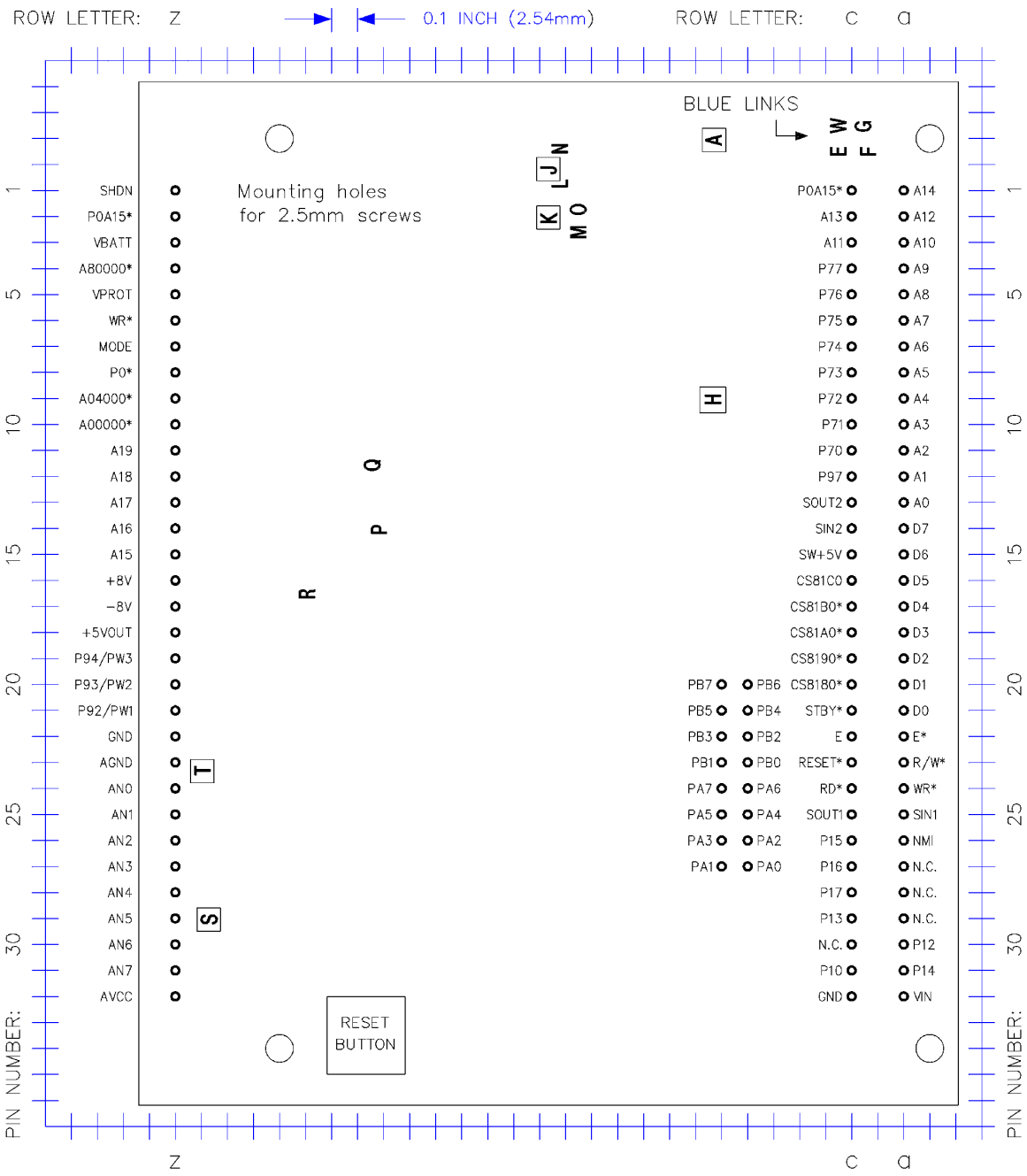
- Ø Interfase: paralela, para mayor velocidad y facilidad de programación, además, no hay escasez de líneas digitales.

- Ø Número de canales: 4 para en un solo chip tener la posibilidad de conectar 2 señales de temperatura y 2 de presión.

- Ø Temperatura de operación: por lo menos de –10 a 70 °C

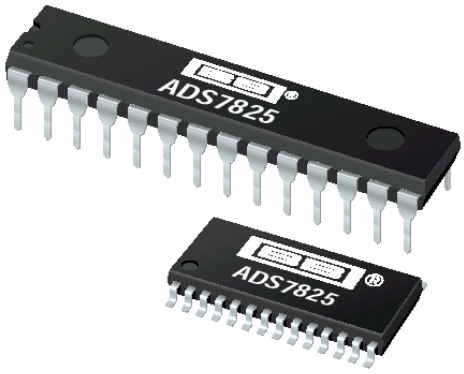
- Ø Buena linealidad y buen tiempo de muestreo y conversión.

Figura 19. Distribución de Pines TDS2020F



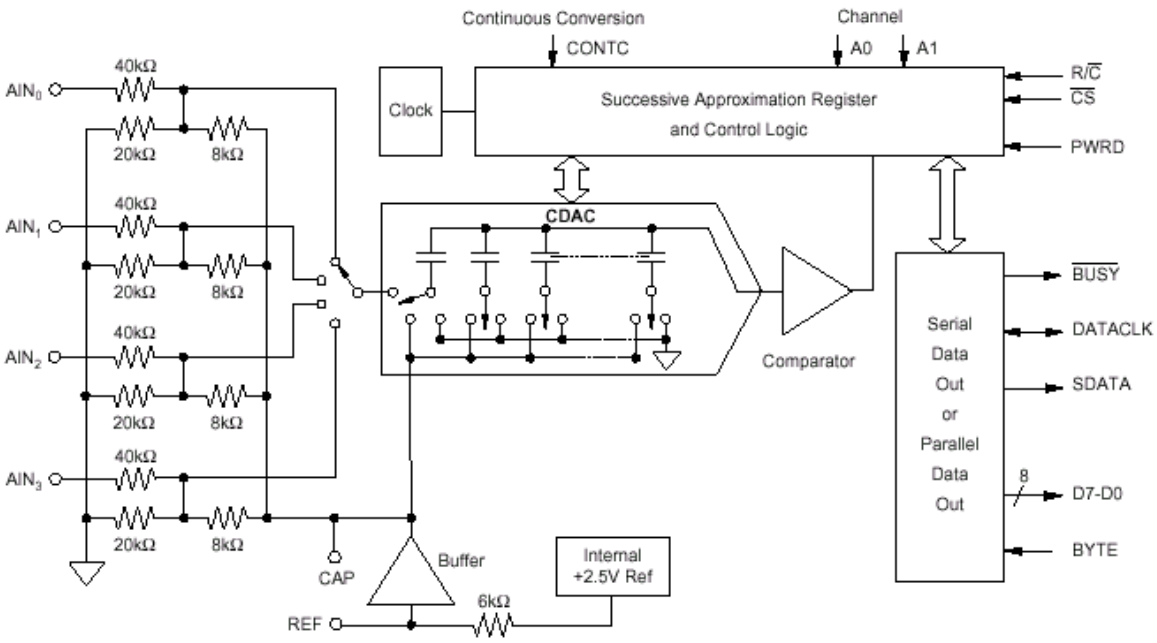
Con estos formulamientos se escogió el circuito integrado referenciado como ADS7825. Una foto de este chip en sus dos presentaciones (DIP y SOIC) se muestra a continuación en la figura 20.

Figura 20. Conversor Análogo-Digital ADS7825



Un diagrama de bloques de su estructura interna se muestra en la figura 21.

Figura 21. Diagrama de Bloques del ADS7825

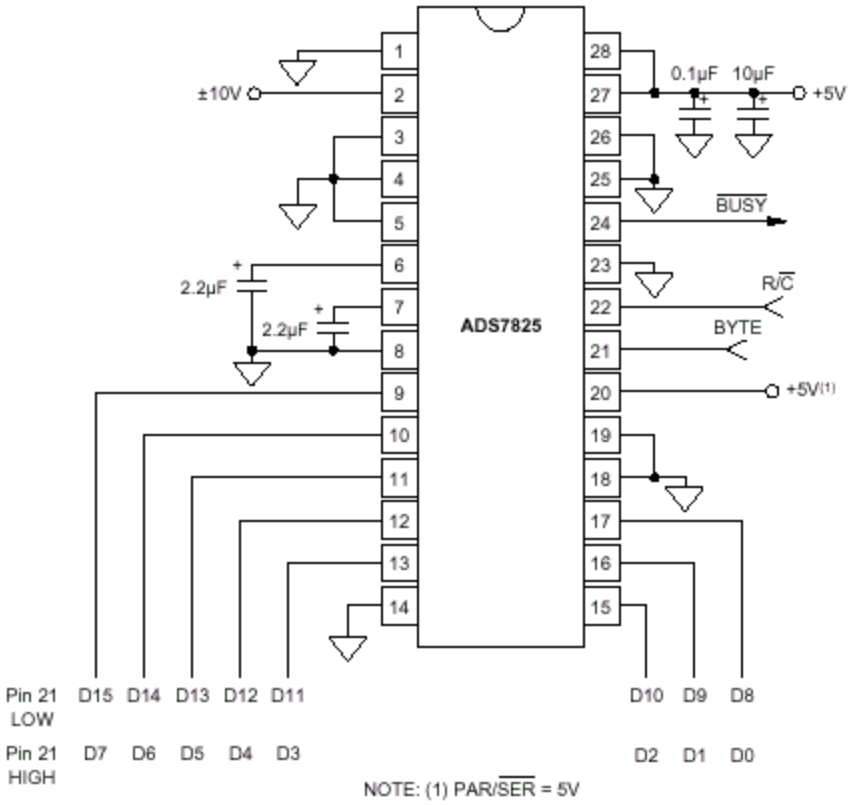


Sus 4 entradas (AIN1, AIN2, AIN3, AIN4), tienen un rango de 10V y por tanto, las señales de los sensores o transmisores deberán adaptarse a este nivel. También se observa que este conversor tiene, tanto interfase serial como paralela, dado que esta última es más rápida y se implementa con mayor facilidad en software, se opta por este camino.

También puede usar una referencia externa o en su defecto su referencia interna de voltaje.

La conexión de este circuito integrado para uso con interfase paralela será de la forma mostrada en la figura 22.

Figura 22. Conexión Típica del ADS7825



4.3 ACONDICIONAMIENTO DE SEÑAL

Las señales de los sensores o transmisores deben ajustarse al nivel de entrada del conversor análogo-digital; esta función la ejecuta un amplificador, el cual para esta aplicación, debe ser de las mejores características por tanto, se emplea un amplificador de instrumentación en vez de un amplificador operacional.

Se escogió de la Marca Burr-Brown de Texas Instruments el circuito integrado referenciado como INA118, el cual incorpora en una sola pastilla 3 amplificadores operacionales, red de resistencias y deja al usuario solo dos

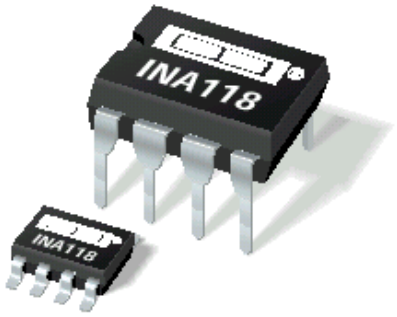
bornes libres para que con solo una resistencia de precisión, ajuste el valor de la ganancia según una fórmula dada que se expondrá más adelante.

El INA118 tiene las siguientes características generales:

- Voltaje de offset bajo: 50µV máx.
- Deriva baja: 0.5µV/°C máx.
- Baja corriente de polarización de entrada: 5nA máx.
- CMR alto: 110 dB mín.
- Entradas protegidas a ± 40 V.
- Amplio rango de alimentación: ± 1.35 a ± 18V.
- Baja corriente en estado inactivo: 350µA
- Encapsulado tanto en DIP como en SO-8.

En la figura 23 observa los dos tipos de encapsulado DIP y SO-8

Figura 23. Encapsulados del INA118



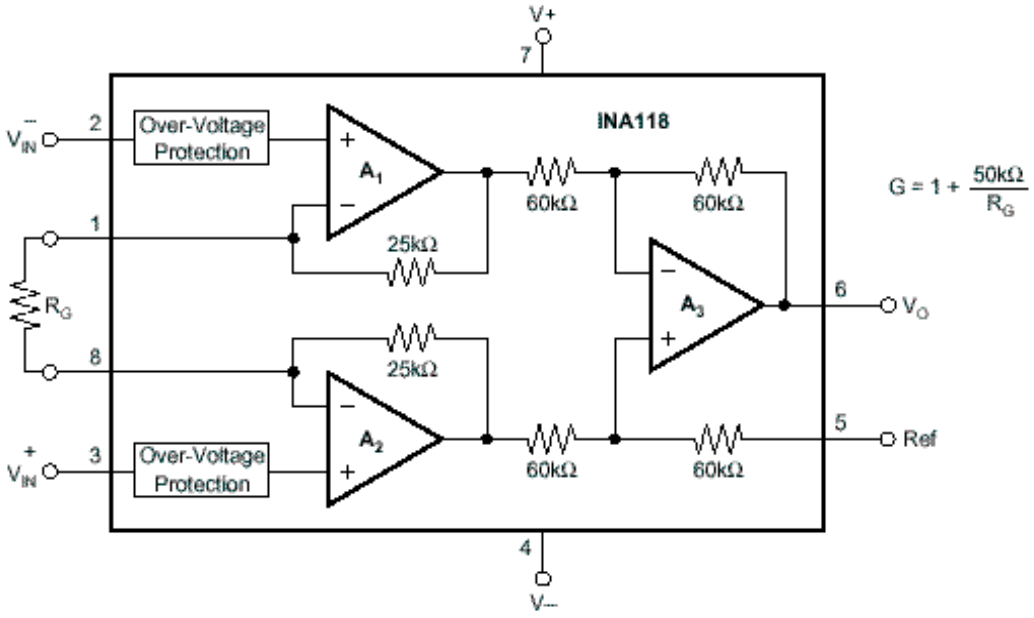
El INA118 es un amplificador de instrumentación de propósito general de baja potencia que ofrece exactitud excelente. Su diseño de 3 amplificadores operacionales y su pequeño tamaño lo hacen ideal para un amplio rango de aplicaciones. La circuitería de entrada de realimentación de corriente provee ancho de banda amplio aún con ganancia alta (70 kHz a G = 100).

Un simple resistor externo ajusta cualquier ganancia de 1 a 10,000. La protección interna en la entrada puede resistir hasta ± 40V sin dañarse. Ver figura 24.

El INA118 es ajustado por láser para ofrecer un voltaje de "offset" muy bajo (50µV), deriva baja (0.5µV/°C) y alto rechazo al modo común (110dB a G = 1000). Opera con fuentes de alimentación tan bajas como ±1.35V, y la corriente de inactividad es solo de 350µA, ideal para sistemas operados con batería.

El INA118 está disponible en DIP plástico de 8 pines, y en encapsulados de montura superficial SO-8, y tiene un rango de temperatura de -40°C a +80°C.

Figura 24. Diagrama de Bloques del INA118



La figura 24 muestra las conexiones básicas requeridas para la operación del INA118. Las aplicaciones con fuentes de alimentación ruidosas o con alta impedancia, pueden requerir capacitores de desacople cercanos a los pines del dispositivo como se muestra.

La salida está referenciada al terminal de salida (llamado "Ref"), que normalmente está aterrizado. Ésta debe ser una conexión de baja impedancia para asegurar buen rechazo al modo común. Una resistencia de 12Ω en serie con el pin "Ref", causará que un dispositivo típico degrade a aproximadamente 80dB CMR (G = 1).

- AJUSTANDO LA GANANCIA.

La ganancia del INA118 se ajusta conectando un sencillo resistor externo, R_G, entre los pines 1 y 8 según la siguiente ecuación:

$$G = 1 + 50k\Omega/R_G$$

Los valores de resistores y de ganancias comúnmente usados se muestran en la Figura 25.

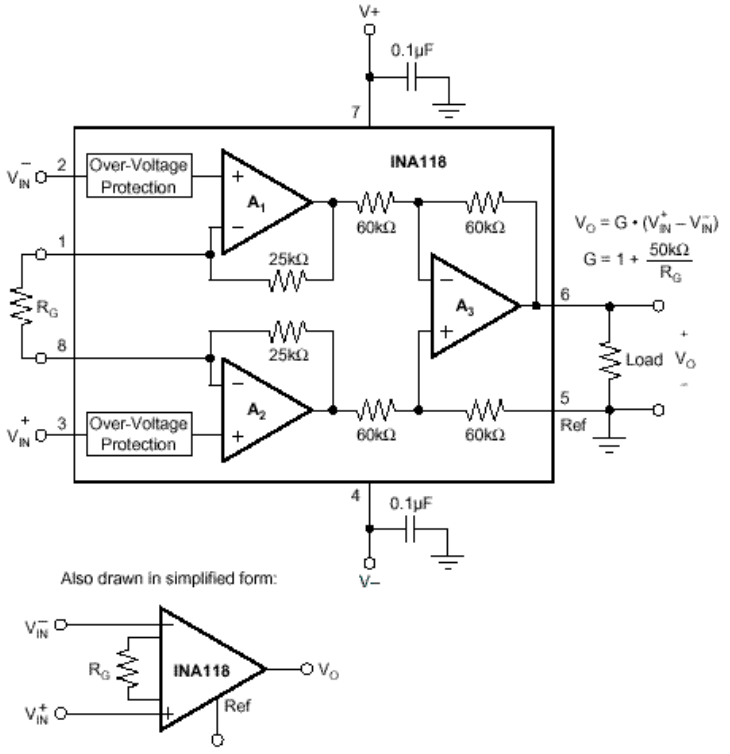
El término 50kΩ en la ecuación viene de la suma de las dos resistencias internas de realimentación de A_1 y A_2. Estas resistencias de película metálica dentro del chip son ajustadas por láser para valores absolutos exactos.

La exactitud y el coeficiente de temperatura de estas resistencias están incluidos en la exactitud de la ganancia y en las especificaciones de derivas del INA118.

Figura 25. Conexión Típica del INA118

DESIRED GAIN	R_G (Ω)	NEAREST 1% R_G (Ω)
1	NC	NC
2	50.00k	49.9k
5	12.50k	12.4k
10	5.556k	5.62k
20	2.632k	2.61k
50	1.02k	1.02k
100	505.1	511
200	251.3	249
500	100.2	100
1000	50.05	49.9
2000	25.01	24.9
5000	10.00	10
10000	5.001	4.99

NC: No Connection.

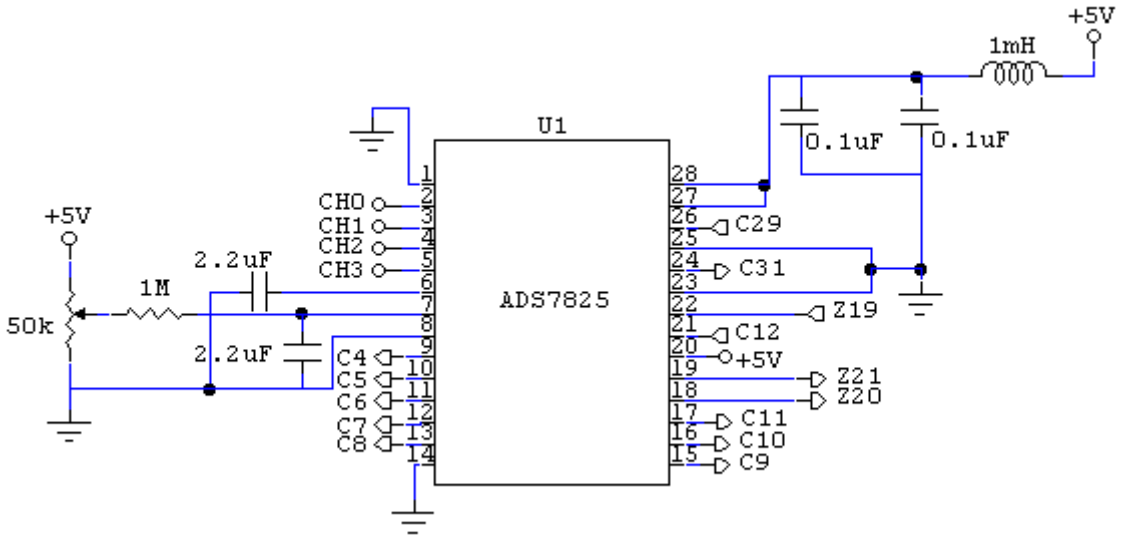


La estabilidad y la deriva de temperatura del resistor externo de ajuste de ganancia, R_G, también afecta la ganancia. La contribución de R_G a la exactitud de la ganancia y a la deriva se puede inferir directamente de la ecuación de ganancia. Los valores bajos de resistencia requeridos para alta ganancia pueden hacer importante a la resistencia de los cables. Adicionar "sockets" al integrado y a la resistencia de ganancia contribuye, con un error de ganancia adicional (posiblemente un error de ganancia inestable), en ganancias de aproximadamente 100 o más grandes.

4.4 CONEXIÓN DEL ADS7825 AL TDS2020F

A continuación en la figura 26 se observa el conexionado necesario para el conversor A/D ADS7825. Las señales a convertir están dentro del rango de 0 a 10V y pertenecen a los sensores de temperatura y presión.

Figura 26. Conexiones del ADS7825 a la CPU



De la figura 26 se detalla lo siguiente:

- Ø Pin 14; DGND: Tierra del circuito.

- Ø Pin 2 y 4; Vo.RTDa: Señales provenientes de la etapa de acondicionamiento de la RTD.

- Ø Pin 3 y 5; Vo.SP: Señales provenientes de los sensores de presión.

- Ø Pin 9 al 16; D7...D1: Salida digital convertida; es una salida de 8 bits pero el convertidor es de 16 bits; cuando BYTE (pin 21) es alto, la salida corresponderá a los bits menos significativos y cuando sea bajo, corresponderá a los 8 bits más significativos.

- Ø Pin 19 y 18; A0, A1: Señales provenientes de la tarjeta CPU que permiten seleccionar cual canal se va a convertir; cuando ambas están en bajo el canal a convertir es el cero, o sea la señal Vo. RTD a y cuando A0 es alto y A1 es bajo, la señal a convertir es la del canal 1, es decir Vo.SP.

- Ø Pin 21; BYTE: Si esta señal de entrada es baja, los ocho bits más significativos serán válidos cuando BUSY suba. Si es alto, los ocho bits menos significativos serán válidos cuando BUSY suba. Esta señal es una salida de la tarjeta controladora.

- Ø Pin 22; R/C*: Esta señal permite iniciar la conversión manteniéndola 40ns en bajo, y es una salida de la tarjeta CPU y su temporización se manejará por software.

- Pin 24; BUSY*: Esta señal de salida irá a bajo y permanecerá así hasta que la conversión se complete y el registro de salida sea actualizado. Esta señal es una entrada de la tarjeta controladora, que le permitirá saber al programa cuando puede leer datos válidos a la salida del convertidor.

Nota: Los asteriscos al final de una señal indican que ésta es activa en bajo.

4.5 CONEXIÓN DE SENSORES DE TEMPERATURA

En la figura 27 se observa el circuito de acondicionamiento de señal diseñado para la salida del sensor de temperatura RTD PT100. El circuito emplea el integrado REF200 como referencia de corriente; su configuración permite obtener 200µA. Esta corriente se utiliza para excitar la RTD. Debido a que ésta es un sensor pasivo, es decir, no genera tensión sino un cambio de resistencia, se requiere un medio para excitarla y medir en ella la tensión que se produce al paso de la corriente; de este modo, al tener una corriente constante la medición será directamente proporcional a la variación de resistencia en la RTD. Posterior a la etapa de excitación está la etapa de amplificación, la cual se lleva a cabo mediante el amplificador de precisión INA 118, el cual está trabajando con ganancia de 1846.99 para obtener a la salida un rango de 0 a 10V para una temperatura de entrada oscilando entre 0 y 70°C. La ecuación de la tensión de salida Vo.RTDa es:

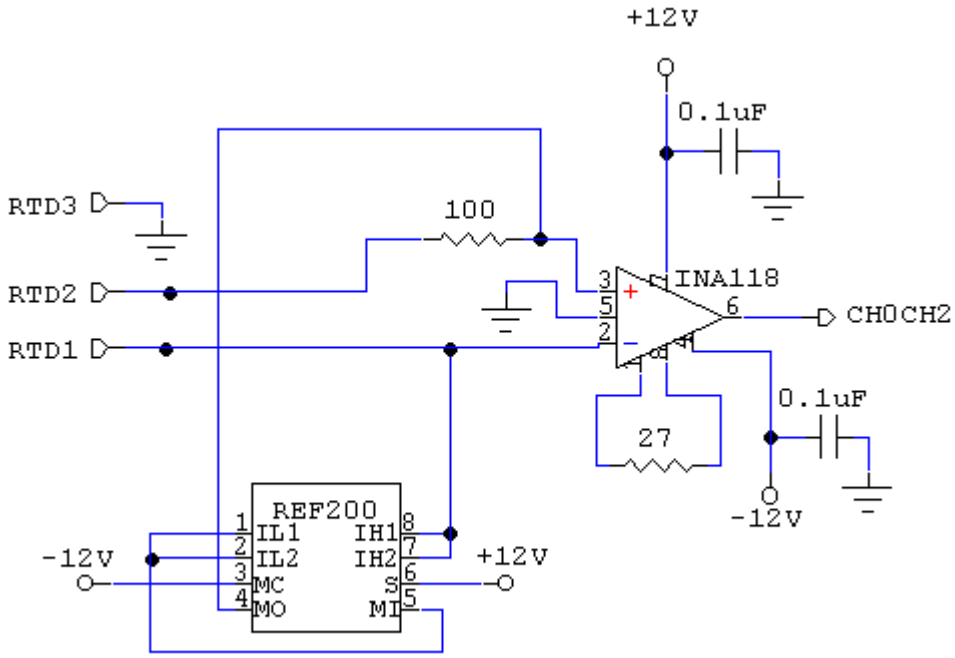
$$Vo.RTDa\ (T) = G * 200E\text{-}6 * (R_{RTD}(T) - 100)\ V,$$

Donde G= 1846.99
$R_{RTD}(T)$ es la resistencia de la RTD a la temperatura T.
Vo.RTDa(T) es la tensión de salida del circuito acondicionador a la misma temperatura T.

En la figura 27 se observa el circuito acondicionador de señal para un RTD-PT100, y del cual de detalla lo siguiente:

- DGND: Tierra del Circuito
- DVSP: Vs^+ (Alimentación Positiva)
- DVSN: Vs^- (Alimentación Negativa)
- Vo.RTDa: Salida Amplificada de 0 a 10 V para un Rango de Temperatura de 0 a 70°C.
- INA118: Amplificador de Instrumentación
- REF200: Referencia de Corriente
- RTD (1,2 Y 3): Salidas del Sensor RTD PT100

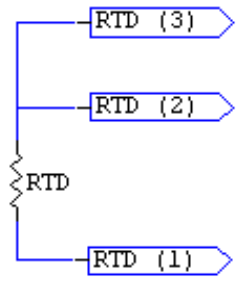
Figura 27. Acondicionador de Señal para una RTD PT100.



Nota: Los valores de los componentes del circuito de la figura 27 son resultado de los cálculos teóricos; posteriormente se ajustarán a sus correspondientes valores comerciales más cercanos.

A continuación se observa la figura 28 que esquematiza las salidas del sensor RTD PT100 para su adecuada conexión al circuito de acondicionamiento de la figura 27.

Figura 28. Bornes Tí picos de una RTD



Se seleccionó la siguiente RTD de la empresa Murphy.

Figura 29. RTD PT-100

- Marca: Murphy
- Construcción de Acero inoxidable
- Salida 100ohm, PT-100, 3 HILOS
- Conector Metalico nema 4X, NEC Class 2
- Exactitud: 0,12%
- Rango de Operación: hasta 470 °F
- Termopozo de Acero 304 con rango de operación hasta 7000psi

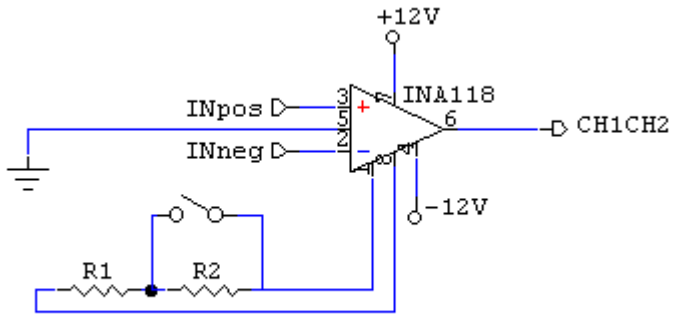
4.6 CONEXIÓN DE SENSORES PRESION

Se seleccionaron transductores de presión de 0-100mV, los cuales tiene 4 hilos: 2 para alimentación (rojo + y negro -) y 2 para la señal de medición (verde + y blanco -). El consumo de un transductor es de aproximadamente 2mA.

También se tuvieron en cuenta durante la selección a los transductores con salida amplificada directa de 0-5V o 4-20mA pero fueron descartados por su consumo de corriente (15mA por cada transductor, y en el caso de turbina se necesitan dos).

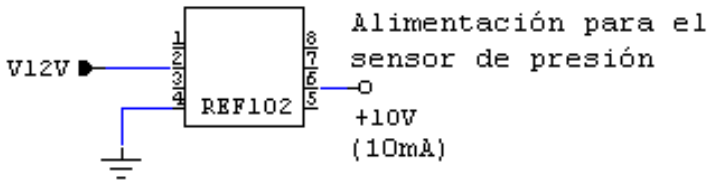
Por tanto, la salida de los transductores seleccionados debe ser amplificada para que alcance los niveles de entrada del conversor análogo digital. (0-10V)
Se usó el circuito de la figura 30 que usa el INA118 explicado anteriormente.
Se usan las dos resistencias R1 y R2 para cuadrar un valor de ganancia de 100.

Figura 30. Amplificación de los Transductores de Presión.



El siguiente circuito muestra la alimentación usada para estos sensores extraída de una referencia de voltaje (REF102) de 10V, este chip asegurará una alimentación precisa para estos transductores.

Figura 31. Alimentación para los sensores de presión.



Se seleccionaron los siguientes sensores:

Ø Sensor de Presión:

Figura 32. Sensor de Presión Barksdale

- § Marca: Barksdale
- § Construcción de Acero inoxidable
- § Salida No-Amplificada
- § Estable sensor de silicio
- § Exactitud del ±0,25%
- § Voltaje de entrada: 10 voltios
- § Salida Full Escala: 100mV
- § Peso: 130 gramos
- § Rango de Presión: 0-300 y 0-1500 psig
- § Rango de Operación: -18 a 71 °C

Ø Sensor de Presión Diferencial:

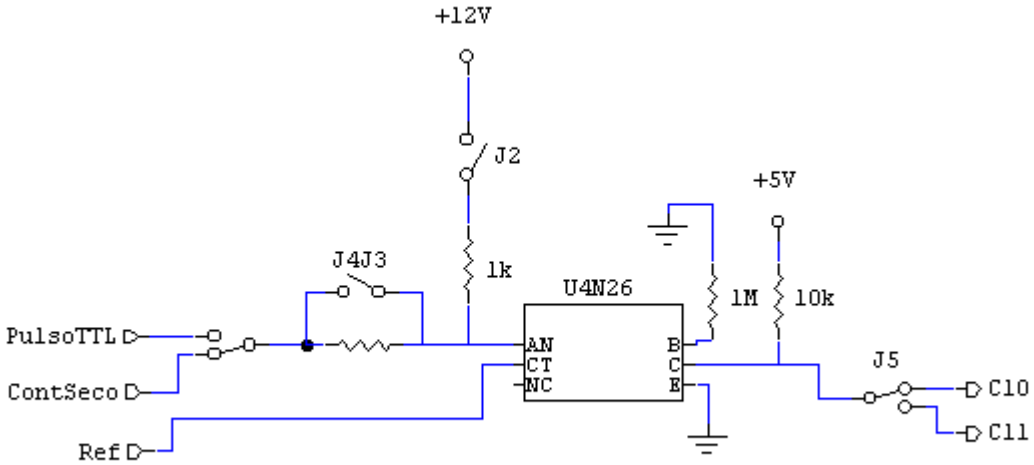
Figura 33. Sensor de Presión Diferencial

- § Marca: BAILEY
- § Construcción de Acero inoxidable
- § Salida 4-20mA
- § Exactitud del ±0,25%
- § Rango de Presión diferencial: 0-360" H20
- § Rango de Operación: -18 a 71 °C

4.7 CONEXIÓN DE ENTRADA DE PULSOS.

Para la medición con turbina se requiere una entrada de pulsos por cada brazo. Esta entrada debe ser adaptada para leer pulsos digitales o el estado de un contacto seco debido a que unas turbinas usan alguno de estos dos sistemas. Por tanto, se dejó seleccionable por ¨jumper¨ esta configuración.

Figura 34. Circuito para entrada de pulsos



5. GENERALIDADES DE FORTH

Todo desarrollo de programas tiene sus dificultades, pero la tarea frente al programador de un sistema embebido es particularmente exigente.
El espacio de memoria disponible será estrictamente limitado, y el desempeño o el rendimiento del procesador de control siempre debe satisfacer las demandas del sistema de control.

Los mismos requerimientos de código rápido y compacto, han obligado a que muchos sistemas embebidos sean programados en assembler, el cual incluso para el más experimentado programador, sigue siendo una tediosa labor propensa a errores. En muchas aplicaciones, assembler ha dado camino a lenguajes de alto nivel (high-level languages, HLLs), pero para sistemas embebidos la opción HLL es a menudo poco práctica. Muchos de los procesadores usados son simples recursos de bajo costo, con una limitada o inexistente opción de HLLs. Y aún, si buenos compiladores de alto nivel estuvieran disponibles, su uso implica una inevitable e inaceptable falta de eficiencia en el código de ejecución.

En los años 60's el estadounidense Charles Moore estaba escribiendo un software para controlar telescopios en la Kitt Peak National Observatory en Arizona. Insatisfecho con los lenguajes convencionales se propuso diseñar un nuevo lenguaje que haría su trabajo más fácil y productivo. El resultado fue un lenguaje altamente distintivo, el cual se llamó Forth.

5.1 ARQUITECTURA

Implícito en la estructura de cada lenguaje de programación, está la arquitectura de la máquina ejecutando el programa. Esto puede corresponder a un procesador real en el caso de assembler, o a procesadores virtuales altamente abstractos, en el caso de HLLs.

Para muchos HLLs la operación clave de su procesador virtual es la asignación de nuevos valores a variables. De este modo, en pascal para el incremento de la variable A en uno; simplemente requiere una asignación como esta:
A : = A + 1 ;

No hay necesidad que el programa deduzca explícitamente a cerca del lugar donde se almacena el valor de A, o cómo el computador usa sus registros internos y la unidad aritmética lógica, para obtener el resultado deseado. De todos estos detalles engorrosos se encarga un interpretador o compilador el

cual traduce las afirmaciones del HLL a un código expresado en el conjunto de instrucciones del computador. Los HLLs hacen programadores muy productivos y producen código que puede ser corrido sobre una amplia variedad de procesadores.

Sin embargo, nada es gratis; la traducción del HLL al "set" de instrucciones nunca es 100% eficiente. Así que el precio de usar un HLL es que toma mucho más tiempo en ejecutar programas más largos. Esta es la básica ineficiencia por la cual los HLLs se hacen inapropiados para sistemas embebidos, forzando a los programadores a usar assembler. Forth está entre los dos extremos, HLLS y assembler; proporcionando al programador un lenguaje altamente expresivo y un modelo implícito de ejecución de programa, el cual puede estar muy cerca de las operaciones de procesadores reales.

5.2 LA PILA

Una pila es simplemente un conjunto de números donde el último puesto es el primero en ser sacado. Muchos procesos modernos incorporan hardware de pila y las pilas son usadas por casi todos los lenguajes de programación.

De cualquier modo, mientras que la mayoría de lenguajes ocultan la operación de la pila para el programador, en Forth el programador controla la pila directamente.

Algunos ejemplos darán claridad.

El siguiente es un ejemplo en Forth: 3 4 +

Se oprime "Enter" y el computador retornará 'OK' . Forth ha confirmado que esto está bien. Pero que ha pasado? Cuando se oprimió enter , el interpretador de Forth reconoce 3 como un número y lo pone en la pila, seguida y de igual forma 4 fue puesto en la pila. El "+" es un ejemplo de una palabra en Forth.

Forth es en últimas un lenguaje con dos elementos básicos: números y palabras.

Para determinar el significado de "+" , el interpretador de Forth consulta el diccionario Forth que contiene detalles de todas las palabras definidas, junto con el nombre de la palabra, cada entrada en el diccionario incluye instrucciones definiendo como las operaciones se asocian con la palabra. Como se puede esperar "+" suma el primero y el segundo número y pone el resultado en la cima de la pila.

Para confirmar esto, se puede usar la palabra "." Que saca el valor de la cima de la pila y despliega este valor oprimiendo después "Enter". Forth responderá

con ' 7 ok' ; 3+4 es ciertamente 7 . Pero porqué en el ejemplo, el + aparece después de los números a ser sumados? Forth usa "reverse Polish notation" (RPN), en que los operandos preceden al operador. Esto puede parecer incómodo, pero tiene la ventaja de ser lo más cercano posible a la verdadera forma en que opera el computador.

Antes de que un computador pueda desarrollar cualquier operación, este debe tener los operadores (variables) a la mano; con RPN, operadores y operandos son automáticamente presentados al computador en la secuencia correcta.

Los programas en Forth son diseñados por un proceso de definición de nuevas palabras requeridas para implementar el objetivo del programa. Cada nueva palabra puede tener un corto claro y conveniente nombre descriptivo; haciendo que el programa resultante sea lo más entendible posible.

Es importante apreciar que todas las palabras en Forth, sean predefinidas o definidas por el programador; tienen exactamente el mismo estado; son simples entradas en el diccionario de Forth. Programar en Forth, es de este modo equivalente a extender el lenguaje predefinido, para crear un rico lenguaje conveniente según las necesidades. Esta naturaleza extensible de Forth, es su más distintiva caracterí stica.

5.3 EJEMPLOS

Forth viene con un kit inicial de algunos cientos de palabras predefinidas. La mayorí a de ellas cambian la pila de alguna manera y los programadores de Forth tienen adaptada una conveniente notación para describir como una palabra en Forth afecta la pila.

Por ejemplo, la instrucción 2DUP, duplica el primer y segundo item de la pila. Esta operación puede ser definida así: 2DUP (n1 n2 - - n1 n2 n1 n2), donde (n1 n2) es el estado inicial de la pila antes de la ejecución de 2DUP, con n2 en la cima de la pila; y (n1 n2 n1 n2) es el estado de la pila después de la ejecución de 2 DUP.

Otros ejemplos de palabras de Forth predefinidas que suelen ser usadas son:
DUP (n1 - - n1 n2) : Duplica la cima de la pila.
DROP (n1 - -) : Elimina la cima de la pila.
SWAP (n1 n2 - - n2 n1) : intercambia las dos ultimas cifras de la cima de la pila.
/ (n1 n2 -- cociente) : divide n1 entre n2.

Las palabras predefinidas en Forth pueden ser muy útiles, *pero el verdadero poder de Forth está en las nuevas palabras que se pueden crear*. Suponga que se quiere una palabra que borre el segundo item de la pila. Este es un requerimiento común. Desde el final de una serie de pila, manipular a menudo

los 2 últimos ítems de la pila, pero solo nos interesa uno. Muchas implementaciones de Forth ya incluyen esta palabra, llamada NIP, pero si NIP no está predefinido en alguna versión de Forth; este es un ejemplo para crearlo:

```
:NIP \ n1 n2 - - n2
SWAP DROP;
```

Tomando este programa palabra por palabra se tiene:

- Ø : es una palabra predefinida de Forth indicando el comienzo de una definición de una nueva palabra.
- Ø 'NIP' es el nombre de la nueva palabra.
- Ø 'SWAP' y 'DROP' son palabras predefinidas en Forth.
- Ø ; es una palabra predefinida que indica el final de una nueva definición de palabra.

Todo programa en Forth que defina nuevas palabras sigue este básico patrón. El programa empieza con ':' ,seguido por el nombre de la nueva palabra, continuando con la definición de la nueva palabra, escrita en palabras Forth previamente definidas; y termina con ';' .

Programar en Forth consiste esencialmente en la creación de nuevas palabras requeridas para su archivo.

Como ilustración del estilo de programación Forth, considere un programa para calcular las raíces cuadradas por el método de Newton. Este método es basado en el principio que si *m* es un estimado de la raíz cuadrada de *N*, entonces *½(m + N/m)* es una mejor estimación. Tomando *N/2* como el valor inicial de m y sucesivamente aplicando *(m+N/m)/2* se pueden generar raíces cuadradas de gran exactitud.

De esta forma, se puede crear este programa alrededor de dos nuevas palabras:

Primera_Estimación la cual calcula N/2, y *Mejor_Estimación* que calcula (m + N/m)/2.

La primera estimación definida por:

```
:primera_estimación
\ N -- N N/2
DUP 2 /;
```

Y mejor estimación definida por:

```
:mejor_estimación
\ N m - - N (m+ N/m)/2
2 DUP / + 2 /;
```

Las operaciones detalladas son mostradas en la tabla 3:

PALABRA EN FORTH	EFECTO EN LA PILA
DUP	N - - N N
2	N N - - N N 2
/	N N 2 - - N N / 2
2DUP	N m - - N m N m
/	N m N m - - N M N/m
+	N M N/m - - N (M + N/m)
2	N (m + N/m) - - N (m + N/m) 2
/	N (m + N/m) 2 - - N (m + N/m)/2

Tabla 3. Secuencia de Operaciones de un programa en Forth

La nueva palabra mejor_estimación, tiene que ser aplicada sucesivamente; y esto se puede lograr mediante un DO LOOP, de la forma:

```
limit index DO  Forth words
LOOP
```

'limit', 'index' , son dos números que controlan el número de LOOPs realizados, y el total de ejecuciones del DO LOOP, es simplemente la repetición de las palabras entre DO y LOOP (limit - index) número de veces. Si optamos por desarrollar 5 interacciones sucesivas de mejor_estimación, entonces el programa raíz cuadrada, puede ser escrito así :

```
: Raíz cuadrada  \ N - - raíz cuadrada
primera_estimación  5  0
mejor_estimación  LOOP  NIP;
```

El NIP final simplemente pone en orden la pila, eliminando el segundo elemento no deseado. El ejemplo acepta solo números enteros mayores que 1, retornando una respuesta entera, y además el número de interacciones es fijo. Pero lo que se pretende ilustrar es la esencia de Forth .

5.4 FORTH Y VELOCIDAD

La rápida ejecución es un importante atributo de los lenguajes de programación. Pero será un beneficio poco práctico si no es acompañado por un aceptable corto tiempo de desarrollo. Desafortunadamente, hay un conflicto entre estos dos requerimientos.

Un HLL (lenguaje de alto nivel) puede ser traducido a un "set" de instrucciones de procesador, para ser compilado o para ser interpretado. Para un programa compilado la traducción es hecha una vez por todo el programa (llamado programa fuente) , y este es el programa objeto resultante (OBJ), que es conservado para ejecución. Para un programa interpretado, la traducción se hace declaración por declaración cada vez que el programa es ejecutado.

La ejecución de un programa interpretado es muy lento; pero la interpretación tiene la importante ventaja que no tiene fallas sujetas a cambios hechos en el programa HLL.

Con un programa compilado sin embargo, generalmente es necesario recompilar todo el programa, cada vez que se realiza un cambio. Como una regla, la interpretación se orienta a rápido desarrollo, pero lenta ejecución; y la compilación se orienta a una rápida ejecución y lento desarrollo.

Forth es un lenguaje de interpretación y compilación; ofrece lo mejor de los dos mundos.

Si entramos la siguiente línea en Forth:

```
3  4  +
```

Se obtiene '7 ok' como respuesta; entonces estamos usando Forth como un lenguaje puramente interpretador. Sin embargo, una definición de una nueva palabra; por ejemplo:

```
: cuadrado  \ n  - - n*n
  DUP * ;
```

Llama al compilador de Forth, el cual crea una nueva entrada en el diccionario. Esta entrada incluye el nombre de la nueva palabra: cuadrado, junto con el "set" de punteros que se refieren a las otras palabras en la definición; ejemplo : DUP y * . Estas palabras pueden a su vez referirse a otras palabras, hasta que en ultimas, la secuencia de punteros finaliza en palabras Forth primitivas definidas en código de máquina.

Como todas las nuevas palabras son compiladas individualmente; Forth tiene la facilidad de desarrollo de un lenguaje interpretador. Sin embargo Forth también puede competir con la velocidad de ejecución de un lenguaje compilador, una característica resultante de la definición de palabras primitivas en código de maquina, de esta forma, ellas no tienen que ser traducidas cada vez que son llamadas.

Cada programa Forth es en últimas, una palabra Forth, la cual se ejecuta siguiendo los pasos a través de todas las palabras componentes. Esto es equivalente a llamar una sucesión de subrutinas en un convencional HLL. El uso liberal de subrutinas, está sujeto a una buena práctica de programación, pero en un convencional HLL, cada vez que una nueva subrutina es llamada,

un área especial de memoria (stack frame), tiene que ser establecida, para retener la dirección de retorno, así como los valores actuales de las variables requeridas por los llamados a las subrutinas. Cualquier subrutina anidada dentro de la subrutina llamada, complica todo de ahí en adelante.

En general, crear esta área de pila (stack frame), es muy costoso en tiempo de procesador. El factor clave en la velocidad de ejecución de Forth, es el procesador orientado a pila; el cual proporciona localizaciones implícitas de memoria para toda la información requerida en un llamado de subrutina; salvando de este modo el esfuerzo requerido para crear a propósito estas áreas de memoria.

La pila descrita en el ejemplo dado anteriormente, es también llamada "date stack" (pila de datos), la cual provee una localización implícita para las variables locales; una segunda pila, el retorno de pila, contiene subrutinas o direcciones de palabras. El procesador virtual de Forth, es así una máquina de dos pilas, una pila de dato y una pila de retorno; ambas juegan un papel crucial en la ejecución de Forth.

5.5 PROCESADORES FORTH

Dado que el procesador virtual de Forth es tan simple, este es un claro incentivo para alcanzar lo último en la ejecución de Forth: implementar una especie de máquina directamente en hardware; por ejemplo, construir un procesador con dos pilas de hardware dedicadas y un lenguaje ensamblado de palabras de Forth.
Esto se puede hacer de dos formas: con un solo chip, o con el diseño de una tarjeta base.

El RTX2000, producido por Harris Semiconductor, es un ejemplo de la primera propuesta. Este es un microcontrolador de 16 bits y 10 MHZ con un multiplicador de hardware de 16*16 y dos pilas, la de datos (data stack) y la de retorno (return stack), ambas de 256 palabras.

El RTX2000 es diseñado con arquitectura RISC (Reduced Instruction Set Computer); como tal, este no tiene microcódigo, y la mayoría a de instrucciones del procesador se ejecutan en un solo ciclo. El "set" de instrucciones incluye muchas palabras comúnmente definidas de Forth, tales como DUP, SWAP y DROP. La arquitectura interna del RTX2000 está organizada alrededor de 4 buses de información separadas, cada uno de los cuales puede ser usado simultáneamente con el bus de datos de la memoria principal. Esta arquitectura altamente paralela permite a cuatro palabras Forth ser ejecutadas simultáneamente, produciendo una muy alta tasa de velocidad de procesamiento.

El MF1600 producido por Hull-based Advanced Microprocessor Design, es una máquina Forth basada en dos tarjetas de alta velocidad y funciones de manejo de bits para implementar un procesador de 20 Mghz, el cual ejecuta operaciones enteras y de punto fijo por encima de 6.67 MIPS. La pilas de datos y retorno en el MF1600 son ambas de 1024 palabras. El MF1600 es un procesador de microcódigo con un "set" de instrucciones implementado con unas 180 palabras Forth . Estas incluyen todas las operaciones básicas de Forth , así como un número de funciones más poderosas, cubriendo operaciones como comparación de cadenas y soporte de estructuras y ciclos de control; además, hay espacio dentro del microcódigo ROM, para instrucciones de aplicaciones específicas.

5.6 FORTH EN EL MERCADO

Hace 20 años, Charles Moore fue el único programador Forth en el mundo; hoy Forth está disponible en cada procesador comercial y está disfrutando de un "boom" de popularidad. En parte esto se atribuye, a la altamente distintiva naturaleza de Forth. Este ofrece desarrollo interactivo, rápido, veloz ejecución, combinado con acceso directo al hardware; tres sólidas virtudes de los caprichos de la moda, que pueden asegurar que Forth permanezca como un lenguaje importante en la demanda mundial de control en tiempo real.

6. TRANSFERENCIA DE CUSTODIA

Muchos computadores de flujo modernos y digitales, los cuales se consideran para transferencia de custodia, son 99.9% exactos o mejores, y no están sujetos a derivas (cambios de parámetros internos) con el tiempo; es decir, no se descalibran con el paso del tiempo, o es despreciable su cambio.

El único problema o requisito es abastecerlos con la información adecuada. Las entradas típicas incluyen presión diferencial, presión, temperatura, gravedad específica, densidad, poder calorífico, y/o frecuencia si se trata de medidores de turbina.

La selección, instalación, mantenimiento y pruebas de estos dispositivos, es crítica en una aplicación de transferencia de custodia. Solo se deberían considerar los dispositivos de la más alta calidad con excelentes especificaciones de exactitud y estabilidad.

Todos los instrumentos de la tubería, tubos colectores (manifolds), etc. deben ser cuidadosamente considerados y se deberán tratar como cualquier otro elemento de transferencia de custodia.

El costo de los transductores para medir las variables físicas del gas fluyendo es más alto, pero la exactitud potencial del sistema es mayor. Esto puede justificar el costo adicional si los volúmenes a ser medidos en una estación dada, son lo suficientemente grandes.

En estaciones de medida muy grandes, el gas natural se vende, con frecuencia, por unidades de energía. En muchos casos, la visualización del volumen no se requiere y solo las unidades de energía de ventas finalizadas, se visualizan en el medidor de velocidad de flujo y en el totalizador.

Si cada dispositivo de entrada se instala adecuadamente, se calibra exacta y frecuentemente y se mantienen rigurosos registros, el usuario debería estar confiado de que tiene un sistema de medida muy exacto.

Ya que toda la computación se hace en forma digital, no hay derivas de temperatura en estos circuitos. Los computadores de flujo digitales dan excelente exactitud y estabilidad a largo plazo.

6.1 INFLUENCIAS DEL CAPÍTULO 21 DE API

Este capítulo del API se denomina "Medición de flujo usando sistemas electrónicos de medida". Con este documento API, los fabricantes y usuarios de la industria se han beneficiado. Los fabricantes tienen que cumplir unos requisitos mínimos cuando diseñan un producto para medir flujo y los usuarios de la industria tienen un documento de apoyo durante el proceso de licitación y otras referencias.

Este estándar ha ayudado a describir los trabajos internos del computador de flujo. Aspectos tales como almacenamiento de datos históricos, tasas de muestreo y metodología de cálculo se han descrito en este documento.

6.2 SEGURIDAD Y NORMATIVA

Desde su introducción en los 80's, el uso de los computadores de flujo se ha expandido a todos los mercados en el globo. Con este crecimiento ha aparecido la necesidad de cumplir con prácticas y estándares internacionales, regionales y locales. Estos estándares y prácticas abarcan calidad, interoperabilidad y seguridad del producto. Siglas tales como CE, CSA, CENELEC, BASEEFA, UL, y FM forman ahora parte del diccionario de la industria del control de procesos. (Ver tabla 4). En términos de computadores de flujo, no se puede esperar que un único dispositivo cumpla todos los estándares en cada esquina del mundo. Sin embargo, se está haciendo común encontrar computadores de flujo certificados para múltiples estándares. Los avances en el diseño del hardware electrónico han hecho posibles estas múltiples certificaciones para muchos computadores de flujo.

• CE--European Conformity: Conformidad Europea.
• CSA--Canadian Standards Association: Asociación Canadiense de Estándares.
• CENELEC--European Committee for Electrotechnical Standardization: Comité Europeo para Estandarización Electrotécnica.
• BASEEFA--British Approvals Service for Electrical Equipment in Flammable Atmosphere: Servicio de Aprobación Británico para Equipo Eléctrico en Atmósferas Inflamables.
• UL--Underwriters Laboratories: Laboratorios Subscriptores.
• FM--Factory Mutual: Comunidad de Fábricas.

Tabla 4. Siglas de Entidades Normativas

Desde su humilde comienzo como un simple contador mecánico, el computador de flujo ha crecido en las dos últimas décadas para ocupar un puesto significativo en las aplicaciones de medición de fluidos. Hoy en día se puede escoger entre una variedad de productos, desde los básicos hasta los sofisticados.

Actualmente, estos dispositivos pueden llevar a cabo avanzados cálculos que alguna vez pertenecieron solo al dominio de máquinas muy costosas. Los avances en el diseño electrónico y de software han incrementado en gran medida la flexibilidad, simplicidad de uso e interconectividad del computador de flujo.

7. DISEÑO DEL FIRMWARE

El firmware o software de la CPU se desarrolla en el lenguaje Forth descrito en el capítulo 5. Sin embargo, las pruebas preliminares de las normas del AGA (American Gas Asociation), se hicieron en un lenguaje de alto nivel llamado Matlab, el cual ayudó a corroborar los algoritmos propuestos por el estándar AGA y a entender la metodología de desarrollo de las fórmulas.

Las normas a implementar en este proyecto serán la AGA7, para medidor tipo turbina y la AGA8 para el cálculo del factor de compresibilidad (este factor es usado por la norma anterior). Esta última norma es la más extensa e involucra gran rigor matemático. El AGA8 tiene 2 versiones: la del 1985 y la de 1992. Se escogió esta última por su actualización. Además, esta norma propone 2 métodos para deducir el factor de compresibilidad llamados: Detail y Gross. Se seleccionó este último dado que es el de menos complejidad y además, se ajusta a las condiciones de operación del gasoducto, las cuales están dentro del rango de aplicación del método asegurando óptima exactitud. Además, el método Detail requiere una cromatografía completa (extendida) del gas en el sitio donde se instale el computador de flujo, y no todos los sitios la tienen.

En el presente capitulo se explica la implementación de cada una de estas normas y seguidamente se detallan algunas de las características del software embebido como las rutinas de interrupción y adquisición de señales; finalmente se expone el diagrama de flujo del código final ejecutable.

7.1 DESARROLLO DE LA NORMA AGA8-GROSS2

7.1.1 Teoría del Método. La ecuación de estado aplicada para hallar el factor de compresibilidad por este método, sigue un modelo tipo virial basado en el trabajo de Schouten (1990); este método también se conoce como el modelo "SGERG". Usa un rango de valor calorífico de gas natural, densidad relativa (gravedad específica) y contenidos diluidos como parámetros de caracterización generales, en lugar de la composición detallada de un gas natural. El modelo predice con gran precisión factores de compresibilidad de gases naturales que tengan concentraciones de componentes dentro de los rangos dados en la siguiente tabla, y con menos de 0.1% molar de agua y 0.05% molar de sulfuro de hidrógeno.

CANTIDAD	RANGO
Densidad Relativa[1]	0.554 a 0.87
Valor Calorífico Gross[2]	477 a 1150 Btu/scf
Valor Calorífico Gross[3]	18.7 a 45.1 MJ/m^3
Porcentaje Molar de Metano	45.2 a 98.3
Porcentaje Molar de Nitrógeno	0.3 a 53.6
Porcentaje Molar de Dióxido de Carbono	0.04 a 28.94
Porcentaje Molar de Etano	0.24 a 9.53
Porcentaje Molar de Propano	0.02 a 3.57
Porcentaje Molar total de Butanos	0.01 a 1.08
Porcentaje Molar total de Pentanos	0.002 a 0.279
Porcentaje Molar de Hexanos	0.0005 a 0.1004
Porcentaje Molar de Helio	0 a 0.158

Tabla 5. Rango de Aplicación del Método AGA8 Gross 2

[1] Condiciones de Referencia: Densidad Relativa a 60° F, 14.7 psia
[2] Condiciones de Referencia: Combustión a 60° F, 14.7 psia; densidad a 60° F, 14.7 psia
[3] Condiciones de Referencia: Combustión a 25° C, 0.101325 Mpa; densidad a 0° C, 0.101325 Mpa.

Una ecuación virial de estado es una expansión polinomial en densidad. Cada término de densidad es precedido por un coeficiente virial. Los coeficientes viriales son funciones de la temperatura y la composición. La aplicación de la ecuación virial de estado modelo "SGERG", trunca la ecuación virial del factor de compresibilidad después del tercer coeficiente virial. Este truncamiento provee una alta precisión para transmisión en gaseoductos y condiciones de distribución, con límites en presiones y densidades.

El método gross aproxima una mezcla de gas natural tratándola como una mezcla de tres componentes: un componente equivalente de hidrocarbono (por ejemplo un componente pseudo-hidrocarbonado), nitrógeno y dióxido de carbono. El equivalente de hidrocarbono, CH, es usado para representar colectivamente todos los hidrocarbonos encontrados en la mezcla del gas. Nitrógeno y dióxido de carbono son componentes diluidos.

El modelo "SGERG" fue desarrollado usando unidades internacionales .

El método gross predice el factor de compresibilidad de un gas natural dadas presión y temperatura, usando tres de los siguientes 4 parámetros:

1. El valor calorífico volumétrico con condiciones de referencia de 77° F, 14.696 psia (25°C, 0.101325 MPa) para valor calorífico molar ideal y 32°F, 14.696 psia (0°C, 0.101325 MPa) para densidad molar.
2. Densidad relativa (gravedad específica) con condiciones de referencia de 32° F 14.696 psia (0°C, 0.101325 MPa).

3. Fracción molar de dióxido de carbono
4. Fracción molar de nitrógeno.

Dos métodos usan estos parámetros. El método uno usa valor calorífico, densidad relativa y contenido de dióxido de carbono como entradas. El método dos usa densidad relativa contenido de dióxido de carbono y contenido de nitrógeno como entradas. El método utilizado es en este proyecto es el dos.

El modelo "SGERG" expresa el factor de compresibilidad en términos de la densidad molar (d), el segundo coeficiente virial de la mezcla (B_{mix}), y el tercer coeficiente virial de la mezcla (C_{mix}):

$$Z = 1 + B_{mix}\, d + C_{mix}\, d^2$$

$$B_{mix} = \sum_{i=1}^{N} \sum_{j=1}^{N} B_{ij}\, x_i x_j$$

$$C_{mix} = \sum_{i=1}^{N} \sum_{j=1}^{N} \sum_{k=1}^{N} C_{ijk}\, x_i x_j x_k$$

Donde :
Z= factor de compresibilidad
B_{mix}= segundo coeficiente virial de la mezcla
C_{mix}= tercer coeficiente virial de la mezcla
D= densidad molar (moles por unidad de volumen)
B_{ij}= componente individual de interacción del segundo coeficiente virial
C_{ijk}= componente individual de interacción del tercer coeficiente virial
x_i x_j x_k= fracciones molares de los componentes del gas
N= número de componentes de la mezcla del gas

Los índices i, j, k, representan componentes moleculares de la mezcla del gas. El modelo "SGERG" trata al gas natural como si estuviera basado en solo tres componentes, donde los componentes de CO_2 y N_2 son los diluidos. Todos los componentes hidrocarbonados son tratados como un simple componente equivalente hidrocarbonado, CH.

Los términos B_{ij} y C_{ijk} son los coeficientes viriales de interacción. Estos son dependientes de la temperatura. Para aplicaciones en gas natural con los límites específicos de esta ecuación, la dependencia de temperatura para estos términos es una función cuadrática.

Expandiendo las ecuaciones anteriores para B_{mix} y C_{mix} e identificando todos los términos necesarios para resolver el modelo "SGERG", se tiene:

$$B_{mix} = B_{CO2\text{-}CO2}\, x^2_{CO2} + B_{N2\text{-}N2}\, x^2_{N2} + B_{CH\text{-}CH}\, x^2_{CH} + 2\, B_{CO2\text{-}N2}\, x_{CO2} x_{N2} + 2\, B_{CO2\text{-}CH}\, x_{CO2} x_{CH} + 2\, B_{N2\text{-}CH}\, x_{N2} x_{CH}$$

$$\begin{aligned} C_{mix} = {} & C_{CO2\text{-}CO2\text{-}CO2}\, x^3_{CO2} + C_{N2\text{-}N2\text{-}N2}\, x^3_{N2} + C_{CH\text{-}CH\text{-}CH}\, x^3_{CH} \\ & + 3\, C_{CO2\text{-}CO2\text{-}N2}\, x^2_{CO2}\, x_{N2} + 3\, C_{CO2\text{-}CO2\text{-}CH}\, x^2_{CO2}\, x_{CH} \\ & + 3\, C_{CO2\text{-}N2\text{-}N2}\, x^2_{N2}\, x_{CO2} + 3\, C_{CO2\text{-}CH\text{-}CH}\, x^2_{CH}\, x_{CO2} \end{aligned}$$

$$+ \; 3\, C_{N2\text{-}N2\text{-}CH}\, x^2_{N2}\, x_{CH} + 3\, C_{N2\text{-}CH\text{-}CH}\, x^2_{CH}\, x_{N2}$$
$$+ \; 6\, C_{CO2\text{-}N2\text{-}CH}\, x_{CO2}\, x_{N2}\, x_{CH}$$

A continuación se muestran las ecuaciones y métodos necesarios para calcular los términos iterativos del segundo y tercer coeficiente virial de las ecuaciones anteriores.

Ø Coeficientes viriales Iterativos para Nitrógeno y Dióxido de Carbono

Los valores B_{ij} de los términos para nitrógeno y dióxido de carbono son expresados en (dm^3/mol) y son dados por:

$$B_{ij} = b_0 + b_1 T + b_2 T^2$$

Donde valores de b_0 , b_1 y b_2 son dados en la siguiente tabla , T es la temperatura en Kelvin.
De igual modo los valores de C_{ijk} de los términos para nitrógeno y dióxido de carbono y son expresados en (dm^6/mol^2) y son dados por:

$$C_{ijk} = c_0 + c_1 T + c_2 T^2$$

Donde los valores de c_0 , c_1 y c_2 son dados en la siguiente tabla , T es la temperatura en Kelvin.

Fluid for B_{ij}	b_0 (dm^3/mol)	b_1 (dm^3/mol)	b_2 (dm^3/mol)
N2-N2	-0.144600	$0.74091*10^{-3}$	$-0.911950*10^{-6}$
CO2-CO2	-0.868340	$0.40376*10^{-2}$	$-0.516570*10^{-5}$
CO2-N2	-0.339693	$0.161176*10^{-2}$	$-0.204429*10^{-5}$

Tabla 6. Coeficientes viriales iterativos para Nitrógeno y Dióxido de Carbono

Fluid for C_{ijk}	c_0 (dm^6/mol^2)	c_1 (dm^6/mol^2)	c_2 (dm^6/mol^2)
N2-N2-N2	$0.78498*10^{-2}$	$-0.39895*10^{-4}$	$0.611870*10^{-7}$
CO2-CO2-CO2	$0.205130*10^{-2}$	$0.34888*10^{-4}$	$-.837030*10^{-7}$
CO2-N2-N2	$0.552066*10^{-2}$	$-0.168609*10^{-4}$	$0.157169*10^{-7}$
CO2-CO2-N2	$0.358783*10^{-2}$	$0.806674*10^{-5}$	$-0.325798*10^{-7}$

Tabla 7. Coeficientes viriales Iterativos para el equivalente hidrocarbonado, CH

Los coeficientes viriales restantes y fracciones molares necesitadas para el cálculo del factor de compresibilidad de un gas, son las cantidades referentes a los equivalentes hidrocarbonados, CH. Los términos Iterativos del segundo y tercer coeficiente virial para los equivalentes hidrocarbonados, CH deben ser calculados del valor calorífico molar ideal gross del equivalente hidrocarbonado (H_{CH} en KJ/mol a 25^0 C y 0.101325 MPa). EL valor calorífico molar puede ser determinado por uno de dos métodos que son expuestos en el apéndice B del reporte AGA 8 GROSS. En este caso, como se expuso anteriormente se utilizará el método dos.
Las ecuaciones para el segundo y tercer coeficientes viriales Iterativos, del equivalente hidrocarbonado son:

$$B_{CH\text{-}CH} = B_0 + B_1 H_{CH} + B_2 H^2_{CH} \quad y$$

$$C_{CH\text{-}CH\text{-}CH} = C_0 + C_1 H_{CH} + C_2 H^2_{CH}$$

Donde B_0 , B_1 , B_2 , C_0 , C_1 , y C_2 son funciones dependientes de temperatura, definidas así :

$$B_i = b_{i0} + b_{i1} T + b_{i2} T^2 \quad i = 0,1,2$$

$$C_i = c_{i0} + c_{i1} T + c_{i2} T^2 \quad i = 0,1,2$$

Las constantes en las ecuaciones anteriores son dadas en la siguiente tabla , T en Kelvin.

	i	b_{i0}	b_{i1}	b_{i2}
$B_0(dm^3/mol)$	0	-0.425468	$0.2865*10^{-2}$	$-0.462073*10^{-5}$
$B_1(dm^3/KJ)$	1	$0.877118*10^{-3}$	$-0.556281*10^{-5}$	$0.881510*10^{-8}$
$B_2(dm^3mol/KJ^2)$	2	$-0.824747*10^{-6}$	$0.431436*10^{-8}$	$-0.608319*10^{-11}$
	i	c_{i0}	c_{i1}	c_{i2}
$C_0 (dm^6/mol^2)$	0	-0.302488	$0.195861*10^{-2}$	$-0.316302*10^{-5}$
$C_1(dm^6/mol^2)$	1	$0.646422*10^{-3}$	$-0.422876*10^{-5}$	$0.688157*10^{-8}$
$C_2 (dm^6/mol^2)$	2	$-0.332805*10^{-6}$	$0.223160*10^{-8}$	$-0.367713*10^{-11}$

Tabla 8. Coeficientes viriales Iterativos para el equivalente hidrocarbonado, CH

El segundo coeficiente virial de interacción para el equivalente hidrocarbonado CH , con nitrógeno N_2 es calculado usando la relación:

$$B_{N2\text{-}CH} = (0.72 + 1.875 * 10^{-5} (320 - T)^2) (B_{N2\text{-}N2} + B_{CH\text{-}CH})/2$$

Donde T está dado en kelvins. Para el equivalente hidrocarbonado CH, con el dióxido de carbono CO_2 , la relación es:

$B_{CO2\text{-}CH} = -0.865(B_{CO2\text{-}CO2}\ B_{CH\text{-}CH})^{1/2}$

El tercer coeficiente virial iterativo para el dióxido de carbono y nitrógeno con el equivalente hidrocarbonado son calculados así :

$C_{N2\text{-}CH\text{-}CH} = (0.92 + 0.0013\ (T - 270))\ (C^{2}_{CH\text{-}CH\text{-}CH}\ C_{N2\text{-}N2\text{-}N2})^{1/3}$

$C_{N2\text{-}N2\text{-}CH} = (0.92 + 0.0013\ (T - 270))\ (C^{2}_{N2\text{-}N2\text{-}N2}\ C_{CH\text{-}CH\text{-}CH})^{1/3}$

$C_{CO2\text{-}CH\text{-}CH} = 0.92\ (C^{2}_{CH\text{-}CH\text{-}CH}\ C_{CO2\text{-}CO2\text{-}CO2})^{1/3}$

$C_{CO2\text{-}CO2\text{-}CH} = 0.92\ (C^{2}_{CO2\text{-}CO2\text{-}CO2}\ C_{CH\text{-}CH\text{-}CH})^{1/3}$

$C_{CO2\text{-}N2\text{-}CH} = 1.10\ (C_{CO2\text{-}CO2\text{-}CO2}\ C_{N2\text{-}N2\text{-}N2}\ C_{CH\text{-}CH\text{-}CH})^{1/3}$

La ecuación para el calculo de la presión usando el método GROSS, es obtenida por sustitución de la ecuación:

$Z = 1 + B_{mix}d + C_{mix}d^{2}$ en

$d = P/ZRT$ resultando asi:

$P = dRT(1 + B_{mix}d + C_{mix}d^{2})$

7.1.2 Implementación del Método. Se implementó la norma AGA 8 Gross 2 primero en Matlab y seguidamente en Forth siguiendo los procedimientos de cómputo propuestos por esta norma para el cálculo del factor de compresibilidad. Este método solicita como entrada el previo conocimiento de la densidad relativa y las fracciones molares de N_2 y CO_2; y por otro lado la medición de la temperatura y la presión en línea.

Los siguientes 5 pasos fueron empleados para el cómputo del factor de compresibilidad Z (se lista la referencia de página de la norma para mayor facilidad):

1. Se hace la entrada de información de caracterización del gas, correspondiente al método dos de la tabla B.2-1 de la norma AGA 8 pg. 84. El computo del valor molar gross del hidrocarburo equivalente, y otros parámetros son resueltos, usando el método dos del apéndice B.2 de la norma AGA 8 pág. 87.
2. Se hace la entrada de la temperatura y presión las cuales calculan el factor de compresibilidad deseado.
3. Se calculan el segundo y tercer coeficiente virial, Bmx y Cmx en la ecuación de estado. Estos coeficientes son función de la temperatura absoluta, el calor molar gross del hidrocarburo y la fracción molar de los componentes del gas, que no son hidrocarburos.

4. La densidad molar *d*, es calculada usando la ecuación de estado para la presión, dada en la ecuación 43 del AGA 8 pág. 33.
5. El factor de compresibilidad Z es calculado usando la ecuación 25 del AGA 8 pág. 29.

§ Computo de otras cantidades:

Con base en el factor de compresibilidad y la fracción molar y usando las relaciones dadas en las ecuaciones 1 a 12 de la pág. 25 y el apéndice C pág. 125 del AGA 8. También pueden ser calculadas otras cantidades como la densidad de masa y el factor de compresibilidad tanto en las condiciones contractuales como las de flujo.

La próxima figura representa un diagrama de flujo que ilustra el uso del programa.

Un programa de aplicación y cuatro subrutinas son mostradas en la figura. Las rutinas usadas para cada cálculo son mostradas a la izquierda del bloque de cómputo.

Los bloques y las principales subrutinas son:

1. Inicialización de constantes (PARAMGS).
2. Cálculo de las cantidades que dependen de la caracterización del gas (CHARGS).
3. Cálculo de las cantidades que dependen de la temperatura (VIRGS).
4. Cálculo de las cantidades que dependen de la densidad (PGROSS, DGROSS, ZGROSS).

Para un cálculo eficiente, el programa de aplicación debe llamar las subrutinas en la secuencia mostrada en la figura 36. Esta secuencia es eficiente porque minimiza cálculos innecesarios.

A continuación en la figura 35 y 36 se muestra el diagrama de flujo del método Gross-2.

Figura 35. Diagrama de Flujo para hallar H_{CH}

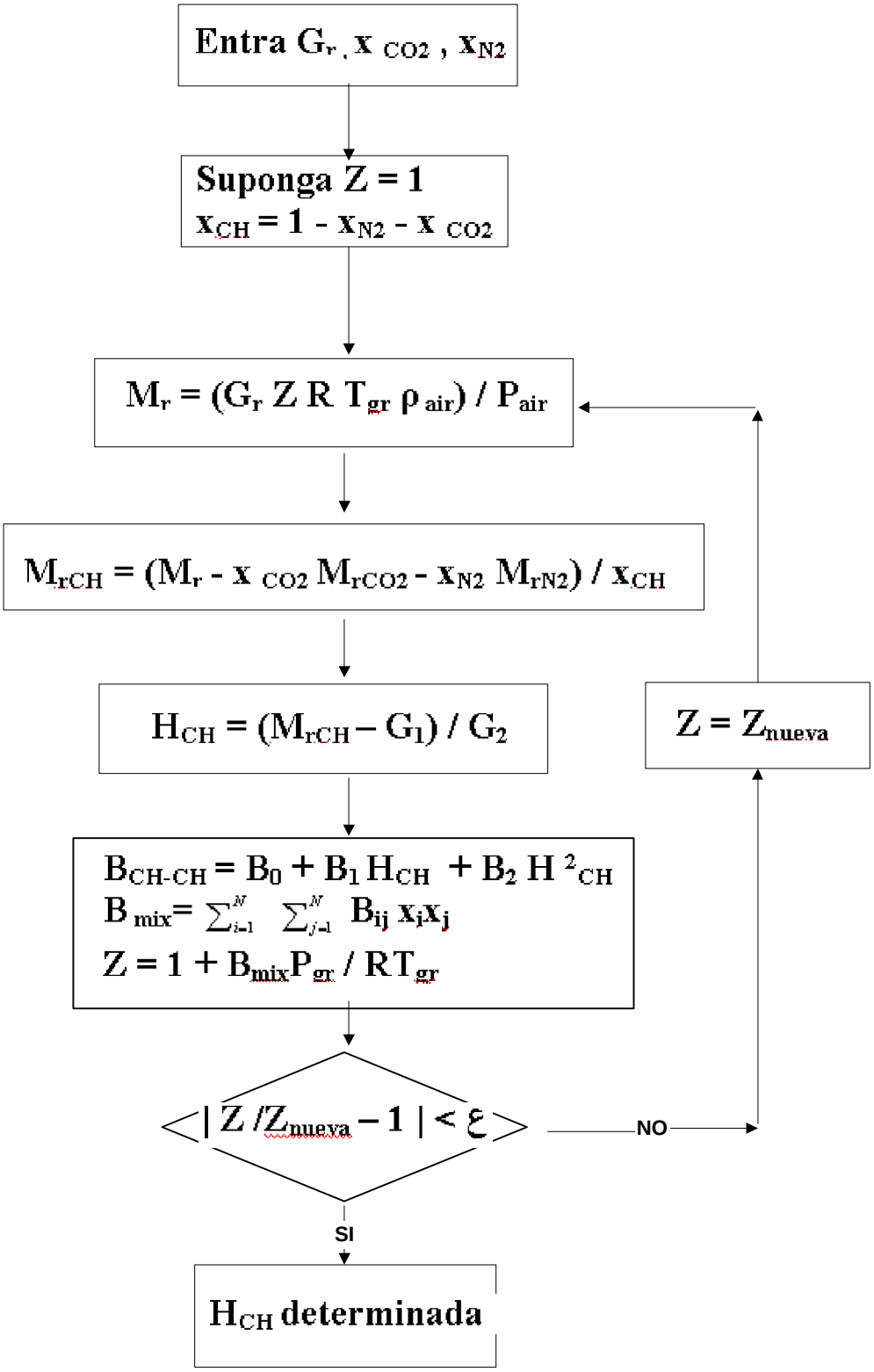
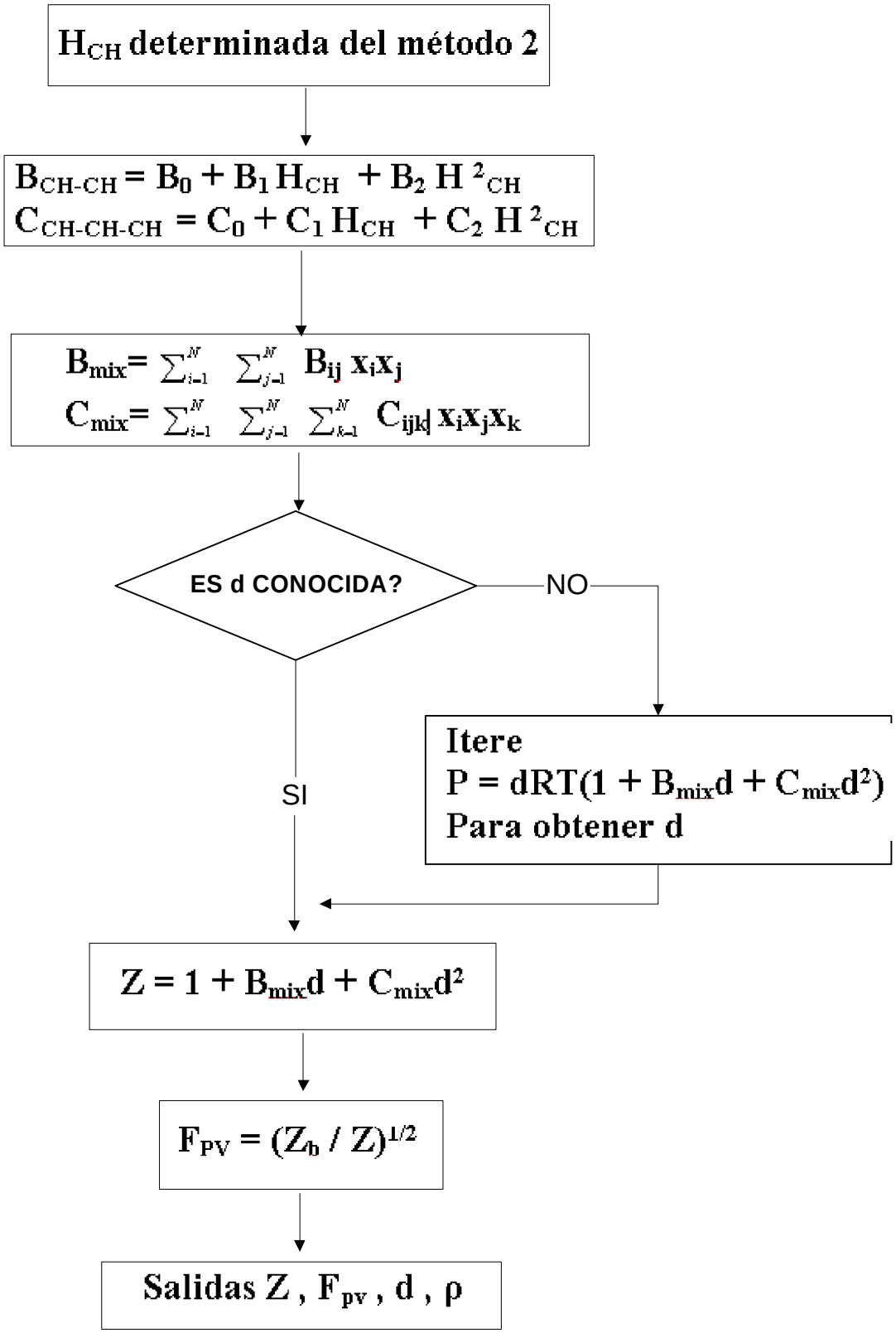


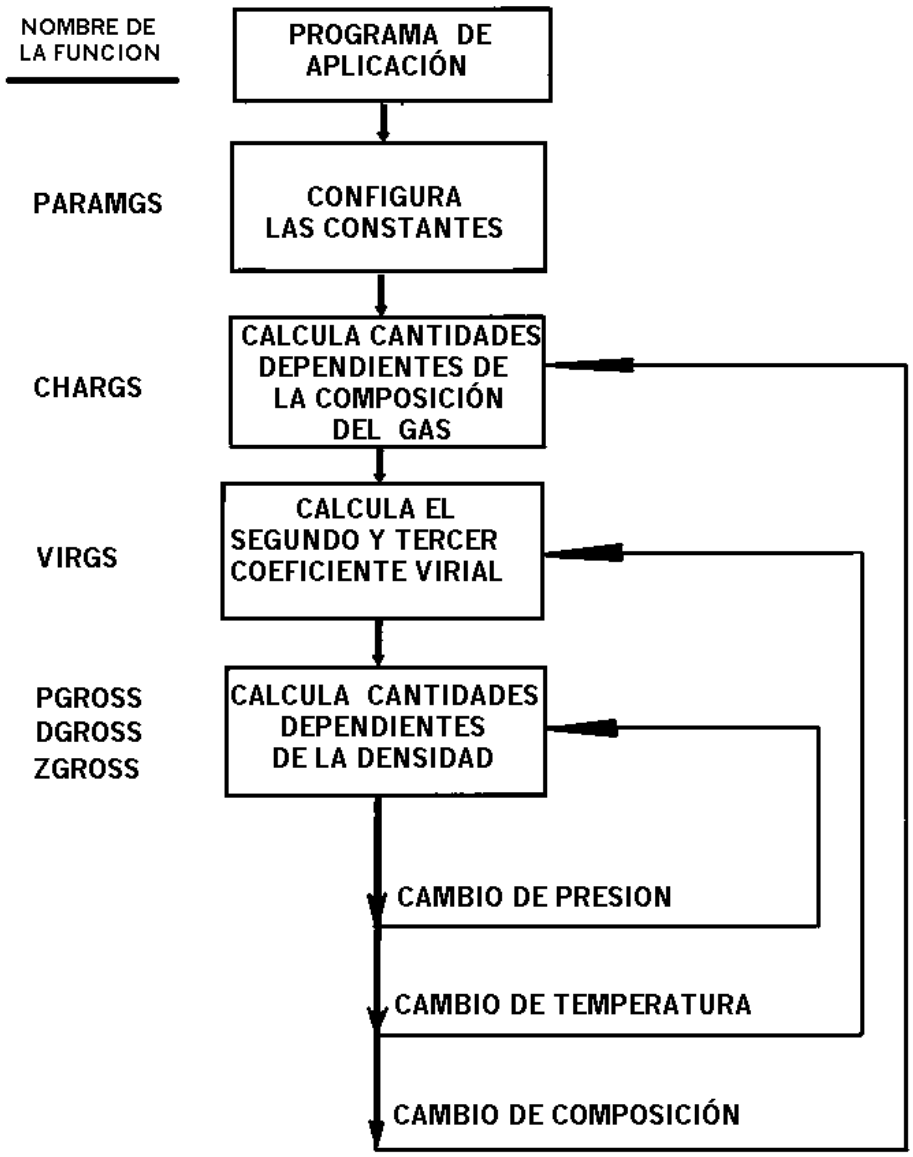
Figura 36. Diagrama de Flujo AGA8 Gross-2



§ Cálculo de cantidades que dependen de los componentes:

En el primer bloque de cálculo las cantidades de la ecuación de estado y el peso molecular, son especificados. Para mayor eficiencia en cálculos, la subrutina PARAMGS es llamada solo cuando empieza la aplicación.

Figura 37. Diagrama de Flujo Rutinas AGA8 Gross-2



§ Cálculo de cantidades que dependen de la caracterización del gas:

En el segundo bloque de cálculo, la caracterización de la mezcla del gas es identificada. Esta subrutina es llamada de nuevo, solo si hay cambios en la caracterización de la mezcla del gas.

§ Cálculo de cantidades que dependen de la temperatura:

En el tercer bloque de cálculo, la temperatura debe ser especificada. VIRGS es llamada por DGROSS. VIRGS calcula el segundo y tercer coeficiente virial de la mezcla del gas a una temperatura absoluta T. La subrutina VIRGS nunca es llamada a menos que haya un cambio en la temperatura del gas.

§ Cálculo de cantidades que dependen de la densidad:

El último bloque determina las cantidades que dependen de la densidad. La función DGROSS requiere de la presión absoluta P como entrada. DGROSS usa el método Chamber (Referencia 4, apéndice B.8 AGA 8) para hallar la densidad de la mezcla del gas a temperatura absoluta T y presión absoluta P. Los límites de este método están especificados en el modelo SGERG. DGROSS llama la función PGROSS que calcula la presión para una temperatura y densidad específicas. La función PGROSS llama a ZGROSS que calcula el factor de compresibilidad para una temperatura y densidad específicas.

A continuación se exponen las subrutinas usadas:

El código fuente en Matlab de las subrutinas listadas a continuación está disponible en el anexo 1 y en lenguaje Forth en el anexo 2.
Seguidamente se explica la función de cada subrutina.

§ SUBRUTINA CHARGS.

Determina el valor calorífico del hidrocarburo equivalente, usando el método dos del apéndice B.2 del AGA 8.
Los valores de entrada de la subrutina son: el número del método (2) ; el calor volumétrico (KJ/dm3), la densidad relativa y cinco elementos de un arreglo que contiene las fracciones molares de N2, CO2, H2 y CO. CHARGS retorna el factor de compresibilidad base y la densidad a 60 F y 14.73 psia.

§ SUBRUTINA DGROSS.

Determina la densidad molar de una mezcla de gas natural dadas presión y temperatura. La presión debe estar en Mpa y la temperatura debe estar en Kelvin. La densidad es hallada en mol/dm^3.

§ SUBRUTINA PARAMGS.

Inicializa las constantes utilizadas por la ecuación de estado SGERG. Estas constantes incluyen el peso molecular del nitrógeno y de dióxido de carbono, la constante del gas y las constantes usadas en la ecuación de estado.

§ SUBRUTINA PGROSS.

PGROSS calcula la presión de la mezcla del gas natural como una función de densidad y temperatura. La densidad debe estar en mol/dm^3 y la temperatura debe estar en kelvin. La presión es retornada en Mpa.

§ SUBRUTINA VIRGS.

VIRGS calcula el segundo y tercer coeficiente virial de la mezcla del gas a una temperatura dada. Los coeficientes constantes de la tabla 7 y 8 del AGA8 (pgs. 31 y 32) son almacenados en arreglos (matrices) y combinados usando reglas definidas por las ecuaciones 30 al 42 del AGA 8 (pg. 30 en adelante).

§ SUBRUTINA ZGROSS.

ZGROSS calcula el factor de compresibilidad de la mezcla del gas natural como una función de la densidad y la temperatura. La densidad debe estar en mol/dm3 y la temperatura debe estar en kelvin.

7.2 DESARROLLO DE LA NORMA AGA7

El medidor de turbina es un dispositivo de medición de velocidad. Este depende del flujo del gas, que causa que el rotor del medidor gire a una velocidad proporcional la rata de flujo. Las revoluciones del rotor son contadas mecánica o eléctricamente, y son convertidas a un registro volumétrico total continuo.

El volumen registrado está a las condiciones de presión y temperatura del sistema en ese momento, por lo cual deberá ser corregido a las condiciones base especificadas para propósitos de facturación.

La ley de los gases se expresa así :

- (Pf) (Vf) = (Zf) (N) (R) (Tf) para condiciones de flujo
- (Pf) (Vb) = (Zb) (N) (R) (Tb) para condiciones base

V= volumen
Z = factor de compresibilidad
N = numero de moles del gas

T = temperatura absoluta
R = constante universal de gases
f= condiciones de flujo
b = condiciones base

- RATA DE FLUJO A CONDICIONES DE FLUJO

QF = VF / T
Donde
QF = rata de flujo a condiciones de flujo
VF = volumen en un periodo determinado a condiciones de flujo
= pulsos totales x 1/K sobre la salida eléctrica.
T = tiempo
K = pulsos por pies cúbicos

- RATA DE FLUJO A CONDICIONES BASE

Qb = Qf Fpm Ftm Zb/Zf
Donde:
Fpm = factor de presión
Fpm = Pf/Pb
Donde
Pf = presión de flujo en psia
Pb = presión base o contractual

Ftm = factor de temperatura de flujo
Ftm = Tb/ Tf
Donde
Tb = temperatura base o contractual en ^{o}R
Tf = temperatura de flujo en ^{o}R

Zb/Zf = multiplicador de compresibilidad
Donde
Zb = compresibilidad a condiciones base
Zf = compresibilidad a condiciones de flujo

La ecuación de rata de flujo a condiciones base; puede expresarse también así :

Qb = (C) (K) (Qf/1000) ((Pf + Pa)/Pb) (519.67/(Tf + 459.67)) ((Tb + 459.67)/519.67) Fpv^2

Donde
Qb = rata de flujo a condiciones base (MSCFH)
C = factor de calibración
K = span scaling factor
Qf = rata de flujo a condiciones de flujo (pies cúbicos por hora)
Pf = presion estatica psig

Pa = presión atmosférica psia
Pb = presión base en psi
Tf = temperatura de flujo en grados F
Tb = temperatura base en grados F
Fpv = factor de supercompresibilidad

7.3 CARACTERISTICAS DEL SOFTWARE EMBEBIDO

A continuación se expondrán las principales características que ya posee el software del sistema embebido.

7.3.1 Conversión análogo digital. El puerto 8 es configurado como un puerto paralelo de 8 bits de entrada. Para usarlo como 8 canales de 10-bits A a D se llama la palabra de conversión A-D que viene en el sistema Forth TDS2020F. Esta no sólo hace la conversión analógica , también establece los registros necesarios para convertir el puerto 8 a 8 entradas análogas y para hacer selección de canal. Por ejemplo 3 A-D retornará en la pila el valor digital del canal 3.

La cuenta de "full" escala es de 1023 y corresponde con una entrada igual al voltaje de referencia nominal de +5v,y un resultado de 0 es dado cuando la entrada digital es conectada a tierra.

Las siguientes definiciones son palabras para obtener un número en milivoltios.

```
: MV ( canal – milivoltios)  A-D 5000 1023 */ ;
```

Una rápida versión de A-D es disponible como una librería llamada #A-D.TDS, la tabla toma tiempos de conversión aproximados:

	Usando H8/532 PROM	Usando PROM Externa
Basado en A-D	127µs	249µs
Archivo #A-D.TDS	21µs	32µs

Tabla 9. Tiempos de Conversión en A-D, TDS.

El voltaje de referencia para el A-D es de +5V para el TDS2020*F*. El regulador usado es de 5.00±0.05V. Si esta referencia es necesitada externamente, se toma de los pines AVCC y AGND. Estos tienen los voltajes actuales de alimentación para el conversor A a D.

7.3.2 Relojes y Contadores. Las librerías que asisten a operaciones con relojes y contadores en el TDS2020F son:

#COUNTER.TDS	Contador de pulsos de 32-bits. Palabras que recuperan el conteo del contador, lo ponen en cero, arranque y parada.
#FREQ.TDS	Medida de frecuencias hasta 1MHz.
#PERIOD.TDS	Medidas de señales con periodos entre 18.8Hz y 33kHz.
#TIMING.TDS	Crea retardos en milisegundos en tiempo real aún cuando se use multitarea. Obtiene tiempos reales como números sin signo de 48-bit , precisión de 814ns.

Tabla 10. Relojes y contadores.

Se tienen cuatro relojes hardware uno es de 8 bits y 4 de 16 bits:

- Ø Los relojes de 8-bit pueden ser sincronizados internamente a 814ns, 6.51µs o 104ms o también pueden ser habilitados usando un reloj externo , para ser usado como un contador de eventos externo.

- Ø Los timers de 16-bit pueden ser fijados a incrementos internos cada 470ns, 814ns o 3.255µs y alguno de los 3 puede usar un reloj externo.

Los pines de estos contadores están principalmente en el Puerto 7, el cual tiene entradas tipo "Schmitt trigger". Esto significa que si la señal suministrada tiene niveles lógicos imperfectos serán corregidos.

Todos los cuatro relojes pueden ser leídos y programados por software y cada uno puede generar una interrupción en desbordamiento para permitir extensión a 32 bits o más. Cada reloj tiene dos salidas que capturan los registros que pueden ser, una bandera cuando el contador corresponde al valor de salida. Las 8 banderas pueden ser chequeadas usando interrupciones o generando salidas hardware. Cada uno de los 3 contadores de 16 bits también tienen un modo de captura de entradas en recepción de una señal externa, el valor del contador es transferido a registros de 16 bits y el tiempo exacto de transición puede ser leído por software.

7.3.3 Fecha y Hora. Los siguientes son las librerías disponibles para el manejo de la fecha y hora en el sistema.

#TIMEEUR.TDS	Mantiene el TDS2020*F* en sincronía con el tiempo nacional.
#TIMEUK.TDS	Estándares para telefonía de radio transmisiones.
#TIMEUSA.TDS	
#DIARY.TDS	Días de la semana, días del año y número de semana, calculados diariamente .

Tabla 11. Fecha y hora.

RELOJ DE TIEMPO REAL:	PCF8583 "chip" de reloj en el bus I^2C mantenido por una batería en el TDS2020BYN ("piggyback") TDS2020BYD (separada) batería de tarjeta o batería externa.
RELOJ DE TARJETA:	Operada solo por software. No trabaja cuando está desenergizada o en "standby".

Tabla 12. Mecanismos de reloj en el TDS2020*F*:

En una aplicación de memoria no volátil que necesite fecha y hora se usan las palabras NOW y HRS para seleccionar la fecha y hora actual en le reloj de tarjeta, luego se usa W! para transferir una copia al reloj de tiempo real.

Por ejemplo a las 9pm en navidad:

```
25.12.01 NOW   21.00.00 HRS W!
```

Si se apaga la tarjeta; el reloj de tarjeta se perderá pero una batería conectada al "pin" VBAT (z3), mantendrá el reloj de tiempo real. Cuando se enciende la tarjeta, el reloj de tarjeta tendrá la fecha y hora por defecto 1.01.84 y 00.00.00 . Se debe usar w@ para establecer la nueva hora en el reloj de tarjeta a partir del reloj de tiempo real.

El reloj de tarjeta puede ser visto mas fácilmente con .DATE y .TIME los cuales despliegan vía puerto serial a su terminal (a menos que EMIT este redireccionado).

La razón de este esquema es evitar retardos en los tiempos obtenidos. Usando reloj de tarjeta una aplicación puede mostrar el momento actual para el microsegundo mas cercano.

Para guardar el sincronismo del reloj se sugiere que una aplicación use w@ a intervalos regulares (una vez por hora o por día), su tiempo de ejecución de alrededor de 4.2ms no será importante (7.4ms cuando se usa la tarjeta de desarrollo TDS2020DV).

Ejemplo:

```
   12.25.01 NOW   21.00.00 HRS W!
```

y cuando la fecha es recuperada con .DATE la respuesta es 12.25.01.

Ø RELOJ NO-VOLATIL Y RAM

El reloj PCF8583 tiene 8 bytes para el reloj , 8 bytes de alarma y 240 bytes de RAM. La conexión del accesorio TDS2020BYN (batería) tornará al reloj no volátil con un consumo típico de 3,5µA incluyendo una RAM de 128k bytes. El TDS2020F contiene circuitos para evitar corrupción mientras se energiza o desenergiza. El sistema no realiza ningún uso de las direcciones internas $11

a $FF inclusive del PCF8583 , estos 239 bytes son disponibles como RAM no volátil. Las palabras W! y W@ proveen el acceso solo si se necesitan los registros de reloj.

Ø RELOJ DE TARJETA

Existen 3 palabras Forth usadas que retornan el valor de una variable, estas son TICKS, MINS y DAYS. Estas pueden ser accesadas directamente para obtener funciones especiales.
El reloj trabaja usando interrupciones de desbordamiento del reloj 3. Cada 53.333 ms esta interrupción incrementa el registro de "ticks" y cada minuto se resetea e incrementa los registros de minutos. Una vez por día los minutos de registros son reseteados y el registro día es incrementado.

La mayoría de palabras de manejo de tiempo y hora se trabajan con estas 3 variables. La interrupción está escrita en código de máquina y toma al menos el 0.1 % del tiempo del procesador.

Aunque el reloj 3 es usado para este propósito específico , también esta libre para otros más; por ejemplo ambos, captura de salida y captura de entrada, no son usados por la función de reloj.

Las interrupciones de tiempo pueden ser deshabilitadas mientras se leen registros para así evitar resultados no deseados, pero el tiempo no se perderá si ellas son rehabilitadas dentro de los siguientes 53.3 ms.

7.3.4 Soporte de Teclado.

INKEY	(– n)	Explora el teclado. n=0 si todas las teclas están sin oprimir. n=1 a 64 si alguna tecla es presionada.
NEWKEY	(– n)	Llama a INKEY pero solo retorna n=1 a 64 si una nueva tecla es oprimida.
LASTKEY	(– a)	Utiliza la variable contenida en el codigo de la ultima tecla encontrada por NEWKEY .

Tabla 13. Comandos usados para soporte de teclado.

Si una tecla es presionada durante múltiples llamadas de NEWKEY solo la primera vez mostrará cual es presionada. Esta es la palabra más usada.
Un 1 es puesto en cada una de las 8 salidas del Puerto B. Sin embargo las palabras de teclado ponen el puerto B en un modo especial mediante la cual estas salidas no aparecen de inmediato.

Ellas solo ocurren si el procesador llama la dirección HEX 82B0. Ocho veces en sucesión un diferente formato es puesto en el puerto B y entonces la dirección 81B0 es leída.

Una fila de la matriz es leída cada vez y un formato muestra cual tecla fue cerrada. El software decodifica 8 patrones para dar el número de tecla presionada. Si más de una tecla es presionada el número mayor es retornado.

Normalmente los tiempos de ejecución para la palabras de manejo de teclado son:

- INKEY 17µs
- NEWKEY 65µs

Estos son algunos ejemplos en el uso de palabras que manejan teclados.

El primer ejemplo es una palabra que espera hasta que una tecla es presionada. Esto es parecido a KEY en la entrada del puerto serial.

```
: KEY ( – n ) \ Espera por una tecla, n=1 a 64
   BEGIN REGULAR NEWKEY ?DUP UNTIL ;
```

La palabra REGULAR es incluida para mostrar que se puede tener el sistema haciendo muchas otras trabajos mientras espera por una tecla.

De esta forma se define REGULAR para incluir cualquier código que deba ejecutarse todo el tiempo, pero que guarde un tiempo de ejecución máximo de 100 ms así que la respuesta a una tecla no sea retardada.

Usualmente se necesita un valor asociado con cada tecla , el cual no siempre es una simple tecla numérica. Por ejemplo, en un teclado alfanumérico el código ASCII será buscado.
La conversión se realiza mejor usando un tabla de la siguiente forma:

```
HEX CREATE CODES \
   34 C, 65 C, 2C C, 78 C, BA C, 43 C, 4F C, 7E C,
   36 C, A2 C, 4D C, 69 C, 11 C, 0F C, EE C, 23 C,
   56 C, EF C, FF C, A2 C, 33 C, 89 C, 2A C, 12 C,
   9A C, C5 C, 5D C, 9E C, FC C, 23 C, 75 C, 82 C,
   E5 C, 6A C, A6 C, 9D C, E8 C, 4C C, 44 C, 73 C,
   7A C, AF C, BB C, 43 C, 68 C, 23 C, 11 C, 24 C,
   67 C, 87 C, 98 C, 63 C, 0F C, E3 C, E0 C, F0 C,
   55 C, FD C, AF C, F2 C, 45 C, 76 C, 9A C, 0D C,

: TRANSLATE ( n1 – n2 ) \ Cambia tecla de cod n1 a n2
   CODES + 1- C@ ; DECIMAL
```

En este ejemplo el código particular que se escogió para cada tecla fue arbitrario, el contenido real de la tabla dependerá de la aplicación.

El ejemplo final combina las dos anteriores y pone el código generado , asumido ASCII, en el LCD.

```
: REQUEST ( – ) \ Entrada de teclado pasa a LCD
   WIPE           \ limpia LCD
   PAD 40 BLANK   \ limpia buffer
   PAD            \ arranque del buffer
   BEGIN KEY TRANSLATE DUP $0D <>
   WHILE DUP LCDEMIT OVER C! 1+
   REPEAT 2DROP ;
```

Se escapa del la palabra REQUEST solo cuando se presiona la tecla que traduce el código 0D (ASCII de retorno de carro). El mensaje tipeado es puesto sobre el LCD y también dejado en el espacio temporal de memoria PAD para posterior procesamiento por el programa.

7.3.5 Manejo de LCD.

#CURSOR.TDS	Movimiento del cursor en LCDs alfanuméricos
#LM018.TDS	Corrige la posición de caracteres en LCDs de 2 lineas
#LM041.TDS	Corrige la posición de caracteres en LCDs de 4 lineas
#LM087.TDS	Corrige la posición de caracteres en LCDs 16 x 1
#EXTMESS.TDS	Cadenas de mensajes almacenadas fuera del diccionario
#MESSAGE.TDS	Cadenas de mensajes con índice

Tabla 14. Librerías usadas para el manejo de LCD.

Las rutinas de despliegue alfanumérico son implementadas en el TDS2020*F*. Use PRIME para poner automáticamente el LCD en secuencia de inicialización.

Por ejemplo suponga que tenemos un display con 2 líneas de caracteres, cada una 7x5 matriz de puntos. Lo siguiente puede dar las características requeridas.

```
: INITIALISE ( – ) \ Inicialización al energizar
  0          \ cursor no titila
  1          \ el cursor es requerido
  1          \ Display visible al encendido
  0          \ Display no se desplazará al escribir
  0          \ Dirección de carácter no se autoincrementa
  $38        \ Configura Display para 2 líneas y caracteres de 7 x 5
  PRIME ;    \ Deja el Display listo para uso.
```

Después de entrar la definición anterior, digite INITIALISE y "enter" para obtener el display listo para uso.

Para calcular el número requerido para el tipo de LCD usado empiece con 30 HEX. Añada 8 si el display es de dos líneas, y adicione 4 si el display es de matriz de 10 x 7 en vez de 7 x 5.

Display de 4 líneas alfanuméricas como el LM041 son eléctricamente como lo de 2 líneas. Aún un display de una línea de 16 caracteres desplegada es eléctricamente como uno 2 líneas de 8 caracteres.

También es de notar que aún en un display de dos líneas la segunda línea no sigue en numeración a la primera. Por ejemplo un LM018 tiene los caracteres 0 a 39 en la línea superior y los caracteres 64 a 79 en la inferior. Use los archivos como #LM041.TDS y #LM087.TDS como modelo para correctos posicionamientos.

Estos son algunas muestras del uso de palabras para manejo de LCD. Es aconsejable utilizar mensajes en el diccionario con la palabra ." transfiriéndolas al LCD revectorizando EMIT y usando LCDEMIT en vez de la palabra <EMIT> que escribe sobre el puerto serial.

Suponga que se quiere desplegar un mensaje como: " Muévase a la derecha 23 metros". El código básico puede ser:

```
: <LCD ( – ) \ Redirecciona EMIT al  LCD
   'EMIT @ R> 2>R ['] LCDEMIT 'EMIT ! ;
: LCD> ( – ) \ Restaura el vector
   2R> >R 'EMIT ! ;
: "START" ( – ) WIPE <LCD ." Listo para empezar" LCD> ;
: "GOING" ( – ) WIPE
   <LCD ." Mover a la       por       metros" LCD> ;
: "GONE" ( – ) WIPE <LCD ." Operación completada" LCD> ;
```

Ahora la palabra START mostrará que está listo para empezar, y GOING que la operación está en marcha. Para llenar con blancos el segundo mensaje, se hace lo siguiente:

```
: "RIGHT" ( – ) 5 AT ! <LCD ." derecha" LCD> ;
: "LEFT"  ( – ) 5 AT ! <LCD ." izquierda " LCD> ;
: "DISTANCE" ( n – ) \ muestra distancia n con
                     \  2 lugares decimales
   S>D               \ convierte n to un # doble de 32-bit
   14 AT !           \ usa la 14ava posición en el  display
   <LCD  2 .#  LCD> ; \ formato numérico en el LCD
```

Nótese que la dirección “left/right” y la distancia numérica pueden ser cambiados por fuera usando “GOING “ otra vez. Las palabras “RIGHT” “ LEFT” y “DISTANCE” se escriben para seleccionar áreas del LCD.

7.4 DESARROLLO DEL FIRMWARE EJECUTABLE

La palabra creada en Forth “ALL” la cual corre todo lo relacionado al teclado, el display y sus tres modos, sería el programa principal del ECOFLOW.

Paralelamente la toma de datos de presión y temperatura se hace por una interrupción periódica. La entrada de pulsos de la turbina es conectada directamente a una interrupción de hardware (IRQ1 activada por flanco). De este modo, cada vez que llega un nuevo pulso de la turbina se puede contabilizar el tiempo transcurrido entre el anterior pulso y éste, usando el número de TICKS del sistema con la instrucción @TIME que devuelve el número de ticks transcurridos en el día. (Un tick corresponde a 53.333ms).

Empleando este procedimiento de cálculo del intervalo de tiempo transcurrido entre un pulso y el siguiente, la frecuencia instantánea de flujo correspondería al inverso de este valor. Al tener la frecuencia instantánea se procede a situarla en la tabla de frecuencias de la turbina para extraer de ella la constante de calibración y la rata de flujo calibrada correspondiente a esa frecuencia instantánea calculada.

Las siguientes líneas de código muestra como se halla la frecuencia entre dos pulsos:

```
@TIME TNEW 2!!
1E0 TNEW 2@@ Told 2@@ D- D>F F/
18.75E0 F* FREQ F!!
TNEW 2@@ Told 2!!
```

Este esquema actualiza la rata de flujo instantáneamente tan pronto llega un pulso. Además como se manejan dos interrupciones, es necesario configurar el registro de interrupciones dándole mayor prioridad a la interrupción de hardware (IRQ1) que es que la que detecta los pulsos de la turbina. El código para esta configuración es:

```
: INIT-IRQ
  DIS                          \ Deshabilita interrupciones
  $FFFC C@ $60 OR $FFFC C!     \ Habilita IRQ0 y IRQ1
  $FFF0 C@ $88 AND             \ Les pone prioridad
  $FE OR $FFF0 C!              \ alta prioridad a IRQ1 (7)
  EIS                          \ Habilita interrupciones
;
```

Una rutina de manejo de interrupción corresponde a IRQ1 y que se activa aleatoriamente tan pronto sucede un pulso de la turbina y otra interrupción que se ejecuta periódicamente (cada 50ms) y está encargada de la toma de muestras de presión y temperatura así como de estar pendiente si ya llegó el fin de una hora y/o de un día.

La rutina para el manejo de interrupción IRQ1 de los pulsos se muestra a continuación.

```
: IRQ1 \ Se activa con cada flanco de caida en el pin C27

@TIME TNEW 2!! \ Saca el # de TICKS que van en el dia y lo guarda
STOP-LOG       \ Para la toma de muestras.

#IRQ1 2@@ D0= IF  @TIME Told 2!! AGA8-G2 THEN

#IRQ1 2@@ D0= NOT IF
 1E0
 TNEW 2@@ Told 2@@ D< IF
 1620000. Told 2@@ D- TNEW 2@@ D+  D>F F/  \ Halla la FREQ en hz
 18.75E0 F* FREQ F!!   \ transcurrido  entre dos interrupciones
 ELSE  TNEW 2@@ Told 2@@ D- D>F F/  \ Halla la FREQ en hz equivalente
 18.75E0 F* FREQ F!! THEN

 TNEW 2@@ Told 2!!     \ Guarda el numero de TICKS actual en Told
 360E3 1PULSE F@@ F* RQMAX F@@ F/ FREQ F@@ F* %RQFI F!!  \ % de rata

 INTERP-C \ Con % interpolo y hallo el VOL. CALIBRADO NO CORREG.(QFNC)

 QFNC-HR F@@ QFNC F@@ F+ QFNC-HR F!!
 CDAY-NC F@@ QFNC F@@ F+ CDAY-NC F!!

 FIRST @@ 1 = IF  \Si ya transcurrieron 2 seg donde se hallaron P T y Z
  AGA7       \ Se corrige el flujo con AGA7 y sale el vol. correg QFC
  QFC-HR F@@    QFC F@@ F+    QFC-HR F!!    \ Y se acumula en el sumador
  CDAY-C F@@ QFC F@@ F+    CDAY-C F!! \ a su vez en el CURR DAY corregido
 THEN

 #IRQ1 2@@ 3 UM/MOD DROP 0= IF AGA8-G2 THEN

THEN

#IRQ1 2@@ 1. D+ #IRQ1 2!!  \ incrementa numero de pulsos
#IRQ1-H 2@@ 1. D+ #IRQ1-H 2!!
#IRQ1-D 2@@ 1. D+ #IRQ1-D 2!!
START-LOG
RETURN;

-60 +ORIGIN ASSIGN IRQ1  \ Asigna la palabra IRQ1 a la interrupción IRQ1
```

Luego de hallar la frecuencia instantánea se calcula el porcentaje de rata equivalente usando:

```
360E3 1PULSE F@@ F* RQMAX F@@ F/ FREQ F@@ F* %RQFI F!!
```

Y con este valor se calibra el flujo con la tabla de la turbina pero aún no se ha corregido a la presión y temperatura contractuales. Por eso, seguidamente se llama a la rutina AGA7 para hallar la rata de flujo calibrada corregida habiendo previamente hallado (en un pulso anterior) el factor de compresibilidad Z con el llamado de la rutina denominada AGA8-G2.

Para la segunda rutina de interrupción asociada con la interrupción de software periódica se tiene el siguiente código:

```
: INT \ The interrupt routine, one pass through a regularly occurring

  STOP-LOG
  #INT @@ 2 MOD 0= IF  LOG THEN   \ Cada  2 DELAYS
  #INT @@ 1+ DUP #INT !!  \ Incrementa #INT
  40 MOD 0= IF           \ CADA 2 segs hace el PROMED de P y T
     PROM2               \ ." =PT= "
     FIRST @@ 1 = IF      \ Si NO es la primer vez que se enciende el CF
     AVERAGE  THEN \ entonces hace el promedio acumulativo
     FIRST @@ 0= IF       \ Si se acaba de encender el CF
       PREPR F@@ PREAV F!!  \ Establece condiciones iniciales
       TEMPR F@@ TEMAV F!!
       1 FIRST !! \ quita bandera de primera vez
       INIT-IRQ
     THEN \ y halla FPV (ZETA)para ser usado por AGA7 en IRQ1.
  THEN

 \ #INT @@ 830 MOD 0= IF AGA8-G2 THEN  \ Cada 8 segundos (OPCIONAL)

  #INT @@ 1200 MOD 0= IF   \ Cada 60 segundos guardo la PR y TEM
    PREAV F@@ PRES-W  #MIN @@ FLOATSD D+ 2DUP POINT-P 2!!  EF!
    TEMAV F@@ TEMP-W  #MIN @@ FLOATSD D+ 2DUP POINT-T 2!!  EF!
    #MIN @@ 1+ #MIN !!
    0 #INT !!              \  RESETEO DE INT
  THEN

  MINS @ 60 MOD            \ VERIFICA SI ES FIN DE HORA
  0= IF
     YAH @@ 0= IF
       1 YAH !!   PROMED-HR
     THEN
  ELSE  0 YAH !!
     0. #IRQ1-H 2!!         \  RESETEO DE IRQ-H
  THEN

  MINS @ 0= IF              \ VERIFICA SI ES FIN DE DÍA
     YAD @@ 0= IF
     1 YAD !!   PROMED-DY THEN
     ELSE  0 YAD !!
     0. #IRQ1-D 2!!         \  RESETEO DE IRQ-D
  THEN

  START-LOG
```

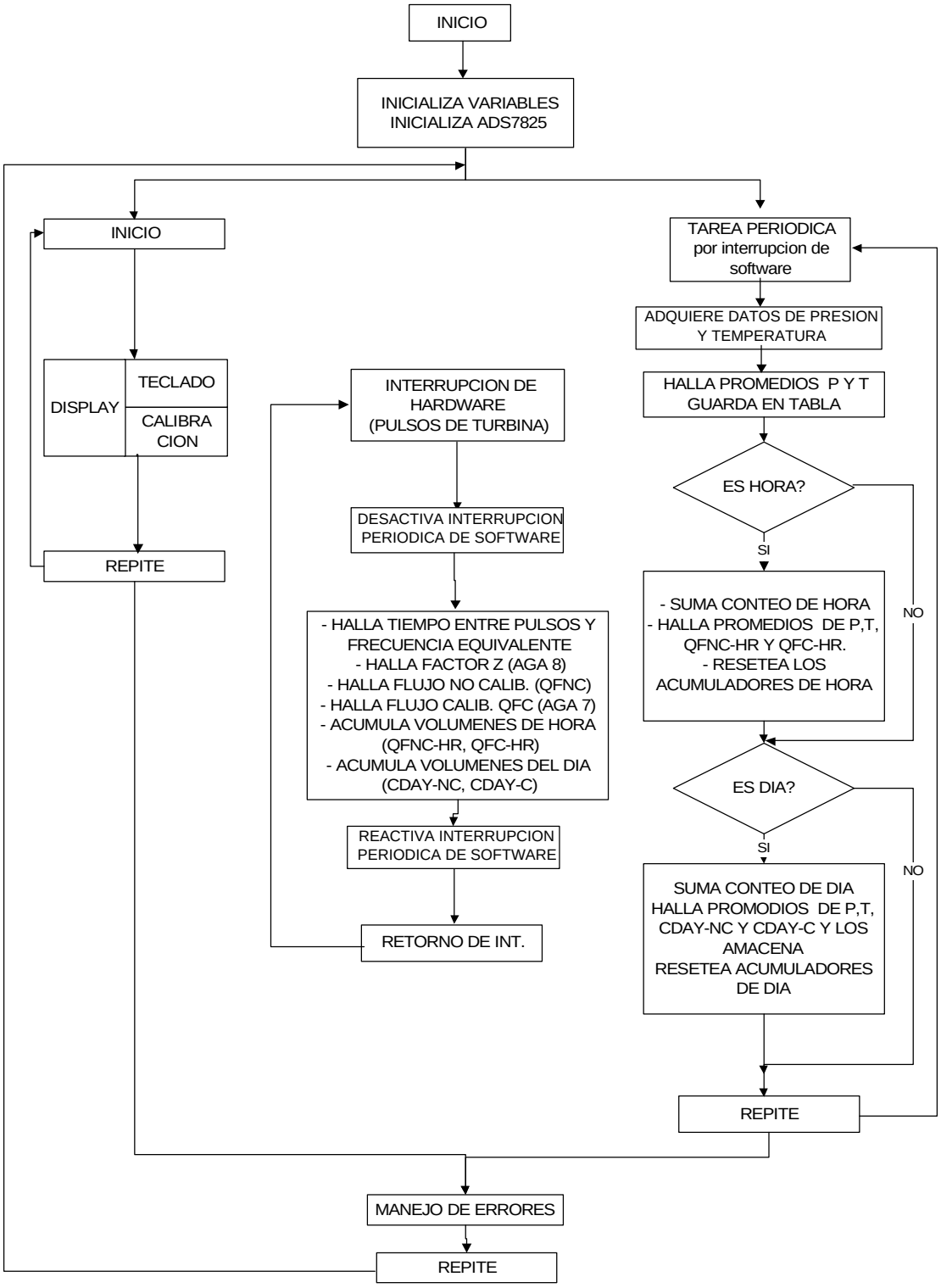
```
EDELAY @@ LATER RETURN;        \ tiempo fijo a la siguiente interrup.

-27 +ORIGIN ASSIGN INT  \ asigna la palabra INT a la int. Del Timer3
```

Con el uso de la palabra LOG se adquieren las muestras de presión y temperatura que se van almacenando y cada 2 segundos se muestra su promedio en el display. A su vez cada minuto se halla el promedio de cada uno de estos 2 segundos para luego en el fin de la hora calcular el promedio horario de presión y temperatura.

Así pues cada 2 segundos se obtienen 40 muestras de cada variable.

En base a todo este potencial de rutinas explicadas, se desarrolló el software final, el cual corresponde al diagrama de flujo mostrado en la siguiente figura y cuyas líneas de código con comentarios de explicación se detallan en el anexo dos.

Figura 38. Diagrama de Flujo del Firmware Ejecutable



8. PRUEBA EN CAMPO

8.1 ACTIVIDADES REALIZADAS

- Calibración de Sensores de Presión y Temperatura
- Instalación y Montaje en el City-Gate de Barranca
- Configuración del ECOFLOW2020T con las variables propias del lugar.
- Adecuación en campo de Hardware para lectura de pulsos.
- Ajuste de parámetros de muestreo para igualar lecturas.
- Afinación de algoritmos de corte de día.
- Puesta en marcha de la medición.

8.2 DESCRIPCIÓN DE LA PRUEBA

El computador de flujo ECOFLOW se instaló en serie con el computador de flujo base del city gate de Barranca (COGB), American Meter (AE5000). Se tomaron los registros de presión, temperatura, volumen corregido, y no corregido, de los dos equipos. Los parámetros de entrada para los computadores de flujo, fueron:

- Nitrógeno: 0,01563
- CO2: 0,00039
- Gravedad Específica: 0,565
- Presión Atmosférica: 14,519 psia
- Temperatura Base: 60 ºF
- Presión Base: 14,65 psig

8.3 RESULTADOS DE LA PRUEBA

La Tabla 14 y la Figura 15 muestran las lecturas de presión promedio día.

Lectura Presión (psig)Promedio Día			
Día	AE5000	ECOFLOW	% Error
1	239,92	239,75	0,07
2	239,16	237,99	0,49
3	238,88	236,80	0,87
4	240,27	238,46	0,75
5	239,90	240,00	0,04

Tabla 15. Presión Promedio

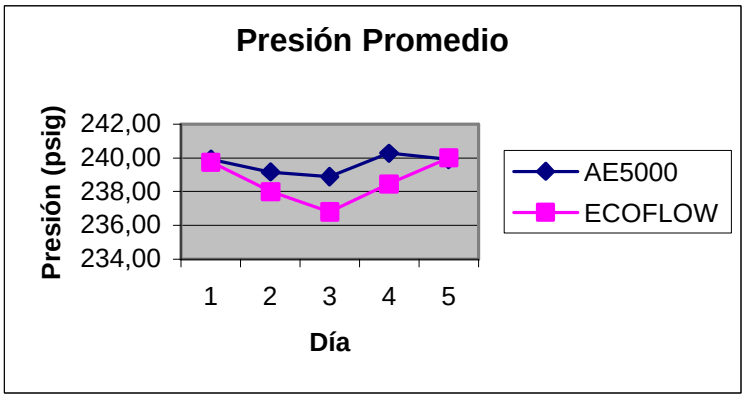


Figura 39. Presión Promedio

Lectura Temperatura (ºF) Promedio Día			
Día	AE5000	ECOFLOW	% Error
1	70,64	69,38	1,78
2	71,27	67,26	5,63
3	71,21	64,86	8,92
4	72,16	66,47	7,89
5	72,72	70,10	3,61

Tabla 16. Temperatura Promedio

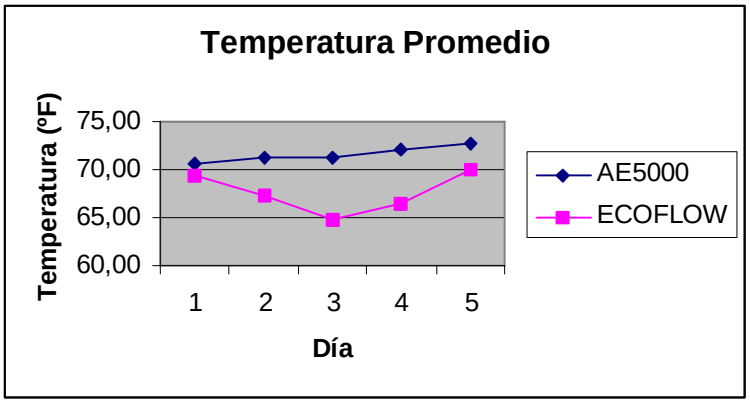


Figura 40. Temperatura Promedio

El volumen corregido es uno de los parámetros más importantes en la industria del gas, puesto que con dicho valor se realiza la transferencia de custodia. El volumen corregido proviene de un ajuste realizado al volumen no corregido. A continuación se presentan algunos reportes hora de los volúmenes corregido y no corregido tomados de los computadores de flujo durante los días de la prueba.

	AE5000		ECOFLOW	
Día 1 Time	CorVol MCF	UnCor MCF	CorVol MCF	UnCor MCF
07:00PM	45	2	46,65	2,66
08:00PM	28	2	28,87	1,64
09:00PM	21	1	21,65	1,23
10:00PM	14	1	14,40	0,82
11:00PM	11	1	10,80	0,61

Tabla 17. Volumen día_1.

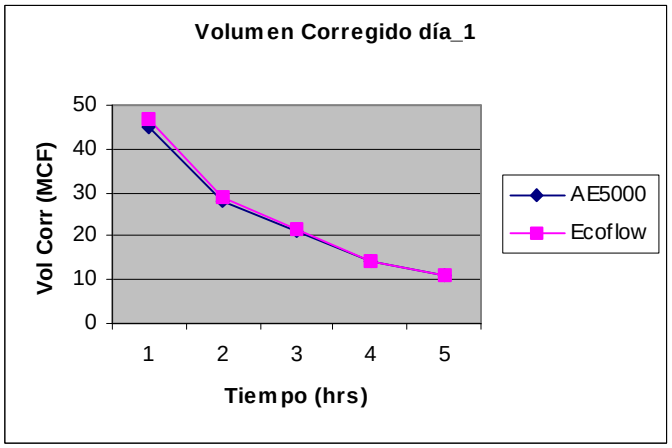


Figura 41. Volumen corregido día_1

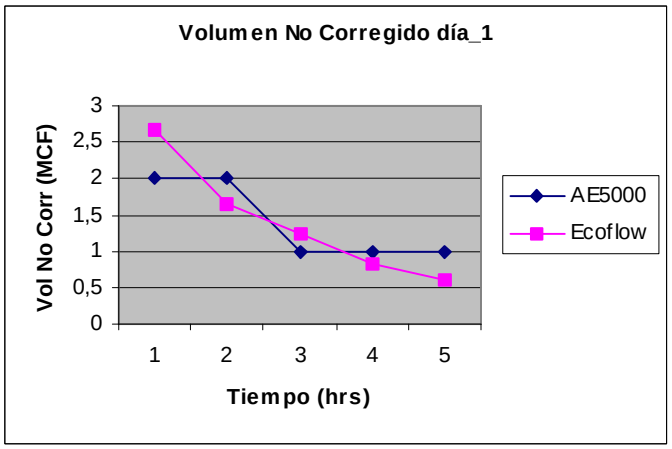


Figura 42. Volumen no corregido día_1

	AE5000		ECOFLOW	
Día 2 Time	CorVol MCF	UnCor MCF	CorVol MCF	UnCor MCF
12:00AM	10	0	10,79	0,61
01:00AM	12	1	10,79	0,61
02:00AM	9	0	10,80	0,61
04:00AM	15	1	16,23	0,92
05:00AM	51	3	52,59	2,98
06:00AM	67	4	69,00	3,90
07:00AM	74	4	74,51	4,21
08:00AM	82	4	85,29	4,82
09:00AM	106	7	107,99	6,15
10:00AM	143	8	145,45	8,39
12:00PM	83	4	83,31	4,81
01:00PM	44	3	43,62	2,51
02:00PM	34	2	36,66	2,11
03:00PM	30	2	33,08	1,91
04:04PM	39	2	39,97	2,31
05:00PM	47	3	49,06	2,81
06:00PM	60	3	59,82	3,41
07:00PM	45	3	43,90	2,51
08:00PM	30	1	28,15	1,60
09:00PM	22	2	21,12	1,20
10:00PM	16	0	15,82	0,90
11:00PM	12	1	12,30	0,70

Tabla 18. Volumen día_2.

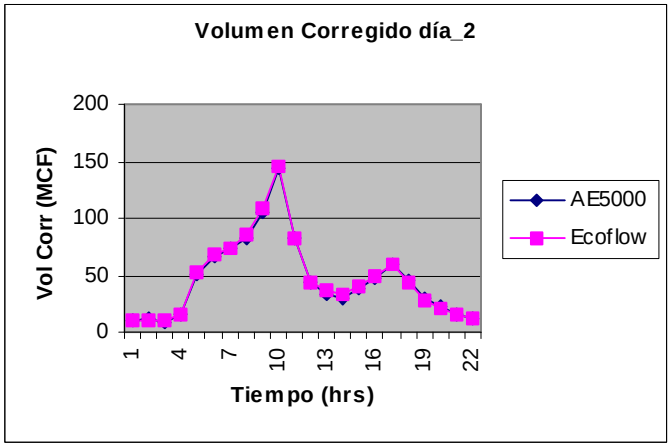


Figura 43. Volumen corregido día_2.

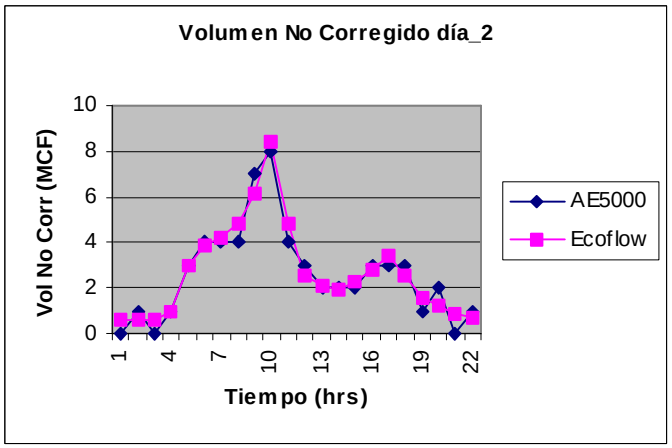


Figura 44. Volumen no corregido día_2.

	AE5000		ECOFLOW	
Día 3 Time	CorVol MCF	UnCor MCF	CorVol MCF	UnCor MCF
12:00AM	11	1	10,55	0,60
01:00AM	10	0	10,55	0,60
02:00AM	11	1	10,54	0,60
03:00AM	10	1	10,55	0,60
04:00AM	14	0	15,83	0,90
05:00AM	46	3	49,47	2,81
06:00AM	67	4	69,09	3,91
07:00AM	75	4	76,10	4,31
08:00AM	82	5	82,85	4,71
09:00AM	104	6	110,30	6,31
10:00AM	143	8	145,10	8,38
11:00AM	148	9	143,26	8,29
12:00PM	88	5	82,27	4,71
01:00PM	45	3	42,01	2,41
02:00PM	36	2	36,70	2,11
03:00PM	37	2	36,68	2,11
04:00PM	41	2	43,64	2,51
05:00PM	52	3	54,46	3,11
06:00PM	60	4	59,94	3,41
07:00PM	45	2	44,00	2,51
08:00PM	30	2	28,22	1,60
09:00PM	23	1	22,94	1,30
10:00PM	15	1	14,11	0,80
11:00PM	13	1	12,33	0,70

Tabla 19. Volumen día_3.

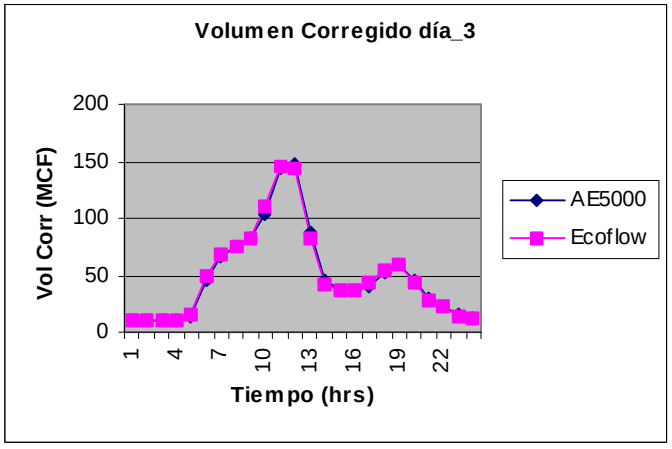


Figura 45. Volumen corregido día_3.

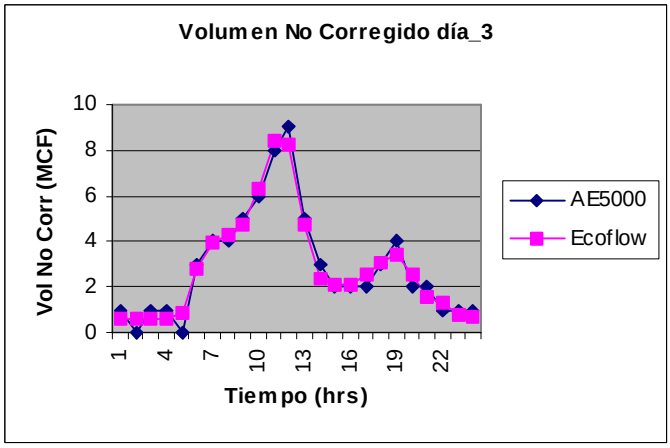


Figura 46. Volumen no corregido día_3.

	AE5000		ECOFLOW	
Día 4 Time	CorVol MCF	UnCor MCF	CorVol MCF	UnCor MCF
12:00AM	10	0	10,56	0,60
01:00AM	11	1	10,55	0,60
02:00AM	10	1	10,55	0,60
03:00AM	11	0	10,55	0,60
04:00AM	15	1	21,12	1,20
05:00AM	53	3	53,09	3,01
06:00AM	68	4	70,82	4,01
08:00PM	31	1	31,66	1,81
09:00PM	24	2	22,92	1,30
10:00PM	18	1	17,63	1,00
11:00PM	14	1	14,08	0,80

Tabla 20. Volumen día_4.

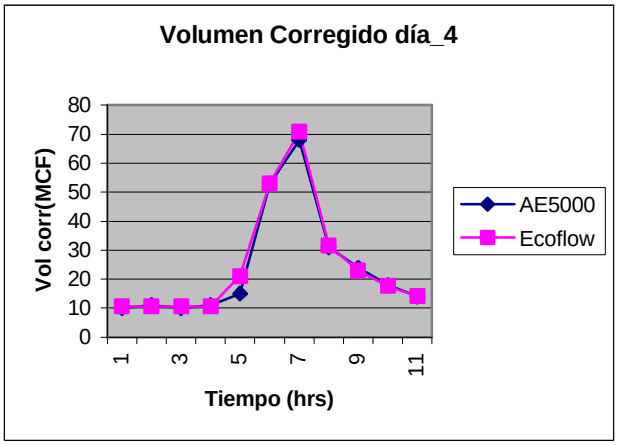


Figura 47. Volumen corregido día_4.

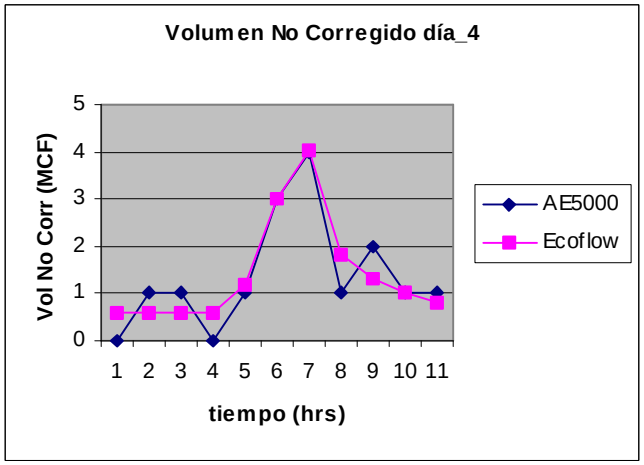


Figura 48. Volumen no corregido día_4.

	AE5000		ECOFLOW	
Día 1 Time	CorVol MCF	UnCor MCF	CorVol MCF	UnCor MCF
12:00AM	10	0	12,33	0,70
01:00AM	11	1	10,58	0,60
02:00AM	10	0	10,58	0,60
03:00AM	11	1	10,58	0,60
04:00AM	12	1	10,57	0,60
05:00AM	33	2	32,23	2,01
06:00AM	63	3	61,73	3,51
07:00AM	74	5	75,56	4,31
08:00AM	75	4	75,58	4,31
09:00AM	83	5	84,20	4,81
10:00AM	106	6	106,13	6,11

Tabla 21. Volumen día_5.

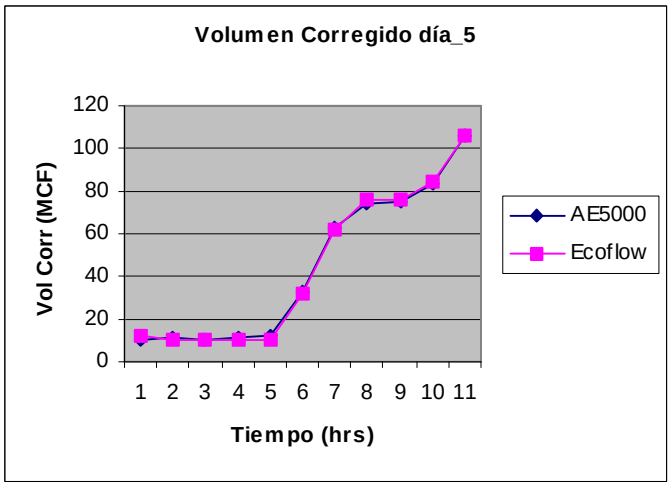


Figura 49. Volumen corregido día_5.

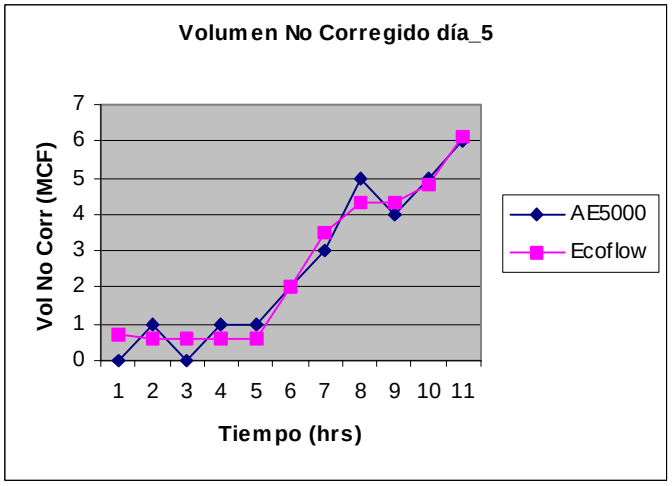


Figura 50. Volumen no corregido día_5.

Lectura Volumen Diario			
Día	AE5000	ECOFLOW	% Error
1	1197	1200,38	0,28
2	1189	1193,73	0,40
3	1206	1209,69	0,31
4	1213	1213,76	0,06

Tabla 22. Volumen corregido diario.

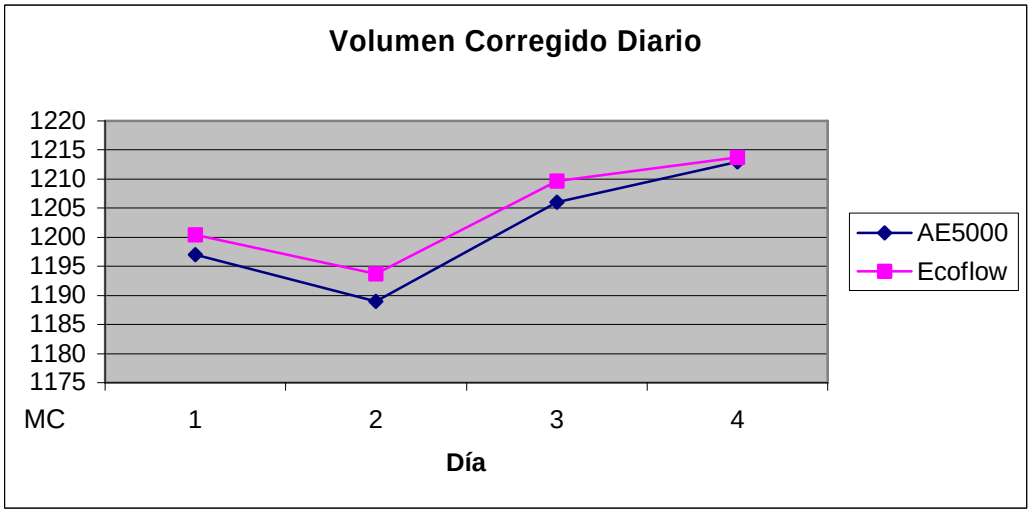


Figura 51. Volumen Corregido

8.4 OBSERVACIONES Y CONCLUSIONES DE LA PRUEBA

- Considerando el computador de flujo AE5000 como un patrón, el ECOFLOW tiene un error promedio del 0,26 % en la medición del flujo.

- El ECOFLOW ha mostrado una confiabilidad y seguridad de operación de al menos 100 horas en campo.

- La forma de promediar las variables de muestreo, presión y temperatura, son diferentes en los dos equipos. El ECOFLOW sigue la norma API 21 para el cálculo de promedios.

- La diferencia en las lecturas de presión y temperatura, también se debe a que los computadores de flujo (AE5000 y ECOFLOW) ubican las tomas de dichas variables en lugares diferentes. El espacio recorrido por el flujo entre los dos equipos, posee diversos obstáculos en el camino como codos y válvulas.

- Las gráficas de volumen corregido garantizan que el ECOFLOW realiza un cálculo real de dicho volumen. La diferencia se debe a que los dos equipos emplean diferentes métodos para el cálculo de Z. El ECOFLOW emplea el método AGA GROSS 2, mientras que el AE5000 utiliza el NX-19.

- El computador de flujo ECOFLOW suministra valores de los volúmenes medidos con mayor exactitud ya que considera hasta cuatro décimales, mientras que el AE5000 realiza aproximaciones de miles. Esta diferencia es más evidente al observar las gráficas de volumen no corregido, donde el AE5000 considera un solo valor entero.

En la siguiente tabla se realiza una comparación del computador diseñado con el computador actualmente usado:

ECOFLOW 2020T	*AE5000*
Registra Volúmenes con resolución de centésimas de pie cúbico	Registra Volúmenes con resolución de miles enteros de pie cúbico
Calcula el Factor Z usando AGA8 Gross 2	Calcula el Factor Z usando NX-19
Velocidad de muestreo de 10 milisegundos	Velocidad de muestreo de 70 milisegundos
Convertidor Análogo-Digital de 16 bits	Convertidor Análogo-Digital de 12 bits
Despliega simultáneamente las 4 variables mas importantes del proceso en el Display	Muestra solo una variable al tiempo en el Display
Fácil y rápida reparación o mantenimiento.	Demorados tiempos de espera e importación de repuestos.
No tiene salidas análogas ni digitales, pero se pueden implementar bajo pedido del cliente.	Tiene salidas análogas y digitales para control de válvulas reguladas o válvulas ON-OFF, sin embargo, no son usadas en ningún lugar.
Adecuación de Hardware y Software de acuerdo al cliente.	Estructura rígida de hardware y software que a veces es subutilizada

Tabla 23. Comparación del ECOFLOW2020 vs AE5000

9. CONCLUSIONES

- El equipo desarrollado durante el presente proyecto, es un prototipo de computador de flujo denominado ECOFLOW, el cual esta habilitado para corregir el flujo de gas medido por los medidores primarios tipo turbina; y si en un futuro se disponen de los recursos técnicos y económicos, podrá ser adaptado para otros medidores como platina de orificio, rotativo, diafragma y ultrasónico.

- El ECOFLOW esta basado en un sistema embebido (TDS2020F) importado. Además, contiene una tarjeta de acondicionamiento (diseñada y construida en un proyecto paralelo), una batería de respaldo, transductores de presión y temperatura, encapsulado, *display* y teclado.

- El lenguaje Forth demostró dar el soporte necesario para el rigor matemático exigido; es de notar que la gran batería de rutinas anexas al hardware adquirido, facilito en gran forma los cálculos requeridos. Se utilizaron los archivos para el manejo de flotantes, adquisición de señales, pantalla, teclado y manejo de variables externas y se dispuso de ellos para incluirlas en el programa principal previamente modificadas.

- El manejo de interrupciones fue vital para el desarrollo del *sofware.* Se logró disponer de la autonomía necesaria que demanda un equipo de tal índole.

- ECOFLOW emplea la norma AGA 8 GROSS 2 para el cálculo del factor de comprensibilidad. Esto es una gran innovación, puesto que los computadores instalados actualmente en Colombia utilizan el NX-19.

- La implementación de la norma AGA 8 para el computador de flujo (firmware) fue exitosa, obteniéndose un error del cero por ciento con relación al reporte AGA 8.

- El equipo fue probado en campo donde cumplió con 100 horas de trabajo, verificandose la funcionalidad y confiabilidad que le puede representar a un sistema de medición.

- Los resultados de las pruebas fueron contrastados con los arrojados por un equipo comercial (AE5000), obteniéndose una diferencia del 0,26 %. Es de resaltar que esta diferencia se debe a que los dos equipos emplean normas diferentes para el cálculo del factor Z, pero lo más

importante es que el ECOFLOW sigue perfectamente la curva de volumen corregido del AE5000.

- De las pruebas hechas en campo se deduce el buen desempeño del computador de flujo comparado con el patrón existente AE5000 .Con esto se concluye que el proyecto culminó satisfactoriamente cumpliendo con las expectativas previstas por el C.I.G. y que se ajustan a las necesidades reales de la industria del gas en Colombia.

- A través de este proyecto se ha incentivado el trabajo interdisciplinario dentro de la universidad y la cooperación interinstitucional, logrando un gran avance en la investigación y el desarrollo tecnológico.

10. RECOMENDACIONES

- Conviene estudiar la incorporación de funciones de control de válvulas de línea on-off y válvulas reguladas a partir del mismo Computador de Flujo.

- Adicionar características de comunicación avanzadas para conectar el Computador de Flujo con sistemas de supervisión y control SCADA.

- Para posibles nuevas funciones del computar de flujo; conviene analizar las rutinas de multitareas que vienen prediseñadas en el TDS2020F; para su posible utilización en manejo de control remoto y aplicaciones de red.

BIBLIOGRAFÍA

BRODIE, Leo and Forth Inc. Starthing Forth, an introduction to the Forth language and Operation System For beginners an Professionals , Prentice Hall, Englewood Cliffs New Jersey , Segunda Edición 1987.

CAMARGO, Faustino. STACH, Darcy. Gas Measurement. Bucaramanga: Seminario de Medición de Gas, CentrOriente S.A., 1999. p. 2_1-2_16.

CONKLIN Edward. RATHER Elizabeth: Forth Programmer`s Handbook,
DANIEL MEASUREMENT AND CONTROL. The Daniel Ultra Tap Gas Flow Meter. Houston : Western Hemisphere Operations, 1997. p.3.

DeBUSK FRED N. Application of Flow Computers for Gas Measurement and Control : Daniel Industries Inc., Salt Lake City, Utah.
Forth Inc. 1998

GALVIS BARRERA, Hernando. Medición y Regulación. Bucaramanga: Especialización en Ingeniería de Gas, Universidad Industrial de Santander, 2000. p.19-38.

GONZÁLEZ VÁSQUEZ, Ricardo. Diseño e Implementación del Hardware y Firmware para un Computador de Flujo tipo turbina usando la norma AGA 7. Bucaramanga : Especialización en Ingeniería de Gas, Universidad Industrial de Santander, 2000. p. 1-33.

RONCANCIO RODRÍGUEZ, Rafael. Diseño e Implementación del Hardware y Software de presentación para el Computador de Flujo de un Medidor de Gas tipo turbina. Bucaramanga : Especialización en Ingeniería de Gas, Universidad Industrial de Santander, 2000. p. 1-35.

DOCUMENTOS DE INTERNET:

- *Flow Computers Fundamentals*
  Jim Griffeth, Bristol Babcock Inc., 2000, Houston, Texas.
  www.pbmsonline.org/papers/gas/ flowcomputersfundamentals.htm
- *Flow Computers*
  OMNI New Letters, Vol 1 Num. I, II y III
  www.omniflow.com/about/recent/newsltr1.html

- *Trends in Electronic Flow Computers*
  www.sensorsmag.com/articles/1099/54/main.shtml

- *A Look at Flow Computers and Software*
  Gas Utility Manager
  www.gasindustries.com/articles/prod100.htm

- *Remote Measure & Control Application Guide*
  www.thomasregister.com/olc/fisherregulators/controlapplication.htm

- TDS2020F Technical Manual Contents
  www.triangledigital.com

ANEXO 1

CODIGO FUENTE EN MATLAB
AGA8 GROSS-2

Las siguientes funciones: CONSTANT, LIMITS, GROSSCOMP y VIRIAL2 simplemente se usan para definir variables globales en grupos separados, los cuales serán invocados posteriormente por las demás funciones.

La rutina llamada PRINCIPAL es el programa principal que invoca las demás funciones explicadas en el capítulo 8 y trabaja con una temperatura y presión hipotética así como una composición de gas propuesta. Las tres variables de salida de todo este conjunto de funciones son la densidad (D), el factor de compresibilidad (Z) y el factor de compresibilidad (Fpv)

CONSTANT.m

```
global PROG RGAS MWX MW
```

LIMITS.m

```
global DHIGH PLOW PHIGH TLOW THIGH
```

GROSSCOMP.m

```
global XX
```

VIRIAL2.m

```
global BB B0 B1 B2 C0 C1 C2 TOLD BBMIX CCMIX
```

PRINCIPAL.m

```
format long ;

LIMITS;
CONSTANTS;
GROSSCOMP;
VIRIAL2;

ICOMP = 2;
METHOD = 2 ;
```

```
TDATA = 65.1;
PDATA = 752;

XK(:,1) = [0.581078; 60; 14.73; 0.002595; 0.005956;  0; 0;];

XK(:,2) = [0.608657; 60; 14.73; 0.031284; 0.004676; 0; 0; ];

%%para pasar de  f a k = 5/9 * (valor + 459.67)
%%para pasar de psia mpa valor * 0.006894757

   for J = 1 : 7
         XD(J) = XK(J,ICOMP);
         XDC(J) = XK(J,ICOMP);
   end

     PARAMGS;
     TBDR = 5/9 * (XD(2) + 459.67);
     PB = XD(3) * 0.006894757;
     SPECG = XD(1);
     HS = 0;
     X(1) = 0;
     for I = 2 : 5
        X(I) = XD(I+2);
     end

     X(1) = 1-X(2) - X(3) - X(4) - X(5);

     [ZB,DB,ERRNUM] =
CHARGS(METHOD,HS,SPECG,X,TBDR,TBDR,PB,TBDR,PB);
     TK =(5/9) * (TDATA + 459.67);
     PMP = PDATA * 0.006894757;

     D = DGROSS (PMP,TK);
     Z = ZGROSS (D, TK);
     FPV = sqrt (ZB/Z);
     Df = D* MWX  / 16.01846; % AQUI VA LA TRANSFORMACION DE
UNIDADES

     disp ('AQUI VAN LAS SALIDAS D Df Z FPV')
     disp(D)
     disp(Df)
     disp(Z)
     disp(FPV)
```

PARAMGS.m

```
CONSTANTS;
VIRIAL2;
```

```
RGAS = 8.31451e-3;
MW(2) = 28.01350;
MW(3) = 44.010;
MW(4) = 2.0159;
MW(5) = 28.01;

B0(2,2) = -0.144600;
B1(2,2) = 0.740910e-3;
B2(2,2) = -0.911950e-6;
B0(3,3) = -0.868340;
B1(3,3) = 0.403760e-2;
B2(3,3) = -0.516570e-5;
B0(4,4) = -0.110596-2;
B1(4,4) = 0.813385e-4;
B2(4,4) = -0.987220e-7;
B0(5,5) = -0.130820;
B1(5,5) = 0.602540e-3;
B2(5,5) = -0.644300e-6;
B0(2,3) = -0.339693;
B1(2,3) = 0.161176e-2;
B2(2,3) = -0.204429e-5;
B0(1,4) = -0.521280e-1;
B1(1,4) = 0.271570e-3;
B2(1,4) = -0.250000e-6;
B0(1,5) = -0.687290e-1;
B1(1,5) = -0.239381e-5;
B2(1,5) = 0.518195e-5;
B0(2,4) = 0.012;

BB(1,1) = -0.425468;
BB(2,1) = 0.877118e-3;
BB(3,1) = -0.824747e-6;
BB(1,2) = 0.286500e-2;
BB(2,2) = -0.556281e-5;
BB(3,2) = 0.431436e-8;
BB(1,3) = -0.462073e-5;
BB(2,3) = 0.881510e-8;
BB(3,3) = -0.608319e-11;

CO(2,2) = 0.784980e-2;
C1(2,2) = -0.398950e-4;
C2(2,2) = 0.611870e-7;
C0(3,3) = 0.205130e-2;
C1(3,3) = 0.348880e-4;
C2(3,3) = -0.837030e-7;
C0(2,3) = 0.552066e-2;
C1(2,3) = -0.168609e-4;
C2(2,3) = 0.157169e-7;
C0(3,2) = 0.358783e-2;
C1(3,2) = 0.806674e-5;
C2(3,2) = -0.325798e-7;
C0(4,4) = 0.104711e-2;
```

```
C1(4,4) = -0.364887e-5;
C2(4,4) = 0.467095e-8;
C0(1,5) = 0.736748e-2;
C1(1,5) = -0.276578e-4;
C2(1,5) = 0.343051e-7;
```

CHARGS.m

```
function [ZB,DB,ERRNUM] = CHARGS(METHOD,HV,GR,X,TH,TD,PD,TGR,PGR)

VIRIAL2;
GROSSCOMP;
LIMITS;
CONSTANTS;

TOLD = 0 ;
ERRNUM = 0;
VIR = -0.12527 + 5.91e-4*TGR - 6.62e-7*TGR^2;
DOAIR = 28.96256/(RGAS*TGR/PGR + VIR);

G1 = -2.709328;
G2 = 0.021062199;
HTV4 = 285.83;
HTV5 = 282.98;

if (METHOD == 2)
   Z0 = 1;

   MR = GR*Z0*RGAS*TGR/PGR*DOAIR;
   MW(1) = (MR - X(2)*MW(2) - X(3)*MW(3) - X(4)*MW(4) -
X(5)*MW(5))/X(1);
   HCH = (MW(1) - G1)/G2;
   BCH  = BB(1,1) + TGR*(BB(1,2)+BB(1,3)*TGR) +
(BB(2,1)+TGR*(BB(2,2)+BB(2,3)*TGR))*HCH ...
      + (BB(3,1)+TGR*(BB(3,2)+BB(3,3)*TGR))*HCH^2 ;

   % CALL VIRGS(TGR,X,BMIX,TEMP,BCH,1,ERRNUM);
   [BMIX,TEMP,ERRNUM]=VIRGS1(TGR,X,BCH,1);
   Z0NEW = 1 + BMIX*PGR/RGAS/TGR;
   while (abs(Z0/Z0NEW - 1) > 0.5e-10)
      Z0 = Z0NEW;

   MR = GR*Z0*RGAS*TGR/PGR*DOAIR;
   MW(1) = (MR - X(2)*MW(2) - X(3)*MW(3) - X(4)*MW(4) -
X(5)*MW(5))/X(1);
   HCH = (MW(1) - G1)/G2;
   BCH  = BB(1,1) + TGR*(BB(1,2)+BB(1,3)*TGR) +
(BB(2,1)+TGR*(BB(2,2)+BB(2,3)*TGR))*HCH ...
      + (BB(3,1)+TGR*(BB(3,2)+BB(3,3)*TGR))*HCH^2 ;
```

```
   % CALL VIRGS1(TGR,X,BMIX,TEMP,BCH,1,ERRNUM);
   [BMIX,TEMP,ERRNUM]=VIRGS1(TGR,X,BCH,1);
   Z0NEW = 1 + BMIX*PGR/RGAS/TGR;
   disp('imprima zonew  ..... DA UN VALOR MEDIANAMENTE
ACEPTABLE');
   disp ( Z0NEW ) ;
   end
end

if (ERRNUM ~= 0)
   return
end

B0(1,1) = BB(1,1) + HCH*(BB(2,1)+BB(3,1)*HCH);
B1(1,1) = BB(1,2) + HCH*(BB(2,2)+BB(3,2)*HCH);
B2(1,1) = BB(1,3) + HCH*(BB(2,3)+BB(3,3)*HCH);

C0(1,1) = -0.302488 + HCH*(0.646422e-3 - 0.332805e-6*HCH );
C1(1,1) = 0.195861e-2 + HCH*(-0.422876e-5 + 0.2231605e-8*HCH );
C2(1,1) = -0.316302e-5 + HCH*(0.688157e-8 - 0.367713e-11*HCH );

XX(1) = X(1);
XX(2) = X(2);
XX(3) = X(3);
XX(4) = X(4);
XX(5) = X(5);

TLOW = 263;
THIGH = 338;
PLOW = 0.5e-9;
PHIGH = 12;
DHIGH = 8 ;

DB = DGROSS(PGR,TGR);
MWX = GR*DOAIR/DB;     %
DB = 0;

TB = (60 + 459.67)/1.8;
PB = 14.73*6894.757/1000000;
DB = DGROSS(PB,TB);

ZB = ZGROSS(DB,TB);

disp('ESTOS SON TEMPERATURA , PRESION BASE, ');
disp(TB);
disp(PB);
disp('estos son los valores base DB ZB CHARGS');
disp(DB);
disp(ZB);
disp('estos son los valores base fin');
```

VIRGS.m

```
function[BMIX,CMIX,ERRNUM] = VIRGS(T,X,OPT)
VIRIAL2;

ERRNUM = 0;

if (T == TOLD)
   BMIX = BBMIX;
   CMIX = CCMIX;
   return
end

X11 = X(1)*X(1);
X22 = X(2)*X(2);
X33 = X(3)*X(3);
X44 = X(4)*X(4);
X55 = X(5)*X(5);
if (OPT == 0 )
   BCH = B0(1,1) + T*(B1(1,1)+B2(1,1)*T);
end
BN2 = B0(2,2) + T*(B1(2,2)+B2(2,2)*T);
BCO2 = B0(3,3) + T*(B1(3,3)+B2(3,3)*T);
BH2 = B0(4,4) + T*(B1(4,4)+B2(4,4)*T);
BCO = B0(5,5) + T*(B1(5,5)+B2(5,5)*T);
if (BCO2*BCH < 0)
   %PAUSE 'VIRGS: RAIZ NEGATIVA'
   disp ('raiz negativa');
   pause(2);
   ERRNUM = 1 ,
   return
end
B12 = (0.72 + 1.875e-5*(320-T)*(320-T))*(BN2 + BCH)/2;  %%%%%%
B13 = -0.865*sqrt(BCO2*BCH);
B14 = B0(1,4) + T*(B1(1,4) + B2(1,4)*T);
B15 = B0(1,5) + T*(B1(1,5) + B2(1,5)*T);
B23 = B0(2,3) + T*(B1(2,3) + B2(2,3)*T);
B24 = B0(2,4);
BMIX = BCH*X11 + BN2*X22 + BCO2*X33 + BH2*X44 + BCO*X55 ...
   +2*B12*X(1)*X(2) + 2*B13*X(1)*X(3) ...
   +2*B14*X(1)*X(4) + 2*B15*X(1)*X(5) ...
   +2*B23*X(2)*X(3) + 2*B24*X(2)*X(4) ;

if (OPT == 1)
   disp('paso por el opt 1 de virg');
   return
end
TOLD = T;
BBMIX = BMIX;

E = 0.92 + 0.0013*(T-270);
```

```
F = 1/3;
C11 = C0(1,1)+ T*(C1(1,1)+C2(1,1)*T);
C22 = C0(2,2)+ T*(C1(2,2)+C2(2,2)*T);
C33 = C0(3,3)+ T*(C1(3,3)+C2(3,3)*T);
C44 = C0(4,4)+ T*(C1(4,4)+C2(4,4)*T);

if ((C11 < 0) | (C33 < 0))
   disp ('invalido termino en virgs');
   ERRNUM = 1 ;
   return
end

C15 = 3*(C0(1,5) + T*(C1(1,5) + C2(1,5)*T));
C23 = 3*(C0(2,3) + T*(C1(2,3) + C2(2,3)*T));
C32 = 3*(C0(3,2) + T*(C1(3,2) + C2(3,2)*T));

CMIX = C11*X11*X(1) + C22*X22*X(2) + C33*X33*X(3) + C44*X44*X(4)
...
   + C23*X22*X(3) + C32*X33*X(2) + C15*X11*X(5) ...
   + E*3*X11*X(2)*(C11*C11*C22)^F ...
   + E*3*X22*X(1)*(C11*C22*C22)^F ...
   + 0.92*3*X11*X(3)*(C11*C11*C33)^F ...
   + 0.92*3*X33*X(1)*(C11*C33*C33)^F ...
   + 1.20*3*X11*X(4)*(C11*C11*C44)^F ...
   + 1.10*6*X(1)*X(2)*X(3)*(C11*C22*C33)^F ;
CCMIX = CMIX;
```

VIRGS1.m

```
function[BMIX,CMIX,ERRNUM] = VIRGS1(T,X,BCH,OPT)

ERRNUM = 0;
VIRIAL2;
if (T == TOLD)
   BMIX = BBMIX;
   CMIX = CCMIX;
   return
end

X11 = X(1)*X(1);
X22 = X(2)*X(2);
X33 = X(3)*X(3);
X44 = X(4)*X(4);
X55 = X(5)*X(5);
if (OPT == 0 )
   BCH = B0(1,1) + T*(B1(1,1)+B2(1,1)*T);
end
BN2 = B0(2,2) + T*(B1(2,2)+B2(2,2)*T);
BCO2 = B0(3,3) + T*(B1(3,3)+B2(3,3)*T);
BH2 = B0(4,4) + T*(B1(4,4)+B2(4,4)*T);
```

```
BCO = B0(5,5) + T*(B1(5,5)+B2(5,5)*T);

if (BCO2*BCH < 0)
  % PAUSE 'VIRGS: RAIZ NEGATIVA'
   disp ('raiz negativa');
   pause(2);
   ERRNUM = 1 ,
   return
end
B12 = (0.72 + 1.875e-5*(320-T)*(320-T))*(BN2 + BCH)/2;  %%%%%%
B13 = -0.865*sqrt(BCO2*BCH);
B14 = B0(1,4) + T*(B1(1,4) + B2(1,4)*T);
B15 = B0(1,5) + T*(B1(1,5) + B2(1,5)*T);
B23 = B0(2,3) + T*(B1(2,3) + B2(2,3)*T);
B24 = B0(2,4);
BMIX = BCH*X11 + BN2*X22 + BCO2*X33 + BH2*X44 + BCO*X55 ...
   +2*B12*X(1)*X(2) + 2*B13*X(1)*X(3) ...
   +2*B14*X(1)*X(4) + 2*B15*X(1)*X(5) ...
   +2*B23*X(2)*X(3) + 2*B24*X(2)*X(4) ;

TOLD = T;
BBMIX = BMIX;

E = 0.92 + 0.0013*(T-270);
F = 1/3;
C11 = C0(1,1)+ T*(C1(1,1)+C2(1,1)*T);
C22 = C0(2,2)+ T*(C1(2,2)+C2(2,2)*T);
C33 = C0(3,3)+ T*(C1(3,3)+C2(3,3)*T);
C44 = C0(4,4)+ T*(C1(4,4)+C2(4,4)*T);

if ((C11 < 0) | (C33 < 0))
   disp ('invalido termino en virgs');
   ERRNUM = 1 ;
   return
end

C15 = 3*(C0(1,5) + T*(C1(1,5) + C2(1,5)*T));
C23 = 3*(C0(2,3) + T*(C1(2,3) + C2(2,3)*T));
C32 = 3*(C0(3,2) + T*(C1(3,2) + C2(3,2)*T));

CMIX = C11*X11*X(1) + C22*X22*X(2) + C33*X33*X(3) + C44*X44*X(4)
...
   + C23*X22*X(3) + C32*X33*X(2) + C15*X11*X(5) ...
   + E*3*X11*X(2)*(C11*C11*C22)^F ...
   + E*3*X22*X(1)*(C11*C22*C22)^F ...
   + 0.92*3*X11*X(3)*(C11*C11*C33)^F ...
   + 0.92*3*X33*X(1)*(C11*C33*C33)^F ...
   + 1.20*3*X11*X(4)*(C11*C11*C44)^F ...
   + 1.10*6*X(1)*X(2)*X(3)*(C11*C22*C33)^F ;
   CCMIX = CMIX;
```

PGROSS.m

```
function [PGROSS] = PGROSS(D,T)
CONSTANTS;
PGROSS = D*RGAS*T*ZGROSS(D,T);
```

DGROSS.m

```
function [DGROSS] = DGROSS(P,T)

LIMITS;
TOL = 1e-6;
X1 = PGROSS (0,T);
X2 = PGROSS (DHIGH,T);
F1 = PGROSS (X1,T) - P;

if (abs(F1) <= TOL )
   DGROSS = X1;
   return;
end

F2 = PGROSS (X2,T) - P;

if (abs(F2) <= TOL)
   DGROSS = X2;
   return;
end

if (F1*F2 > 0)
   disp ('RAIZ NO HALLADA');
   DGROSS = 0;
   return;
end

for IC = 1:100
   X3 = (X1*F2 - X2*F1)/(F2 - F1);
   F3 = PGROSS (X3,T) - P ;
   if (mod (IC,6) == 0)
      DGROSS = ( X1 + X2 )/2;
   else
      if (((F1-F2)*(F1-F3)*(F2-F3)) == 0 )
         return;
      end
      DGROSS = X1*F2*F3/((F1-F2)*(F1-F3)) ...
         + X2*F1*F3/((F2-F1)*(F2-F3)) ...
         + X3*F1*F2/((F3-F1)*(F3-F2)) ;
         if ((DGROSS - X1)*(DGROSS-X2)>= 0)
         DGROSS = (X1 + X2)/2;
         end
```

```
   end

   F= PGROSS(DGROSS, T) -P ;
   if ( abs(F) <= TOL)
      return
   end
   if (F*F3 < 0)
      disp ('dgross es x1')

      X1 = DGROSS;
      F1 = F;
      X2 = X3;
      F2 = F3;
   elseif (F3*F1 > 0)
         X1 = DGROSS;
         F1 = F;
   else
      disp ('dgross es x2')
         X2 = DGROSS;
         F2 = F;
   end
   conta = conta + 1 ;
   disp ('el conta de dgross es')
disp(conta);
end

disp('DGROSS NO CONVERGE');
DGROSS = 0 ;
return ;
```

ZGROSS.m

```
function [ZGROSS] = ZGROSS(D,T)
GROSSCOMP;

ZGROSS = 0;
% CALL VIRGS(T,X,BMIX,CMIX,TEMP,0,ERRNUM);
[BMIX,CMIX,ERRNUM]=VIRGS(T,XX,0);

if (ERRNUM ~= 0)
   return;
end
ZGROSS = 1 + BMIX*D + CMIX*D*D ;
```

ANEXO 2

CODIGO FUENTE DEL FIRMWARE EJECUTABLE

#EXTVAR2.TDS

```
\ Triangle Digital Services Ltd                TDS2020 ANS Utilities

\ #EXTVAR2.TDS   Ver 1.70                      by Peter Rush  12 Oct 99
\ [C] 1999 Triangle Digital Services Ltd

\ Variables in extended RAM for TDS2020

\ ------------------------------------------------------------------------
\ DESCRIPTION

\ If the address area 08800 to 0FB80 is used for extending the Forth
\ dictionary from 16K to 45K bytes, or when using the Alternative Forth
\ ROM, there is not much space left for variables in the first 64K bytes
\ of memory space. This file redefines  VARIABLE  and associated words so
\ that extended memory is used. This can be a 32K, 128K or 512K byte RAM
\ in the 32-pin socket at address 80000.

\ Make link J, cut link K if using a 32K RAM and leave free the 4 socket
\ pins towards the edge of the TDS2020 when inserting the 28-pin device in
\ the 32-pin socket.

\ Hex F0000 to FFFFF is the default area reserved for variables. You
\ can edit this file to choose where variables will be placed in extended
\ memory. There are two numbers which define the variables region: the
\ constant  AREA  defines the 64k-byte page and the number put into VDP
\ below defines the start of variables within that page. For example PCMCIA
\ hard disks driven by file #DOSHD.TDS and/or #CARDHD.TDS use all extended
\ memory up to FE200. All the rest is free so you could edit this file to
\ read as follows:
\    $F CONSTANT AREA
\    $E200 VDP !

\ The order of the definitions in this file is from most common to least
\ used words. This is so you can easily edit out those you do not need.

\ ------------------------------------------------------------------------
\ INSTRUCTIONS FOR USE
```

```
\ Use  !!  @@  +!!  in place of  !  @  +!  when accessing defined
\ variables, but the old  !  etc will still be needed for getting at system
\ user variables etc.

\ Examples:
\   VARIABLE FRED    \ define new variable in extended RAM
\   1234 FRED !!     \ store a number in it
\   FRED @@          \ retrieve value of variable
\   2 FRED +!!       \ add 2 to the variable
\   STATE @          \ get value of a variable in the base RAM

\ To create arrays the word  ALLOT  does not need redefinition. It will
\ function in the same manner, ie:  RAM n ALLOT ROM  will allot a further n
\ bytes to the last defined variable by incrementing VDP as usual.
\ Similarly, it is permissible to use the alternative form  n VDP +!  which
\ will also allocate a further n bytes to the last defined variable. Words
\  C@@  and  C!!  are provided for indexing into arrays. For example:

\   VARIABLE XYZ     \ allocate 2 bytes at start of an array
\   98 ALLOT         \ allocate further 98 bytes to this array
\   34  XYZ 9 +  C!! \ store 34 into   the ninth element of the array
\   XYZ 9 +  C@@ .   \ fetch & display the ninth element of the array

\ Variables that are to be created using  VALUE  after this file has been
\ compiled must be defined after the file #EXTVALU.TDS has also been
\ compiled since  VDP  is now pointing to locations in extended memory, not
\ in Page 0 base RAM. Remove the code in #EXTVALU.TDS which alters the value
\ of VDP if that file is loaded after this one.

\ When using assembler to access variables defined with  VARIABLE  after
\ this file has been compiled, the Data Page Register must give the correct
\ page, which is the constant  AREA . For example to access variable ZZZ :

\             B AREA ##      DPR   LDC,
\               ZZZ ))       R2    MOVI,

\ Before the end of the the same assembler word, the Data Page Register
\ should be restored to page 0:

\             B 0 ##         DPR   LDC,

\ Variables created with  VARIABLE  prior to compilation of this file will
\ refer to base (page 0) RAM not above address hex FFFF, and can still be
\ accessed in assembler like this:

\               YYY ))       R2    MOVI,
```

```
 DECIMAL
\ ---------------------------------------------------------------------
\ EXTENDED MEMORY VARIABLES

$F CONSTANT AREA        \ 64K area for variables. $F defines F0000-FFFFF
0 VDP !                 \ start of applications variables within chosen area
\ $8000 VDP !           \ alternative when RAM chip is only 32K F8000-FFFFF
\ $E200 VDP !           \ alternative when using unmodified hard disk
                        \  support files #DOSHD.TDS or #CARDHD.TDS.
$FFFF $1C +ORIGIN !ODD   \ new end of RAM space

: EVAR  (  --  )     \ Create 16-bit variable in extended RAM
   CREATE  VDP @  ,   2 VDP +!  DOES> @ ;
: !!  ( n addr --  )     \ Store n   to variable in extended RAM
   AREA E! ;
: @@  ( addr -- n )      \ Fetch n from variable in extended RAM
   AREA E@ ;
: +!! ( n addr --  )     \ Add n to variable in extended RAM
   DUP @@ ROT + SWAP !! ;

: C!!  ( char addr --  ) \ Store byte c   to variable in extended RAM
   AREA EC! ;
: C@@  ( addr -- char )  \ Fetch byte c from variable in extended RAM
   AREA EC@ ;
: C+!! ( n addr --  )    \ Add n to byte address in extended RAM
    DUP C@@ ROT + SWAP C!! ;

: 2EVAR (  --  )     \ Create 32-bit variable in extended RAM
   CREATE  VDP @  ,   4 VDP +!  DOES> @ ;
: 2@@ ( addr -- d )      \ Fetch d from variable in extended RAM
   DUP >R 2+ AREA E@ R> AREA E@ ;
: 2!! ( d addr --  )     \ Store d   to variable in extended RAM
   DUP >R AREA E! R> 2+ AREA E! ;

: FEVAR (  --  )     \ Create 48-bit variable in extended RAM
   CREATE  VDP @  ,   6 VDP +!  DOES> @ ;
: F@@ ( addr -- t )      \ Fetch t from variable in extended RAM
   DUP >R 2+ 2+ AREA E@ R> 2@@ ;
: F!! ( t addr --  )     \ Store t   to variable in extended RAM
   DUP >R 2!! R> 2+ 2+ AREA E! ;
```

#7825P-A.TDS

```
\ #7825PAR.TDS  Ver 1.10                    by Peter Rush  25 May 02
\ [C] 2002 Triangle Digital Services Ltd

\ 16-bit 4-channel Analog to Digital ADS7825 parallel interface to TDS2020F.

\ ------------------------------------------------------------------------
\ DESCRIPTION

\ This file implements a fast parallel connection to the ADS7825 chip. See
\ file #7825SER.TDS for an SPI-bus serial interface, slower but with
\ fewer interconnections.

\ A TI/Burr-Brown ADS7825 chip provides a 4-channel 16-bit low power A to D
\ converter. It operates from a single +5V power supply that can be taken
\ from the TDS2020F. Signal inputs are +10 to -10V and the reference is
\ generated internally making this an easy chip to use. You should obtain
\ the data sheet before and check that the device meets your requirements in
\ all details before using this software.

\ The output is twos-complement, ranging from +32767 TO -32768 for
\ +10 to -10V input.

\ ------------------------------------------------------------------------
\ PIN CONNECTIONS

\ Connections assumed follow. The control lines are easily moved to other
\ TDS2020F port bits just by changing the constants in the software.

\  TDS2020F  Port name      ADS7825              Function
\
\    z23    AGND          1                AGND analog ground
\                         2                AIN0
\                         3                AIN1
\                         4                AIN2
\                         5                AIN3
\                         6   2.2uF to AGND     REF Cap
\                         7   2.2uF to AGND     REF OUT
\    z23    AGND          8                AGND analog ground
\     c4    Port 7 bit 7    9              D7 parallel data out
\     c5    Port 7 bit 6   10              D6 parallel data out
\     c6    Port 7 bit 5   11              D5 parallel data out
\     c7    Port 7 bit 4   12              D4 parallel data out
\     c8    Port 7 bit 3   13              D3 parallel data out
\    c32    GND            14              DGND digital ground
\     c9    Port 7 bit 2   15              D2 parallel data out
\    c10    Port 7 bit 1   16              D1 parallel data out
\    c11    Port 7 bit 0   17              D0 parallel data out
```

```
\    z20   Port 9 bit 3   18              Channel address input A1
\    z21   Port 9 bit 2   19              Channel address input A0
\                  20  to VS1 +5V supply   PAR/SER*
\    c12   Port 9 bit 7   21              BYTE
\    z19   Port 9 bit 4   22              R/C* Read/Convert
\                  23  to AGND            CS* Chip Select
\    c31   Port 1 bit 0   24              BUSY* low=converting
\                  25  to AGND            CONTC
\    c29   Port 1 bit 3   26              PWRD  power down
\    z18   +5VOUT        27  100nF+10uF to AGND  VS2 +5V supply
\                       feed +5V via 1000uH inductor
\    z18   +5VOUT        28  see above          VS1 +5V supply

\ ------------------------------------------------------------------------
\ PERFORMANCE

\ Timing of  A-D16  which selects a channel then does a conversion:
\     Using TDS2020DV piggyback:                91us
\     Using Forth in the H8/532 microprocessor PROM:  82us
\     Using this code inside H8/532 processor:       65us

\ Timing of  16CONVERT  which does a conversion without changing channel:
\     Using TDS2020DV piggyback:                21us
\     Using Forth in the H8/532 microprocessor PROM:  19us
\     Using this code inside H8/532 processor:       11us

\ Note that the data sheet limits the repeat conversion time to 25us, do not
\ call  16CONVERT  more often than this. The result of  16CONVERT  is from
\ the previous conversion. This interleaving saves time, the previous result
\ is read while a new conversion is in progress. You can use A-D16 without
\ concern about the interleaving, it always returns at once the value of the
\ channel selected.

\ 16CONVERT can be used as a Forth word as it stands. However the code in it
\ is a good candidate for fast interrupt-based data logging. For example see
\ the file #DOSDUAL.TDS.

 DECIMAL
\ ------------------------------------------------------------------------
\ CONSTANTS

$FF82 CONSTANT 1PORT \ Port 1 data register
$BD   CONSTANT 9DDRL \ Port 9 data direction register literal value
$FFFE CONSTANT 9DDR  \ Port 9 data direction register
$FFFF CONSTANT 9PORT \ Port 9 data register
$FF8E CONSTANT 7PORT \ Port 7 data register
$81E0 CONSTANT APORT \ Port A data register
```

```
7 CONSTANT BYTE      \ Port 9 bit 7 to ADS7825 BYTE input
4 CONSTANT R/C*      \ Port 9 bit 4 to ADS7825 R/C* input
3 CONSTANT A1        \ Port 9 bit 3 to ADS7825 A1 input
2 CONSTANT A0        \ Port 9 bit 2 to ADS7825 A0 input
0 CONSTANT BUSY*     \ Port 1 bit 0 to ADS7825 BUSY* output

\ ---------------------------------------------------------------------
\ WORDS FOR USE IN APPLICATIONS

: ADS7825-INIT ( -- ) \ Initialise connections to ADS7825
   9DDRL 9DDR C!      \ port 9 bits 2,3,4,7 to output
   9PORT BYTE  ZERO   \ BYTE input=0
   9PORT R/C*  ONE    \ R/C* input=1
   9PORT A1    ZERO   \ A1   input=0
   9PORT A0    ZERO ; \ A0   input=0

CODE CHANNEL ( channel -- ) \ Select a channel
   @R7+              R3    MOVI,  \ get channel number
 B R3                    SHLR,  \ bit 0 to carry flag
   CS IF,   B 9PORT )) A0 ## BSETI, \ set   A0 input to ADS7825
      ELSE, B 9PORT )) A0 ## BCLRI, \ clear A0
      THEN,
 B R3                    SHLR,  \ bit 1 to carry flag
   CS IF,   B 9PORT )) A1 ## BSETI, \ set   A1 input to ADS7825
      ELSE, B 9PORT )) A1 ## BCLRI, \ clear A1
      THEN,
 B 9PORT ))        R/C*  ## BCLRI, \ start conversion
 B 9PORT ))        R/C*  ## BSETI, \ terminate conversion pulse
 END-CODE

CODE 16CONVERT (  -- n ) \ 16-bit A-D conversion, n=previous conversion.
                  \ Must not be called more often than every 25us
   R3                  CLR,   \ ensure top byte is clear
 B 9PORT ))        BYTE ##  BCLRI, \ enable high byte output
 BEGIN,                     \ wait for completion of conversion
    B 1PORT ))     BUSY* ## BTSTI, \ test BUSY* line
 EQ N UNTIL,                \ loop if BUSY* still logic 0
 B APORT ))         R3      MOVFPE, \ read high byte of previous data
 B 9PORT ))        BYTE ##  BSETI, \ enable low byte output
 B R3                    SWAP,  \ move to top of R3
 B APORT ))         R2      MOVFPE, \ read low byte of previous data
 B R2             R3      OR,    \ merge in low byte of previous data
 B 9PORT ))        R/C*  ## BCLRI, \ start new conversioN
 B 9PORT ))        R/C*  ## BSETI, \ terminate conversion pulse
   NEXT 2-               JMP,   \ push R3 and return to Forth
```

```
: A-D16  ( channel -- n  ) \ Convert  channel = 0 to 3  to digital,
                  \ value  n = 0 to 65535
   CHANNEL 16CONVERT DROP          \ gives last channel result, so discard
   16CONVERT ;                     \ get conversion from the new channel

\ ---------------------------------------------------------------------
```

FPA.TDS

```
 DECIMAL
\ ---------------------------------------------------------------------
\ VARIABLES

VARIABLE  EXP   VARIABLE SGN    VARIABLE FLG    VARIABLE EXP1
VARIABLE  *SN   VARIABLE /SN    VARIABLE +SN1   VARIABLE +SN2
0 VALUE HI1
0 VALUE HI2
0 VALUE LO1
0 VALUE LO2

\ ---------------------------------------------------------------------
\ PRIMITIVES

CODE <-S ( d1 carry-in-flag -- d2 carry-out-flag ) \ Double shift left
 B 1 @R7                  ROTXR,  \ carry in
   4 @R7                 ROTXL,
   2 @R7                 ROTXL,
 B 1 @R7                  ROTXL,  \ carry out
 END-CODE

CODE S-> ( d1 carry-in-flag -- d2 carry-out-flag ) \ Double shift right
 B 1 @R7                  ROTXR,  \ carry in
   2 @R7                 ROTXR,
   4 @R7                 ROTXR,
 B 1 @R7                  ROTXL,  \ carry out
 END-CODE

CODE DNORM ( r1 -- r2 ) \ Normalise d-word
   2 @R7          R3     MOVI,
   4 @R7          R3     OR,
   EQ N IF,
       B R4             CLR,    \ in case +ve
         2 @R7           TST,
```

```
        MI IF,  B R4         DEC,    \ remember -ve
                R3         CLR,    \ ready to negate
                4 @R7       NEG,    \ negate ls word
                2 @R7 R3    SUBX,   \ negate ms word
                2 @R7 R3    MOVO,   \ complete negation
          THEN,                     \ now have abs value
        BEGIN,                      \ shift double no. left until -ve
          0 @R7             DEC,    \ decrement exponent
         B 0 ##         CCR   LDC,    \ clear carry flag
          4 @R7             ROTXL,
          2 @R7             ROTXL,
        MI UNTIL,
          0 @R7             INC,    \ increment exponent
         B 0 ##         CCR   LDC,    \ clear carry flag
          2 @R7             ROTXR,  \ one shift right
          4 @R7             ROTXR,
          B R4             TST,    \ if sign negative, restore it
        MI IF,    R3         CLR,    \ ready to negate
                4 @R7       NEG,    \ negate ls word
                2 @R7 R3    SUBX,   \ negate ms word
                2 @R7 R3    MOVO,   \ complete negation
          THEN,                     \ sign now restored
      ELSE,   0 @R7           CLR,    \ number is zero, clr exponent
      THEN,
 END-CODE

CODE FS ( r1 -- r2 1|0 ) \ f=1 when r1 negative. r2 = abs(r1)
  R3                    CLR,    \ in case flag will be 0
  2 @R7                    TST,
  MI IF, R3                 INC,    \ to 1
       R2                CLR,    \ ready to negate
       4 @R7                NEG,    \ negate lsw
       2 @R7          R2    SUBX,   \ negate msw
       2 @R7          R2    MOVO,
    THEN,
  @-R7                R3    MOVO,
 END-CODE

: D* ( d1 d2 -- q3 ) \ q3=d1*d2
   TO HI2  TO LO2  TO HI1  TO LO1
   LO1   LO2              UM*
    0    LO2   HI1         UM* D+
        LO1   HI2         UM* D+
         0    HI1   HI2     UM* D+ ;

\ -----------------------------------------------------------------------
\ ANS FORTH FLOATING MATH FUNCTIONS
```

```
CODE F!     ( r aa -- )
   @R7+              R4      MOVI,
   @R7+              R3      MOVI,
   0 @R4             R3      MOVO,
   @R7+              R3      MOVI,
   2 @R4             R3      MOVO,
   @R7+              R3      MOVI,
   4 @R4             R3      MOVO,
 END-CODE

CODE F@     ( aa -- r )
   @R7+              R4      MOVI,
   4 @R4             R3      MOVI,
   @-R7              R3      MOVO,
   2 @R4             R3      MOVI,
   @-R7              R3      MOVO,
   0 @R4             R3      MOVI,
   @-R7              R3      MOVO,
 END-CODE

: FVARIABLE ( -- )     VARIABLE  4 VDP +! ;
: FCONSTANT ( r -- )   CREATE  DP @ F!  6 DP +!  DOES> F@ ;

CODE FDUP ( r -- r r )
   4 @R7             R2      MOVI,
   2 @R7             R4      MOVI,
   0 @R7             R3      MOVI,
   @-R7              R2      MOVO,
   @-R7              R4      MOVO,
   NEXT 2-                   JMP,    \ push R3 and finish

CODE FOVER ( r1 r2 -- r1 r2 r1 )
   10 @R7            R2      MOVI,
   8 @R7             R4      MOVI,
   6 @R7             R3      MOVI,
   @-R7              R2      MOVO,
   @-R7              R4      MOVO,
   NEXT 2-                   JMP,    \ push R3 and finish

CODE FSWAP ( r1 r2 -- r2 r1 )
   4 @R7             R2      MOVI,
   10 @R7            R3      MOVI,
   10 @R7            R2      MOVO,
   4 @R7             R3      MOVO,
   2 @R7             R2      MOVI,
   8 @R7             R3      MOVI,
```

```
  8 @R7             R2      MOVO,
  2 @R7             R3      MOVO,
  0 @R7             R2      MOVI,
  6 @R7             R3      MOVI,
  6 @R7             R2      MOVO,
  0 @R7             R3      MOVO,
 END-CODE

CODE FDROP ( r -- )
  6 ##              R7      ADD,
 END-CODE

: FNEGATE ( r -- -r ) >R DNEGATE R>  ;

: FABS ( r -- |r| )   >R DABS R>  ;

: D>F ( d -- r   Double to float)  31 DNORM  ;

: F*  ( r1 r2 -- r3 ) \ r3=r1*r2
  FS *SN !  1+ EXP !  2>R              \ save sign & exp 1
  FS *SN +!  EXP +!   2R>              \ save sign & exp 2
  D* 2DUP OR                           \ 64 bit multiply = 0 ?
  IF BEGIN
      DUP 16384 AND 0=
    WHILE  -1 EXP +! 2>R 0 <-S 2R> ROT <-S DROP
    REPEAT
    2>R  NIP 2R> ROT 0< ABS 0 D+             \ round up (flag as integer)
    DUP 0<
    IF  0 S-> DROP  1 EXP +!  THEN
    *SN @ 1 =
    IF  DNEGATE  THEN                    \ recover sign
    EXP @                                \ recover exponent
  ELSE  FDROP DROP  0 0 0
  THEN  ;

: F/  ( r1 r2 -- r3 ) \ r3=r1/r2
  FS /SN !  NEGATE EXP !               \ Negate divisor & save
  2DUP OR                              \ Check 0 - mantissa
  IF  DNEGATE 2>R  FS /SN +!  EXP +!      \ make +ve
    2>R 0 0 2R>                        \ 0 quotient to stack
    31 FLG !                           \ max no. of l. shifts
    BEGIN  2R@ D+ DUP 0<
      IF    2>R 0 <-S DROP 2R> 2R@ D-
      ELSE  2>R 1 <-S DROP 2R>
      THEN
      0 <-S DROP  -1 FLG +!   FLG @ 0=
    UNTIL 2R> 2DROP
```

```
    2DROP  EXP @ 1+ DNORM  /SN @ 1 = IF  FNEGATE  THEN
  ELSE  -42 THROW  \ ABORT" Divide by 0 error"
  THEN  ;

: F+  ( r1 r2 -- r3 ) \ r3=r1+r2
  2 PICK 2 PICK OR                          \ r2 non zero
  IF  FS +SN1 !  EXP !  2>R  2 PICK 2 PICK OR
    IF  FS +SN2 !  EXP1 !  EXP @  EXP1 @ - DUP
      IF  DUP 0> IF    0 DO  0 S-> DROP  LOOP
              ELSE  2R@ ROT ABS
                  0 DO  0 S-> DROP  LOOP
                  2R> 2DROP 2>R
              THEN
      ELSE  DROP
      THEN  2R@ +SN1 @ +SN2 @ - EXP @ EXP1 @ MAX EXP !
      IF  D-
      ELSE  D+  1 EXP +!  0 S-> 0 D+ DUP 0<
        IF  1 EXP +!  0 S-> DROP  THEN
      THEN  +SN2 @  IF  DNEGATE  THEN  EXP @ DNORM
    ELSE FDROP 2R@ +SN1 @ IF DNEGATE THEN EXP @
    THEN 2R> 2DROP
  ELSE  FDROP
  THEN ;

: F-  ( r1 r2 -- r3   r3=r1-r2)   FNEGATE  F+  ;

: F>D ( r -- d) \ float to double
  FS SGN !  DUP 0<
  IF  FDROP 0 0
  ELSE  31 SWAP - DUP 0>
    IF  0 DO  0 S-> DROP  LOOP
    ELSE  IF  -43 THROW  THEN         \ ABORT" Overflow in F>D"
    THEN
    SGN @  IF DNEGATE THEN
  THEN ;

: F0<  ( r -- flag )      DROP NIP 0<  ;
: F0=  ( r -- flag )      DROP OR 0=  ;
: F<   ( r1 r2 -- flag )  F- F0<  ;
: FMAX ( r1 r2 -- r3 ) \ r3 is greater of r1 & r2
  FOVER FOVER F<  IF  FSWAP  THEN  FDROP ;
: FMIN ( r1 r2 -- r3 ) \ r3 is lesser of  r1 & r2
  FSWAP FOVER FOVER F< 0= IF  FSWAP  THEN  FDROP ;
: FLITERAL \ Compilation: ( r --  ) Append run-time action to definition
        \ Run-time:    (  -- r ) Place r on the stack
  STATE @ IF  ROT POSTPONE LITERAL  SWAP
          POSTPONE LITERAL  POSTPONE LITERAL
```

```
        THEN ; IMMEDIATE

\ ------------------------------------------------------------------------
\ NON ANS FORTH FLOATING MATH FUNCTIONS

0       16384  1      FCONSTANT %1         \ 1.0
%1 %1 %1 F+ F/         FCONSTANT 0.5E       \ 0.5

: S>F   ( n -- r )         \ Single to float
   S>D  D>F  ;
: F>S   ( r -- n )         \ Float to single
   F>D ?DUP  IF 1+  IF  -43 THROW  THEN  THEN ;   \ ABORT" Overflow in F>S"
: F0>   ( r -- flag )      \ Flag is true if r is greater than 0.
   DROP NIP 0> ;
: F>    ( r1 r2 -- flag )  \ Flag is true if r1 is greater than r2.
   F- F0>  ;
: FINT  ( r1 -- r2 )   \ r2 is the floating point integer
                       \ value of r1 rounded towards zero.
   DUP 32 <  IF  F>D  D>F  THEN ;
: F=    ( r1 r2 -- flag )  \ Flag is true if r1 is equal to r2.
   F- F0=  ;

\ ------------------------------------------------------------------------
\ MORE ANS FORTH FLOATING MATH FUNCTIONS


 DECIMAL
MARKER FP2 \ Discards file #FP2.TDS onwards when executed
\ ------------------------------------------------------------------------
\ CONSTANTS

0       0      0      FCONSTANT %0         \ 0.0
26214   26214 -3      FCONSTANT %.1        \ 0.1
0       20480  4      FCONSTANT %10        \ 10.0
-8219  -27069 -7      FCONSTANT %D0
-15445  18920 -4      FCONSTANT %D1
-1809  -24991 -3      FCONSTANT %D2
-16097  21974 -2      FCONSTANT %D3
-30256 -31554 -2      FCONSTANT %D4
-14355  21744 -1      FCONSTANT %D5
16346  -32760 -1      FCONSTANT %D6
-7680   32767  0      FCONSTANT %D7
3066    22713  0      FCONSTANT %D8
-5035   28461 -1      FCONSTANT %D9
34 CONSTANT FXLIMIT    \ If length of fixed-point format would     - not ANS
                       \ exceed this F. uses scientific format
                       \ instead; it may be set to any value
```

```
                \ greater than PMAX but for strict ANS
                \ compliance it should not be less than 34

10 CONSTANT PMAX       \ Maximum precision - must not exceed 10    - not ANS
                \ The maximum reliable precision is 8. If precision
                \ is set to 10 the 10th digit may be a
                \ non-significant zero. If precision is set to 9 or
                \ 10 the least significant digit may not be
                \ completely accurate.

\ ------------------------------------------------------------------------
\ VARIABLES

VARIABLE FOB   8 VDP +!  \ Floating point output buffer           - not ANS
8 VALUE <PRECISION>    \ Number of significant digits for F. FE. FS.
                \                                - not ANS
0 VALUE OP-EXP         \ Decimal exponent for output             - not ANS
0 VALUE IP-EXP         \ Accumulator for implied exponent         - not ANS
VARIABLE %N

\ ------------------------------------------------------------------------
\ PRIMITIVES

CREATE TEN-POWERS         \ 10**(2**x) for x = 0 to 13           - not ANS
    0 20480     4 , , ,
    0 25600     7 , , ,
    0 20000    14 , , ,
 4096 24414    27 , , ,
58592 18189    54 , , ,
54996 20194   107 , , ,
 4005 24892   213 , , ,
 9189 18909   426 , , ,
30206 21823   851 , , ,
53335 29068  1701 , , ,
15043 25787  3402 , , ,
40367 20293  6804 , , ,
10497 25136 13607 , , ,
53481 19281 27214 , , ,

: 10**D* ( r1 d -- r2 ) \ Primitive used in REPRESENT             - not ANS
                  \ r2 = r1 * (10**d)
   2DUP D0< >R DABS                  \ sign of d to R leaving absolute ud
   9863. 2OVER D< IF -43 THROW THEN     \ pre-emptive test for overflow seed
   2>R %1 2R>                        \ for r3 which will equal 10**ud
   [ TEN-POWERS 14 6 * + ] LITERAL TEN-POWERS
   DO                              \ for each bit of ud ...
     0 S->
```

```
    IF                          \ if bit is set
      I -ROT 2>R F@ F* 2R>      \ multiply r3 by corresponding
                                \  power of 10 from table
    THEN
    2DUP D0= IF LEAVE THEN      \ leave loop if no more bits set
  6 +LOOP
  2DROP                         \ drop exhausted ud
  R>
  IF                            \ if d was negative
    F/                          \   r2 = r1 / (10**ud)
  ELSE
    F*                          \ else r2 = r1 * (10**ud)
  THEN ;

: COUNT-DIGITS ( ud -- u ) \ u is number of decimal digits in ud   - not ANS
  0 >R BEGIN
       2DUP D0= 0=
     WHILE R> 1+ >R 10 M/MOD ROT DROP
     REPEAT 2DROP R> ;

: .D  ( n --  )       \ Display n in decimal free field format    - not ANS
  BASE @ >R DECIMAL . R> BASE ! ;

: ADVANCE ( ca1 u1 -- ca2 u2 ) \ If u1 = 0 then ca2 u2 = ca1 u1
                        \ otherwise ca2 = ca1 + 1 and u2 = u1 - 1
  DUP IF 1 /STRING THEN ;           \                       - not ANS

: BLANKS? ( ca u -- flag )     \ True if string ca u is all blanks - not ANS
  TRUE -ROT OVER + SWAP
  ?DO  I C@ BL <> IF DROP FALSE LEAVE THEN  LOOP ;

: ?FERROR                                       \ - not ANS
  >R SP0 @ SP@ - 2- R> 2* - 0<
  IF  -4 THROW  THEN ; \ ABORT" Data stack underflow in ?FERROR"

: LNP1 ( r1 -- r2 ) \ r2 = ln (r1 + 1) for 0 <= r1 <= 1           - not ANS
  FDUP %D0 F*
  %D1 F+ FOVER F*  %D2 F+ FOVER F*  %D3 F+ FOVER F*
  %D4 F+ FOVER F*  %D5 F+ FOVER F*  %D6 F+ FOVER F*
  %D7 F+ F* ;

\ LOGARITHMS

: FLN  ( r1 -- r2 ) \ r2 is the natural logarithm of r1
  3 ?FERROR  2 PICK 2 PICK OR 0=  2 PICK 0< OR
  IF  -46 THROW  THEN \ ABORT" Invalid argument to LOG"
  1 SWAP 1- %N !  %1 F- LNP1
```

```
   %N @ S>F  %D8 F* F+ ;

: FLOG ( r1 -- r2 ) \ r2 is the base 10 logarithm of r1
   FLN %D9 F*  ;


\ FLOATING POINT OUTPUT

: SET-PRECISION ( u --  ) \ Sets number of significant digits to be used
                  \ for F. FE. FS.
   TO <PRECISION> ;

: PRECISION     (  -- u ) \ Returns number of significant digits used
                  \ for F. FE. FS.
   <PRECISION>  1 MAX  PMAX  MIN ;

: REPRESENT ( r ca u -- n flag1 flag2 )
  \ Places at ca a string u digits long representing the significand of r
  \ as a decimal fraction rounded to a number of significant digits which is
  \ the lesser of u and PMAX,  with the implied decimal point to the left of
  \ the first digit; n is the value of the decimal exponent; flag1
  \ represents the sign of r (true = negative); flag2 is true if r was a
  \ valid floating point number  -  always true in this implementation.
   2DUP 48 FILL                    \ fill string with "0"s
   PMAX MIN                        \ limit precision
   2>R                              \ ca u to R
   FDUP F0=
   IF                              \ if r = 0
     2R> 2DROP FDROP                \   clear stacks
     0 FALSE TRUE EXIT              \   leave parameters and exit
   THEN
   FS                              \ absolute value and sign
   2R> ROT >R 2>R                    \ sign to R beneath ca u
   FDUP FLOG F>S                     \ calculate decimal exponent
   DUP TO OP-EXP                     \ save copy
   NEGATE PMAX +                      \ negate and add PMAX
   S>D 10**D*       ( ... -- r2 )     \ r2 should represent the decimal
                              \  fraction of r "shifted" left by
                              \  PMAX digits so that it can be
                              \  converted to a double-cell integer
                              \  but because the decimal exponent
                              \  calculated by FLOG may not be
                              \  exact and the shifting by PMAX may
                              \  cause overflow some fine tuning is
                              \  needed first; r2 is multiplied or
                              \  divided by 10 to make it as large
                              \  as possible without its binary
```

```
                        \  exponent exceeding 31 while OP-EXP
                        \  is adjusted to match:
DUP 31 <
IF
  BEGIN
    FDUP %10 F* DUP 31 <
  WHILE
    FSWAP FDROP
    OP-EXP 1- TO OP-EXP
  REPEAT  FDROP
ELSE
  BEGIN
    DUP 31 >
  WHILE
    %.1 F*
    OP-EXP 1+ TO OP-EXP
  REPEAT
THEN
F>D                         \ convert to double-cell integer
2DUP COUNT-DIGITS           \ calculate number of digits
DUP PMAX - OP-EXP + TO OP-EXP       \ adjust decimal exponent
R@ 1 MAX                    \ number of digits required
- 0 MAX                     \ number of digits to be rounded off
?DUP
IF                          \ if rounding is needed
  0 >R                      \ seed for non-zero-remainder flag
  1-                        \ reduce digits by 1 less than needed
  BEGIN  ?DUP               \ ...
  WHILE
    >R
    10 M/MOD
    ROT R> R> ROT + >R          \ maintain non-zero-remainder flag
    1-
  REPEAT                    \ ...
  10 M/MOD                  \ reduce by final digit
  ROT DUP 5 =
  IF                        \ check for round up required ...
    DROP R@
    IF   TRUE               \ round to nearest
    ELSE OVER 1 AND         \ if midway round to even
    THEN
  ELSE
    5 >                     \ round to nearest
  THEN                      \ ... flag = true if round up reqd.
  R> DROP                   \ drop non-zero-remainder flag from R
  IF                        \ if round up is required
    2DUP COUNT-DIGITS >R        \ number of digits to R
```

```
      1. D+                    \     add 1
      2DUP COUNT-DIGITS        \     count digits again
      R> <>
      IF                       \     if number of digits has changed
        OP-EXP 1+ TO OP-EXP    \       adjust decimal exponent
      THEN
    THEN
  THEN
  <#                           \ convert double-cell integer
                               \ to string
    BEGIN
      10 M/MOD   ROT 48 + HOLD   2DUP D0=
    UNTIL
  #>
  2R>                          \ ca u from R
  ROT MIN CMOVE                \ move string to ca
  OP-EXP                       \ fetch decimal exponent
  R>                           \ sign from R
  TRUE ;

: F0. ( r=0 --  )      \ Display f.p. zero in fixed-point format   - not ANS
   FDROP S" 0.000000000" DROP PRECISION 1+ TYPE ;

: FE. ( r --  )        \ Display r in engineering format
   FDUP F0=
   IF    F0. ." E0 " EXIT
   THEN
   FOB 3 48 FILL               \ in case PRECISION is less than 3
   FOB PRECISION REPRESENT     \ convert r to string in FOB
   DROP
   IF 45 EMIT THEN             \ if negative display "-"
   1-                          \ subtract 1 from exponent
   DUP 3 MOD                   \ take mod 3
   DUP 0< IF 3 + THEN          \ convert symmetric mod 3 to floored
   1+ >R                       \ number of integer digits to R
   FOB R@ TYPE                 \ display integer digits
   46 EMIT                     \ display "."
   FOB R@ + PRECISION R@ - 0 MAX TYPE  \ display fraction digits
   69 EMIT                     \ display "E"
   R> 1- - .D ;                \ display exponent

: (FS.) ( n flag --  ) \ Display in scientific format number       - not ANS
                \ represented by string in FOB with decimal
                \ exponent = n  and sign = flag
   IF 45 EMIT THEN             \ if negative display "-"
   FOB COUNT EMIT              \ display integer digit
   46 EMIT                     \ display "."
```

```
   PRECISION 1- TYPE                \ display fraction digits
   69 EMIT                          \ display "E"
   1- .D ;                          \ display exponent

: FS. ( r --  )       \ Display r in scientific format
   FDUP F0=
   IF    F0. ." E0 "
   ELSE  FOB PRECISION REPRESENT DROP (FS.)
   THEN ;

: F.  ( r --  )       \ Display r in fixed-point format
   FDUP F0= IF   F0. SPACE EXIT  THEN
   PRECISION >R
   FOB R@ REPRESENT DROP
   OVER DUP 0< IF ABS R@ + 1+ THEN     \ number of digits in fixed
                                \ point format
   1+ OVER IF 1+ THEN                  \ total length of fixed point format
   FXLIMIT U>
   IF (FS.)
   ELSE
     IF 45 EMIT THEN                   \ if negative display "-"
     DUP 0>
     IF                          \ if exponent is positive
       DUP R@ >
       IF                        \ if exponent > precision
         FOB R@ TYPE             \    display string in FOB
         R@ - 0 ?DO 48 EMIT LOOP    \    display required number of "0"s
         46 EMIT                 \    display "."
       ELSE
         FOB OVER TYPE           \ else display integer part of string
         46 EMIT                 \    display "."
         FOB OVER + R@ ROT - TYPE   \    display rest of string
       THEN
     ELSE
       48 EMIT 46 EMIT           \ else display "0."
       ABS 0 ?DO 48 EMIT LOOP       \    display required number of "0"s
       FOB R@ TYPE               \    display string
     THEN  SPACE
   THEN  R> DROP ;

\ ---------------------------------------------------------------------
\ FLOATING POINT INPUT

: >FLOAT ( ca u -- r true | false ) \ If string at ca u represents a valid
                          \ floating point number return its value r
                          \ and true otherwise return false; if syntax
                          \ is valid but number is outside exponent
```

```
                \ range generate an exception
0 TO IP-EXP                 \ initialise exponent accumulator
2>R                         \ save ca u
%0                          \ seed for r
2R@                         \ copy ca u
                            \ sign
OVER C@ 45 =
IF                          \ if first char is "-"
  TRUE >R ADVANCE           \   - sign to R and advance
ELSE
  FALSE >R                  \ else + sign to R
  OVER C@ 43 = IF ADVANCE THEN      \   if char is "+" advance
THEN
                            \ significand integer
DUP 0
?DO
  OVER C@ 48 58 WITHIN
  IF                        \   if next char is a decimal digit
    >R >R                   \     count and addr to R
    %10 F*                  \     multiply total so far by 10
    R@ C@ 48 - S>F F+       \     add next digit
    R> R> ADVANCE           \     count and addr from R and
                            \     advance
  ELSE LEAVE                \   else leave loop
  THEN
LOOP
                            \ decimal point
DUP
IF                          \ if more chars left
  OVER C@ 46 =
  IF                        \   if next char is "."
    ADVANCE                 \     skip it
                            \ significand fraction
    DUP 0
    ?DO
      OVER C@ 48 58 WITHIN
      IF                    \ if next char is a decimal digit
        >R >R               \   count and addr to R
        %10 F*              \   multiply total so far by 10
        R@ C@ 48 - S>F F+   \   add next digit
        IP-EXP 1- TO IP-EXP \   decrement exponent accumulator
        R> R> ADVANCE       \   count, addr from R and advance
      ELSE LEAVE THEN       \ else leave loop
    LOOP
  THEN
THEN
                            \ exponent marker
```

```
  OVER C@ DUP 68 = OVER 69 = OR
  OVER 100 = OR SWAP 101 = OR
  IF                          \ if next char is "D" "E" "d" or "e"
    ADVANCE                     \   skip it
  THEN
                              \ exponent sign
  OVER C@ 45 =
  IF                          \   if next char is "-"
    TRUE >R ADVANCE             \     - sign to R and advance
  ELSE
    FALSE >R                    \   else + sign to R
    OVER C@ 43 = IF ADVANCE THEN      \     if next char is "+" skip it
  THEN
                              \ exponent
  0. 2SWAP                    \ seed for exponent
  BASE @ >R                     \ save base
  DECIMAL >NUMBER               \ parse exponent in decimal
  R> BASE !                   \ restore base

  NIP                           \ drop string address
  IF                          \ if more chars left
    2R> 2DROP 2DROP FDROP       \   clean up stacks
    2R>                         \   ca u from R
    BLANKS?
    IF                        \   if string is all blanks
      %0 TRUE                   \     special case representing zero
    ELSE   FALSE                \   else syntax is invalid
    THEN
  ELSE
    R> IF DNEGATE THEN          \ else incorporate exponent sign
    IP-EXP S>D D+               \   add implied exponent
    10**D*                    \   incorporate exponent into r
    R> IF FNEGATE THEN          \   incorporate sign into r
    TRUE                      \
    2R> 2DROP                   \   drop ca u from R
  THEN ;

: FLOATING (  -- r flag ) \ Convert string at PAD to floating      - not ANS
                    \ point number r. flag true if ok.
  PAD DUP 12 ACCEPT  >FLOAT ;

: F?NUMBER ( ca u -- r 3 | d 2 | n 1 | 0 ) \ Floating point        - not ANS
                                \ replacement vector for ?NUMBER
  2DUP 2>R <?NUMBER> ?DUP
  IF   2R> 2DROP
  ELSE
     2R> >FLOAT DUP IF DROP 3 THEN
```

```
  THEN ;

: FERROR   ( n --  ) \ Floating point replacement              - not ANS
                \ vector for ERROR
  DUP -42 = IF DROP -1 ." FP DIV 0"    ELSE
  DUP -43 = IF DROP -1 ." FP RANGE"    ELSE
  DUP -46 = IF DROP -1 ." FP INV ARG"
          THEN THEN THEN  <ERROR> ;

: FLOAT-ON (  --  ) \ Set ?NUMBER to F?NUMBER so that          - not ANS
                \ interpreter will recognise floating point
                \ numbers. Also revector ERROR to recognise
                \ floating-point error conditions
                \ SHOULD BE INCUDED IN POWER-UP WORD.
  ['] FERROR   'ERROR   !
  ['] F?NUMBER '?NUMBER ! ;

: FLOAT-OFF (  --  ) \ Restore original vectors of ?NUMBER & ERROR - not ANS
                \ This must be done before executing an earlier MARKER
                \ word. To ensure this we incorporate FLOAT-OFF
                \ into re-definitions of MARKER words, see next word:
  ['] <ERROR>    'ERROR   !
  ['] <?NUMBER>  '?NUMBER ! ;

FLOAT-ON

INCLUDE FPB.TDS
```

FPAA2.TDS

```
 DECIMAL
\ ---------------------------------------------------------------------
\ VARIABLES

VARIABLE  #EXP   VARIABLE #SGN    VARIABLE #FLG    VARIABLE #EXP1
VARIABLE  #*SN   VARIABLE #/SN    VARIABLE #+SN1   VARIABLE #+SN2
0 VALUE #HI1
0 VALUE #HI2
0 VALUE #LO1
0 VALUE #LO2

\ ---------------------------------------------------------------------
\ PRIMITIVES
```

```
: #D* ( d1 d2 -- q3 ) \ q3=d1*d2
  TO #HI2  TO #LO2  TO #HI1  TO #LO1
  #LO1   #LO2              UM*
   0    #LO2   #HI1         UM* D+
       #LO1   #HI2         UM* D+
        0    #HI1   #HI2    UM* D+ ;

\ ----------------------------------------------------------------------
\ ANS FORTH FLOATING MATH FUNCTIONS

: #F*  ( r1 r2 -- r3 ) \ r3=r1*r2
  FS #*SN !  1+ #EXP !  2>R              \ save sign & exp 1
  FS #*SN +!  #EXP +!   2R>              \ save sign & exp 2
  #D* 2DUP OR                            \ 64 bit multiply = 0 ?
  IF BEGIN
       DUP 16384 AND 0=
    WHILE  -1 #EXP +! 2>R 0 <-S 2R> ROT <-S DROP
    REPEAT
    2>R  NIP 2R> ROT 0< ABS 0 D+             \ round up (flag as integer)
    DUP 0<
    IF  0 S-> DROP  1 #EXP +!  THEN
    #*SN @ 1 =
    IF  DNEGATE  THEN                    \ recover sign
    #EXP @                               \ recover exponent
  ELSE  FDROP DROP  0 0 0
  THEN  ;

: #F/  ( r1 r2 -- r3 ) \ r3=r1/r2
  FS #/SN !  NEGATE #EXP !               \ Negate divisor & save
  2DUP OR                                \ Check 0 - mantissa
  IF  DNEGATE 2>R  FS #/SN +!  #EXP +!      \ make +ve
    2>R 0 0 2R>                          \ 0 quotient to stack
    31 #FLG !                            \ max no. of l. shifts
    BEGIN  2R@ D+ DUP 0<
      IF    2>R 0 <-S DROP 2R> 2R@ D-
      ELSE  2>R 1 <-S DROP 2R>
      THEN
      0 <-S DROP  -1 #FLG +!   #FLG @ 0=
    UNTIL 2R> 2DROP
    2DROP  #EXP @ 1+ DNORM  #/SN @ 1 = IF  FNEGATE  THEN
  ELSE  -42 THROW  \ ABORT" Divide by 0 error"
  THEN  ;

: #F+  ( r1 r2 -- r3 ) \ r3=r1+r2
  2 PICK 2 PICK OR                       \ r2 non zero
  IF  FS #+SN1 !  #EXP !  2>R  2 PICK 2 PICK OR
```

```
    IF  FS #+SN2 !  #EXP1 !  #EXP @  #EXP1 @ - DUP
      IF  DUP 0> IF    0 DO  0 S-> DROP  LOOP
              ELSE  2R@ ROT ABS
                  0 DO  0 S-> DROP  LOOP
                  2R> 2DROP 2>R
              THEN
      ELSE  DROP
      THEN  2R@ #+SN1 @ #+SN2 @ - #EXP @ #EXP1 @ MAX #EXP !
      IF  D-
      ELSE  D+  1 #EXP +!  0 S-> 0 D+ DUP 0<
        IF  1 #EXP +!  0 S-> DROP  THEN
      THEN  #+SN2 @  IF  DNEGATE  THEN  #EXP @ DNORM
    ELSE FDROP 2R@ #+SN1 @ IF DNEGATE THEN #EXP @
    THEN 2R> 2DROP
  ELSE  FDROP
  THEN ;

: #F-  ( r1 r2 -- r3   r3=r1-r2)   FNEGATE  #F+  ;

: #F>D ( r -- d) \ float to double
  FS #SGN !  DUP 0<
  IF  FDROP 0 0
  ELSE  31 SWAP - DUP 0>
    IF  0 DO  0 S-> DROP  LOOP
    ELSE  IF  -43 THROW  THEN         \ ABORT" Overflow in F>D"
    THEN
    #SGN @  IF DNEGATE THEN
  THEN ;

: #F<   ( r1 r2 -- flag )  #F- F0<  ;


\ ---------------------------------------------------------------------------
\ NON ANS FORTH FLOATING MATH FUNCTIONS

: #F>S   ( r -- n )        \ Float to single
  #F>D ?DUP  IF 1+  IF  -43 THROW  THEN  THEN ;   \ ABORT" Overflow in F>S"
: #F>    ( r1 r2 -- flag )  \ Flag is true if r1 is greater than r2.
  #F- F0>  ;
: #FINT  ( r1 -- r2 )   \ r2 is the floating point integer
                \ value of r1 rounded towards zero.
  DUP 32 <  IF  #F>D  D>F  THEN ;
: #F=    ( r1 r2 -- flag )  \ Flag is true if r1 is equal to r2.
  #F- F0=  ;

 DECIMAL
```

```
\ ----------------------------------------------------------------------
\ CONSTANTS

34 CONSTANT #FXLIMIT    \ If length of fixed-point format would     - not ANS
                \ exceed this F. uses scientific format
                \ instead; it may be set to any value
                \ greater than #PMAX but for strict ANS
                \ compliance it should not be less than 34

10 CONSTANT #PMAX       \ Maximum precision - must not exceed 10    - not ANS
                \ The maximum reliable precision is 8. If precision
                \ is set to 10 the 10th digit may be a
                \ non-significant zero. If precision is set to 9 or
                \ 10 the least significant digit may not be
                \ completely accurate.




\ ----------------------------------------------------------------------
\ VARIABLES

VARIABLE #FOB   8 VDP +!  \ Floating point output buffer           - not ANS
7 VALUE #PRECISION>    \ Number of significant digits for F. FE. FS.
                \                               - not ANS
0 VALUE #OP-EXP         \ Decimal exponent for output              - not ANS
0 VALUE #IP-EXP         \ Accumulator for implied exponent         - not ANS
VARIABLE #%N

: #LNP1 ( r1 -- r2 ) \ r2 = ln (r1 + 1) for 0 <= r1 <= 1           - not ANS
   FDUP %D0 #F*
   %D1 #F+ FOVER #F*  %D2 #F+ FOVER #F*  %D3 #F+ FOVER #F*
   %D4 #F+ FOVER #F*  %D5 #F+ FOVER #F*  %D6 #F+ FOVER #F*
   %D7 #F+ #F* ;

: #FLN  ( r1 -- r2 ) \ r2 is the natural logarithm of r1
   3 ?FERROR  2 PICK 2 PICK OR 0=  2 PICK 0< OR
   IF  -46 THROW  THEN \ ABORT" Invalid argument to LOG"
   1 SWAP 1- #%N !  %1 #F- #LNP1
   #%N @ S>F  %D8 #F* #F+ ;

: #FLOG ( r1 -- r2 ) \ r2 is the base 10 logarithm of r1
   #FLN %D9 #F*  ;

\ ----------------------------------------------------------------------
\ PRIMITIVES
```

```
: #10**D* ( r1 d -- r2 ) \ Primitive used in REPRESENT            - not ANS
                  \ r2 = r1 * (10**d)
  2DUP D0< >R DABS                  \ sign of d to R leaving absolute ud
  9863. 2OVER D< IF -43 THROW THEN     \ pre-emptive test for overflow seed
  2>R %1 2R>                        \ for r3 which will equal 10**ud
  [ TEN-POWERS 14 6 * + ] LITERAL TEN-POWERS
  DO                                \ for each bit of ud ...
    0 S->
    IF                              \ if bit is set
      I -ROT 2>R F@ #F* 2R>          \ multiply r3 by corresponding
                                  \  power of 10 from table
    THEN
    2DUP D0= IF LEAVE THEN          \ leave loop if no more bits set
  6 +LOOP
  2DROP                             \ drop exhausted ud
  R>
  IF                                \ if d was negative
    #F/                             \   r2 = r1 / (10**ud)
  ELSE
    #F*                             \ else r2 = r1 * (10**ud)
  THEN ;

\ FLOATING POINT OUTPUT

: #SETPRECISION ( u --  ) \ Sets number of significant digits to be used
                    \ for F. FE. FS.
  TO #PRECISION> ;

: #PRECISION     (  -- u ) \ Returns number of significant digits used
                    \ for F. FE. FS.
   #PRECISION>  1 MAX  #PMAX  MIN ;

: #REPRESENT ( r ca u -- n flag1 flag2 )
  \ Places at ca a string u digits long representing the significand of r
  \ as a decimal fraction rounded to a number of significant digits which is
  \ the lesser of u and #PMAX,  with the implied decimal point to the left of
  \ the first digit; n is the value of the decimal exponent; flag1
  \ represents the sign of r (true = negative); flag2 is true if r was a
  \ valid floating point number  -  always true in this implementation.
   2DUP 48 FILL                     \ fill string with "0"s
   #PMAX MIN                         \ limit precision
   2>R                              \ ca u to R
   FDUP F0=
   IF                               \ if r = 0
     2R> 2DROP FDROP                \   clear stacks
     0 FALSE TRUE EXIT              \   leave parameters and exit
```

```
THEN
FS                          \ absolute value and sign
2R> ROT >R 2>R                  \ sign to R beneath ca u
FDUP #FLOG #F>S                   \ calculate decimal exponent
DUP TO #OP-EXP                  \ save copy
NEGATE #PMAX +                    \ negate and add #PMAX
S>D #10**D*       ( ... -- r2 )    \ r2 should represent the decimal
                        \  fraction of r "shifted" left by
                        \  PMAX digits so that it can be
                        \  converted to a double-cell integer
                        \  but because the decimal exponent
                        \  calculated by FLOG may not be
                        \  exact and the shifting by PMAX may
                        \  cause overflow some fine tuning is
                        \  needed first; r2 is multiplied or
                        \  divided by 10 to make it as large
                        \  as possible without its binary
                        \  exponent exceeding 31 while OP-EXP
                        \  is adjusted to match:
DUP 31 <
IF
  BEGIN
    FDUP %10 #F* DUP 31 <
  WHILE
    FSWAP FDROP
    #OP-EXP 1- TO #OP-EXP
  REPEAT  FDROP
ELSE
  BEGIN
    DUP 31 >
  WHILE
    %.1 #F*
    #OP-EXP 1+ TO #OP-EXP
  REPEAT
THEN
#F>D                        \ convert to double-cell integer
2DUP COUNT-DIGITS               \ calculate number of digits
DUP #PMAX - #OP-EXP + TO #OP-EXP      \ adjust decimal exponent
R@ 1 MAX                      \ number of digits required
- 0 MAX                       \ number of digits to be rounded off
?DUP
IF                          \ if rounding is needed
  0 >R                      \ seed for non-zero-remainder flag
  1-                        \ reduce digits by 1 less than needed
  BEGIN  ?DUP                   \ ...
  WHILE
    >R
```

```
      10 M/MOD
      ROT R> R> ROT + >R          \ maintain non-zero-remainder flag
      1-
    REPEAT                        \ ...
    10 M/MOD                      \ reduce by final digit
    ROT DUP 5 =
    IF                            \ check for round up required ...
      DROP R@
      IF   TRUE                   \ round to nearest
      ELSE OVER 1 AND             \ if midway round to even
      THEN
    ELSE
      5 >                         \ round to nearest
    THEN                          \ ... flag = true if round up reqd.
    R> DROP                       \ drop non-zero-remainder flag from R
    IF                            \ if round up is required
      2DUP COUNT-DIGITS >R        \ number of digits to R
      1. D+                       \    add 1
      2DUP COUNT-DIGITS           \    count digits again
      R> <>
      IF                          \    if number of digits has changed
        #OP-EXP 1+ TO #OP-EXP     \       adjust decimal exponent
      THEN
    THEN
  THEN
  <#                              \ convert double-cell integer
                                  \ to string
    BEGIN
      10 M/MOD   ROT 48 + HOLD   2DUP D0=
    UNTIL
  #>
  2R>                             \ ca u from R
  ROT MIN CMOVE                   \ move string to ca
  #OP-EXP                         \ fetch decimal exponent
  R>                              \ sign from R
  TRUE ;

: #F0. ( r=0 --  )      \ Display f.p. zero in fixed-point format   - not ANS
  FDROP S" 0.000000000" DROP #PRECISION 1+ TYPE ;

: #FE. ( r --  )        \ Display r in engineering format
  FDUP F0=
  IF    #F0. ." E0 " EXIT
  THEN
  #FOB 3 48 FILL                  \ in case PRECISION is less than 3
  #FOB #PRECISION #REPRESENT      \ convert r to string in FOB
  DROP
```

```
  IF 45 EMIT THEN                  \ if negative display "-"
  1-                           \ subtract 1 from exponent
  DUP 3 MOD                        \ take mod 3
  DUP 0< IF 3 + THEN                \ convert symmetric mod 3 to floored
  1+ >R                         \ number of integer digits to R
  #FOB R@ TYPE                     \ display integer digits
  46 EMIT                       \ display "."
  #FOB R@ + #PRECISION R@ - 0 MAX TYPE  \ display fraction digits
  69 EMIT                       \ display "E"
  R> 1- - .D ;                     \ display exponent

: #(FS.) ( n flag --  ) \ Display in scientific format number       - not ANS
                  \ represented by string in #FOB with decimal
                  \ exponent = n  and sign = flag
  IF 45 EMIT THEN                  \ if negative display "-"
  #FOB COUNT EMIT                    \ display integer digit
  46 EMIT                       \ display "."
  #PRECISION 1- TYPE                \ display fraction digits
  69 EMIT                       \ display "E"
  1- .D ;                      \ display exponent

: #FS. ( r --  )       \ Display r in scientific format
  FDUP F0=
  IF    #F0. ." E0 "
  ELSE  #FOB #PRECISION #REPRESENT DROP #(FS.)
  THEN ;

: #F.  ( r --  )       \ Display r in fixed-point format
  FDUP F0= IF   #F0. SPACE EXIT  THEN
  #PRECISION >R
  #FOB R@ #REPRESENT DROP
  OVER DUP 0< IF ABS R@ + 1+ THEN     \ number of digits in fixed
                           \ point format
  1+ OVER IF 1+ THEN               \ total length of fixed point format
  #FXLIMIT U>
  IF #(FS.)
  ELSE
    IF 45 EMIT THEN               \ if negative display "-"
    DUP 0>
    IF                       \ if exponent is positive
      DUP R@ >
      IF                     \ if exponent > precision
        #FOB R@ TYPE             \    display string in FOB
        R@ - 0 ?DO 48 EMIT LOOP    \    display required number of "0"s
        46 EMIT                  \    display "."
      ELSE
        #FOB OVER TYPE           \ else display integer part of string
```

```
        46 EMIT                 \    display "."
        #FOB OVER + R@ ROT - TYPE   \    display rest of string
      THEN
    ELSE
      48 EMIT 46 EMIT             \ else display "0."
      ABS 0 ?DO 48 EMIT LOOP       \    display required number of "0"s
      #FOB R@ TYPE                 \    display string
    THEN  SPACE
  THEN  R> DROP ;

\ ------------------------------------------------------------------------
\ FLOATING POINT INPUT

: #>FLOAT ( ca u -- r true | false ) \ If string at ca u represents a valid
                        \ floating point number return its value r
                        \ and true otherwise return false; if syntax
                        \ is valid but number is outside exponent
                        \ range generate an exception
  0 TO #IP-EXP                    \ initialise exponent accumulator
  2>R                          \ save ca u
  %0                           \ seed for r
  2R@                           \ copy ca u
                              \ sign
  OVER C@ 45 =
  IF                           \ if first char is "-"
    TRUE >R ADVANCE                \   - sign to R and advance
  ELSE
    FALSE >R                      \ else + sign to R
    OVER C@ 43 = IF ADVANCE THEN      \   if char is "+" advance
  THEN
                              \ significand integer
  DUP 0
  ?DO
    OVER C@ 48 58 WITHIN
    IF                          \   if next char is a decimal digit
      >R >R                        \     count and addr to R
      %10 #F*                       \     multiply total so far by 10
      R@ C@ 48 - S>F #F+             \     add next digit
      R> R> ADVANCE                \     count and addr from R and
                              \     advance
    ELSE LEAVE                    \   else leave loop
    THEN
  LOOP
                              \ decimal point
  DUP
  IF                           \ if more chars left
    OVER C@ 46 =
```

```
  IF                        \   if next char is "."
    ADVANCE                     \     skip it
                            \ significand fraction
    DUP 0
    ?DO
      OVER C@ 48 58 WITHIN
      IF                    \ if next char is a decimal digit
        >R >R                   \   count and addr to R
        %10 #F*                 \   multiply total so far by 10
        R@ C@ 48 - S>F #F+      \   add next digit
        #IP-EXP 1- TO #IP-EXP       \   decrement exponent accumulator
        R> R> ADVANCE           \   count, addr from R and advance
      ELSE LEAVE THEN           \ else leave loop
    LOOP
  THEN
THEN
                            \ exponent marker
OVER C@ DUP 68 = OVER 69 = OR
OVER 100 = OR SWAP 101 = OR
IF                          \ if next char is "D" "E" "d" or "e"
  ADVANCE                       \   skip it
THEN
                            \ exponent sign
OVER C@ 45 =
IF                          \   if next char is "-"
  TRUE >R ADVANCE               \     - sign to R and advance
ELSE
  FALSE >R                      \   else + sign to R
  OVER C@ 43 = IF ADVANCE THEN      \     if next char is "+" skip it
THEN
                            \ exponent
0. 2SWAP                        \ seed for exponent
BASE @ >R                       \ save base
DECIMAL >NUMBER                     \ parse exponent in decimal
R> BASE !                       \ restore base

NIP                           \ drop string address
IF                          \ if more chars left
  2R> 2DROP 2DROP FDROP           \   clean up stacks
  2R>                           \   ca u from R
  BLANKS?
  IF                        \   if string is all blanks
    %0 TRUE                     \     special case representing zero
  ELSE   FALSE                  \   else syntax is invalid
  THEN
ELSE
  R> IF DNEGATE THEN            \ else incorporate exponent sign
```

```
    #IP-EXP S>D D+                \  add implied exponent
    #10**D*                    \  incorporate exponent into r
    R> IF FNEGATE THEN            \  incorporate sign into r
    TRUE                    \
    2R> 2DROP                  \  drop ca u from R
  THEN ;
```

KEYF.TDS

```
HEX CREATE CODES
"U C, "3 C, "2 C, "1 C, BA C, 43 C, 4F C, 7E C,
"D C, "6 C, "5 C, "4 C, 11 C, 0F C, EE C, 23 C,
". C, "9 C, "8 C, "7 C, 33 C, 89 C, 2A C, 12 C,
"E C, "- C, "0 C, "C C, FC C, 23 C, 75 C, 82 C,
DECIMAL

VARIABLE MANY  0 MANY !
VARIABLE MODO
VARIABLE #LIST
VARIABLE #LISTold

20 CONSTANT WIDE   \ Number of characters across the 4 line display.

\ ---------------------------------------------------------------------------
\ WORDS FOR USE IN APPLICATIONS

: {PUT} ( c n --  ) \ Character c is written to position n on display.
   DUP CASE DUP 0                WIDE             WITHIN
            IFCASE                    ENDOF
                  \ line 1 has positions 0 TO WIDE-1
         DUP WIDE              [ WIDE 2* ]  LITERAL WITHIN
            IFCASE  [ 64 WIDE - ] LITERAL +   ENDOF
                  \ line 2 has positions 64 to 63+WIDE
         DUP [ WIDE 2* ]  LITERAL  [ WIDE 3 * ] LITERAL WITHIN
            IFCASE  [ WIDE NEGATE ] LITERAL + ENDOF
                  \ line 3 has positions WIDE to 2*WIDE
         DUP [ WIDE 3 * ] LITERAL  [ WIDE 4 * ] LITERAL WITHIN
            IFCASE  [ 64 WIDE 2* - ] LITERAL + ENDOF
                  \ line 4 has positions 64+WIDE to 63+2*WIDE
         DROP 0 0AT       \ put to home position if not 0 to 4*WIDE-1
         SWAP          \ bring CASE selector back to top of stack
     ENDCASE <PUT> ;       \ place character in corrected position on LCD
```

```
: <LCD    (  --  )  \ Revector EMIT to use LCD
   'EMIT @  R>  2>R   ['] LCDEMIT 'EMIT ! ;
: LCD>    (  --  )  \ Restore vector
   2R>  >R  'EMIT ! ;

 ' {PUT} 'PUT !    \ revector LCD placement primitive to new version
                \ at compile time

\ -------------------------------------------------------------------------
\ DEMONSTRATION    Exclude from final application code

: INILCD (  --  ) \ Test on LM041 display
   0              \ no blink
   0              \ cursor on
   1              \ display on
   0              \ no display shift on write
   1              \ auto incr of char position on write
   $38             \ 2 line LCD (electrically), 5 x 7 matrix
   PRIME             \ initialise LCD
   ['] {PUT} 'PUT !   \ revector LCD primitive to new version {PUT}
 ;

MARKER AQUI0

: TRANSLATE ( n1 - n2 )
CODES + 1- C@ ;

: KKEY ( - n )
BEGIN NEWKEY ?DUP UNTIL ;

: REQU
KKEY  TRANSLATE
;

: REQUEST ( - )
0 MANY !
PAD 15 BLANK  PAD
BEGIN KKEY TRANSLATE DUP "E <>
WHILE  DUP LCDEMIT OVER C! 1+
MANY @ 1+ MANY !
REPEAT 2DROP ;

: REQUESTC ( - )
0 MANY !
PAD 15 BLANK  PAD
BEGIN KKEY TRANSLATE DUP "C = IF 0 MANY ! EXIT THEN DUP "E <>
```

```
WHILE  DUP LCDEMIT OVER C! 1+
MANY @ 1+ MANY !
REPEAT 2DROP ;

: TRADN 0. PAD MANY @ >NUMBER 2DROP DROP ;

: TRADF PAD MANY @ >FLOAT DROP ;
```

LOG111.TDS

```
INCLUDE #7825P-A.TDS
INCLUDE FPA.TDS
INCLUDE FPAA2.TDS
INCLUDE KEYF.TDS

 HEX
A0000. 2CONSTANT CACHE
B0BB8. 2CONSTANT PRES-W     \ 80BB8.
9164A. 2CONSTANT TEMP-W     \ 8164A.
920DC. 2CONSTANT TEMP-HR    \ 820DC.
B3CFC. 2CONSTANT PRES-HR    \ 83CFC.
8591C. 2CONSTANT PRES-DAY
85A48. 2CONSTANT TEMP-DAY
85B74. 2CONSTANT ACQFNC-HR
87794. 2CONSTANT ACQFC-HR
893B4. 2CONSTANT ACQFNC-DAY
894E0. 2CONSTANT ACQFC-DAY
 DECIMAL

\ 8960C.
\ D0000. 2CONSTANT CACHE
\ 9F000. 2CONSTANT PRES-W
\ EF000. 2CONSTANT TEMP-W
\ CF000. 2CONSTANT TEMP-HR
\ C0000. 2CONSTANT PRES-HR
\ CFF00. 2CONSTANT PRES-DAY
\ C0F00. 2CONSTANT TEMP-DAY

\ ---------------------------------------------------------------------
\ VARIABLES

INCLUDE #EXTVAR2.TDS  \ De aqui en adelante cualquier definicion es en MEM EXT
```

```
EVAR FIRST  0 FIRST !!     \ Bandera para indicar primer vez que se enciende.

EVAR EDELAY      \ Number of 813.802ns units between each interrupt
61440 EDELAY !! \ Delay de la palabra INT 12288 (10ms) 24576-20 61440-50

EVAR  #INT       \ Usada para contar el numero de veces que pasa por INT
EVAR  YAH         \ Bandera para indicar que ya hizo lo indicado en la hora
EVAR  YAD

EVAR  #LOG2  0 #LOG2 !!
EVAR  #VEZ   0 #VEZ  !!

2EVAR SLOGP    0. SLOGP 2!!
2EVAR SLOGT    0. SLOGT 2!!
2EVAR #IRQ1    0. #IRQ1 2!!
2EVAR #IRQ1-H    0. #IRQ1-H 2!!
2EVAR #IRQ1-D    0. #IRQ1-D 2!!

FEVAR PREPR  0E0 PREPR F!!
FEVAR TEMPR  0E0 TEMPR F!!

INCLUDE CAL12.TDS

\ ----------------------------------------------------------------------
\ WORD FOR USE IN APPLICATIONS

DECIMAL

: START-LOG (  --  ) \ Enable interrupt which logs data
   $FFB0 6 ONE ;           \ enable  output compare interrupt B, timer 3

: STOP-LOG  (  --  ) \ Disable interrupt which logs data
   $FFB0 6 ZERO  ;         \ disable output compare interrupt B, timer 3

: IRQ1  \ Esta interrupcion se activa con cada flanco de caida en el pin C27

@TIME TNEW 2!! \ Saca el # de TICKS que van en el dia y lo guarda 1TICK=53.33ms
STOP-LOG

#IRQ1 2@@ D0= IF  @TIME Told 2!! AGA8-G2 THEN

#IRQ1 2@@ D0= NOT IF
 1E0
 TNEW 2@@ Told 2@@ D< IF
 1620000. Told 2@@ D- TNEW 2@@ D+  D>F F/  \ Halla la FREQ en hz
 18.75E0 F* FREQ F!!   \ transcurrido (resta de TICKS) entre dos interrupciones
```

```
 ELSE  TNEW 2@@ Told 2@@ D- D>F F/  \ Halla la FREQ en hz equivalente al tiempo
 18.75E0 F* FREQ F!!   \ transcurrido (resta de TICKS) entre dos interrupciones
 THEN

 TNEW 2@@ Told 2!!     \ Guarda el numero de TICKS actual en Told
 360E3 1PULSE F@@ F* RQMAX F@@ F/ FREQ F@@ F* %RQFI F!!  \ % de rata
equivalente

 INTERP-C \ Con % interpolo y hallo el VOL. DEFLUJO CALIBRADO NO
CORREG.(QFNC)

 QFNC-HR F@@ QFNC F@@ F+ QFNC-HR F!!
 CDAY-NC F@@ QFNC F@@ F+ CDAY-NC F!!

 FIRST @@ 1 = IF   \ Si ya transcurrieron 2 seg donde se hallaron P T y FPV (Z)
  AGA7          \ Se corrige el flujo con AGA7 y sale el vol. correg QFC
  QFC-HR F@@    QFC F@@ F+    QFC-HR F!!    \ Y se acumula en el sumador horario
  CDAY-C F@@ QFC F@@ F+    CDAY-C F!! \ a su vez en el CURR DAY corregido
 THEN

 #IRQ1 2@@ 3 UM/MOD DROP 0= IF AGA8-G2 THEN

THEN

#IRQ1 2@@ 1. D+ #IRQ1 2!!  \ incrementa numero de pulsos
#IRQ1-H 2@@ 1. D+ #IRQ1-H 2!!
#IRQ1-D 2@@ 1. D+ #IRQ1-D 2!!
START-LOG
RETURN;

-60 +ORIGIN ASSIGN IRQ1

: INIT-IRQ
\ DIS                   \ Deshabilita interrupciones
  $FFFC C@ $60 OR $FFFC C!   \ Habilita IRQ0 y IRQ1
  $FFF0 C@ $88 AND          \ Les pone prioridad
  $FE OR $FFF0 C!           \ de 7 a IRQ1
\ EIS
;
\ ----------------------------------------------------------

: LOG
  SLOGP 2@@ 1 A-D16 M+ SLOGP 2!!   \ Lee Presion Braz-1
  SLOGT 2@@ 0 A-D16 M+ SLOGT 2!!   \ Lee Presion Braz-2
  #LOG2 @@ 1+ #LOG2 !! ;

: LOG2
```

```
  SLOGP 2@@ 25000 M+ SLOGP 2!!   \ Lee Presion Braz-1
  SLOGT 2@@ 17000 M+ SLOGT 2!!   \ Lee Temperature Braz-1
  #LOG2 @@ 1+ #LOG2 !!  ;

: PROM2
  SLOGP 2@@ D>F #LOG2 @@  S>F F/   MP1 F@@ F* BP1 F@@ F+   PREPR F!!
  SLOGT 2@@ D>F #LOG2 @@  S>F F/   MT1 F@@ F* BT1 F@@ F+   TEMPR F!!
  0 #LOG2 !!  0. SLOGP 2!!  0. SLOGT 2!! ;

: AVERAGE
  PREPR F@@ PREAV F@@ F+ 2E0 F/ PREAV F!!
  TEMPR F@@ TEMAV F@@ F+ 2E0 F/ TEMAV F!! ;

\ ---------------------------------------------------------------------

: INT \ The interrupt routine, one pass through a regularly occurring

  STOP-LOG

  #INT @@ 2 MOD 0= IF  LOG THEN   \ Cada  2 DELAYS

  #INT @@ 1+ DUP #INT !!  \ Incrementa #INT

  40 MOD 0= IF          \ CADA 2 segs hace el PROMED de P y T
    PROM2             \ ." =PT= "
    FIRST @@ 1 = IF      \ Si NO es la primer vez que se enciende el CF
    AVERAGE  THEN \ entonces hace el promedio acumulativo
    FIRST @@ 0= IF       \ Si se acaba de encender el CF
      PREPR F@@ PREAV F!!  \ Establece condiciones iniciales
      TEMPR F@@ TEMAV F!!
      1 FIRST !! \ quita bandera de primera vez
      INIT-IRQ
    THEN \ y halla FPV (ZETA)para ser usado por AGA7 en IRQ1.
  THEN

 \  #INT @@ 830 MOD 0= IF AGA8-G2 THEN  \ Cada 8,3 segundos

  #INT @@ 1200 MOD 0= IF   \ Cada 60 segundos guardo la PR y TEM en vector
    PREAV F@@ PRES-W  #MIN @@ FLOATSD D+ 2DUP POINT-P 2!!  EF!
    TEMAV F@@ TEMP-W  #MIN @@ FLOATSD D+ 2DUP POINT-T 2!!  EF!
    #MIN @@ 1+ #MIN !!
    0 #INT !!  \ \\\\\\\\\\ RESETEO DE INT
  THEN

  MINS @ 60 MOD
  0= IF
     YAH @@ 0= IF
```

```
     1 YAH !!   PROMED-HR
    THEN
  ELSE  0 YAH !!
    0. #IRQ1-H 2!!  \ \\\\\\\\\\ RESETEO DE IRQ-H
  THEN

  MINS @ 0= IF
    YAD @@ 0= IF
    1 YAD !!   PROMED-DY THEN
    ELSE  0 YAD !!
    0. #IRQ1-D 2!!  \ \\\\\\\\\\ RESETEO DE IRQ-D
  THEN

  START-LOG

  EDELAY @@ LATER RETURN;        \ fix time of next interrupt

  -27 +ORIGIN ASSIGN INT  \ associate output compare int B of timer 3

: INITIALISE (  -- false ) \ Set up before main loop, the flag
  DIS
  $FFB0 6 ZERO            \ disable output compare interrupt B, timer 3
   ADS7825-INIT
   CEROS
\  INIT-IRQ
   EIS
   START-LOG  ;

INCLUDE KEY3FF.TDS

: MAIN
  18.12.02 NOW   20.58.00 HRS W!  W@
  INITIALISE    ." START "
  ALL ;

: WORK \ Word to use at power-up
   BEGIN ['] MAIN CATCH ERROR   \ execute WORK
   AGAIN ;
```

KEY3FF.TDS

```
FEVAR VOLT    0E0 VOLT F!!
```

```
FEVAR CASET   0E0 CASET F!!
FEVAR REFCAL  0E0 REFCAL F!!

EVAR IDECO
FEVAR FECHA

\ 7 SET-PRECISION

: L1-  0 AT ! ;
: L2- 20 AT ! ;
: L3- 40 AT ! ;
: L4- 60 AT ! ;
: MEN1 <LCD ." ==== CIG-ECOGAS ====" LCD> ;
: MEN2 <LCD ." === ECOFLOW2020T ===" LCD> ;
: MEN3 <LCD ."           1-DISPLAY             " LCD> ;
: MEN4 <LCD ." 2-CALIBRA  3-CONFIGU" LCD> ;

: MEN WIPE MEN1 MEN2 MEN3 MEN4 ;
: MEN-A WIPE MEN 1 MEN2 ;

: MENDI0 WIPE
L1- <LCD ." MODO DISPLAY" LCD>
L3- <LCD ." Escoja con flechas" LCD>
L4- <LCD ." [CLEAR para volver]" LCD> 1000 MS ;

: MENDI1
CDAY-C F@@ L1- <LCD ." CuDy CF: " #F. LCD>
RQFC      F@@ L2- <LCD ." Rat CFH: " #F. LCD>
PREPR     F@@ L3- <LCD ." Pre PSI: " #F. LCD>
TEMPR     F@@ L4- <LCD ." Temp °F: " #F. LCD>  ;
: MENDI2
PDAY-C F@@ L1- <LCD ." PvDy CF: " #F. LCD>
RQFNC     F@@ L2- <LCD ." URatCFH: " #F. LCD>
FLOWK     F@@ L3- <LCD ." Flw Con: " #F. LCD>
FPV       F@@ L4- <LCD ." FPV    : " #F. LCD>  ;
: MENDI3
#IRQ1   2@@ L1- <LCD ." #PULSOS: " D. LCD>
PREAV   F@@ L2- <LCD ." PREAV : "  #F. LCD>
TEMAV   F@@ L3- <LCD ." TEMAV : "  #F. LCD>
#HORAS  @@  L4- <LCD ." #HORAS: "  . LCD>   ;


: MENCA1 WIPE
L1- <LCD ." *MODO CALIBRACION*" LCD>
L3- <LCD ." Escoja sensor (1-4)" LCD>
L4- <LCD ." [CLEAR para volver]" LCD> 2000 MS ;
: MENCA2
```

```
L1- <LCD ." 1-PRESION - Brazo1" LCD>
L2- <LCD ." 2-TEMPERA - Brazo1" LCD>
L3- <LCD ." 3-PRESION - Brazo2" LCD>
L4- <LCD ." 4-TEMPERA - Brazo2" LCD>  ;
: MENCA3
L1- <LCD ."  ** CALIBRACION **" LCD>
L2- <LCD ." Valores a congelar:" LCD>
L3- <LCD ." Presion [@F]:" LCD>
L4- <LCD ." Tempera[psi]:" LCD> ;
: MENCO WIPE
L1- <LCD ." *MODO CONFIGURACION*" LCD>
L2- <LCD ." Escoja dato (1-4)" LCD>
L3- <LCD ." Mas datos con flechas" LCD>
L4- <LCD ." [CLEAR para volver]" LCD> 2000 MS ;
: MENCO1
L1- <LCD ." 1-MP1 " LCD>
L2- <LCD ." 2-BP1 " LCD>
L3- <LCD ." 3-MT1 " LCD>
L4- <LCD ." 4-BT1 " LCD>  ;
: MENCO2
L1- <LCD ." 1-KCAL " LCD>
L2- <LCD ." 2-KFL  " LCD>
L3- <LCD ." 3-CURRDAY-NC " LCD>
L4- <LCD ." 4-CURRDAY-C  " LCD>  ;
: MENCO3
L1- <LCD ." 1-RQMAX DE TURBINA " LCD>
L2- <LCD ." 2-Fraccion Molar N" LCD>
L3- <LCD ." 3-Fraccion Molar CO2" LCD>
L4- <LCD ." 4-Fraccion Molar H" LCD>  ;

: MENP <LCD ."  PRUEBA " LCD> ;

: PASSW  WIPE DECIMAL
L2- <LCD ." *Entre su password*" LCD>
L3- <LCD ."   y oprima ENTER " LCD>
L4- <LCD ." :: " LCD>
REQUEST TRADN ;

\ ********************************** DISPLAY ***************

: LIST-1 MENDI1 ;
: LIST-2 MENDI2 ;
: LIST-3 MENDI3 ;

: MODO1
WIPE 1 #LIST !  1 #LISTold !
BEGIN
```

```
#LISTold @ #LIST @ <> IF WIPE THEN
#LIST @ #LISTold !

#LIST @ 1 = IF  LIST-1  THEN
#LIST @ 2 = IF  LIST-2  THEN
#LIST @ 3 = IF  LIST-3  THEN

INKEY ?DUP 0> IF
TRANSLATE CASE
"U OF #LIST @ 2 = IF  1 #LIST ! THEN
     #LIST @ 3 = IF  2 #LIST ! THEN ENDOF
"D OF #LIST @ 2 = IF  3 #LIST ! THEN
     #LIST @ 1 = IF  2 #LIST ! THEN ENDOF
ENDCASE
THEN

INKEY TRANSLATE "C = UNTIL
;


\ ***************************** CALIBRACION **************************

: CA-ALL WIPE L1- <LCD ." LEE= " LCD>
L2- <LCD ." PRE:0-300 TEM:32-135" LCD>
L3- <LCD ." Cer[0]-Full[9]-OK[1]" LCD>
L4- <LCD ." Opcion: " LCD>
500 MS  ;

: CALOK 71 AT ! <LCD ." ..CAL.." LCD> 1500 MS ;

: CALI-P1
CA-ALL
5 AT ! PREPR F@@ <LCD  #F. LCD>
\ 25 AT ! REQUESTC TRADF REFCAL F!!
67 AT ! <LCD ." > "  LCD> REQU DUP LCDEMIT
CASE
  "0 OF PREPR F@@ -1E0 #F*  BP1 F!! CALOK ENDOF
  "9 OF MX-P F@@ MP1 F@@ #F* PREPR F@@ #F/  MP1 F!! CALOK ENDOF
  "1 OF 69 AT ! <LCD ." OK " LCD> 1500 MS ENDOF
ENDCASE   WIPE ;

: CALI-T1
CA-ALL
5  AT ! TEMPR F@@ <LCD  #F. LCD>
\ 25 AT ! REQUESTC TRADF REFCAL F!!
55 AT ! REQU DUP LCDEMIT
CASE
```

```
  "0 OF BT1 F@@ 2E0 #F*  TEMPR F@@ #F-  BT1 F!! CALOK ENDOF
  "9 OF MX-T F@@ MT1 F@@ #F* TEMPR F@@ #F/  MT1 F!! CALOK ENDOF
  "1 OF 69 AT ! <LCD ." OK" LCD> 1500 MS ENDOF
ENDCASE   WIPE ;

: CALI-P2
CA-ALL
5 AT ! PREPR F@@ <LCD  #F. LCD>
25 AT ! REQUESTC TRADF REFCAL F!!
55 AT ! REQU DUP LCDEMIT
CASE
  "0 OF BP2 F@@ REFCAL F@@ #F- BP2 F!! CALOK ENDOF
  "9 OF REFCAL F@@ 32767E0 #F/ MP2 F!! CALOK ENDOF
  "1 OF <LCD ." OK " LCD> 1500 MS ENDOF
ENDCASE   WIPE ;

: CALI-T2
CA-ALL
5 AT ! TEMPR F@@ <LCD  #F. LCD>
25 AT ! REQUESTC TRADF REFCAL F!!
55 AT ! REQU DUP LCDEMIT
CASE
  "0 OF BT2 F@@ REFCAL F@@ #F- BT2 F!! CALOK ENDOF
  "9 OF REFCAL F@@ 32767E0 #F/ MT2 F!! CALOK ENDOF
  "1 OF <LCD ." OK " LCD> 1500 MS ENDOF
ENDCASE   WIPE ;

: MODO3
WIPE 1 #LIST !  1 #LISTold !
MENCA1 1000 MS  WIPE
BEGIN
#LISTold @ #LIST @ <> IF WIPE THEN
MENCA2
INKEY ?DUP 0> IF
TRANSLATE CASE
  "1 OF WIPE L1- <LCD ." * PRESION Braz1 *" LCD>  500 MS CALI-P1 ENDOF
  "2 OF WIPE L1- <LCD ." * TEMPERA Braz1 *" LCD>  500 MS CALI-T1 ENDOF
  "3 OF WIPE L1- <LCD ." * PRESION Braz2 *" LCD>  500 MS CALI-P2 ENDOF
  "4 OF WIPE L1- <LCD ." * TEMPERA Braz2 *" LCD>  500 MS CALI-T2 ENDOF
ENDCASE
THEN
50 MS
INKEY TRANSLATE "C = UNTIL
;


\ ************************* CONFIGURACION ************************
```

```
: CONF1 WIPE
L2- <LCD ." OLD= " LCD>  \ PONE
L3- <LCD ." NEW= " LCD>  \ REQUESTC
L4- <LCD ."  [ ENTER o CLEAR ]" LCD> ;

: ASIG   MANY @ 0 <> ;
: INTR-N  @@ 26 AT ! <LCD . LCD> 100 MS 46 AT ! REQUESTC TRADF F>S ASIG
WIPE ;
: INTR-F  F@@ 26 AT ! <LCD #F. LCD> 100 MS 46 AT ! REQUESTC TRADF ASIG
WIPE ;

: MODO4
WIPE    1 #LIST !  1 #LISTold !
DECIMAL
PASSW DUP ." NUMERO=" . 999 <> IF WIPE
L2- <LCD ." !Password Invalido!" LCD> 2000 MS EXIT THEN
MENCO 2000 MS  WIPE
BEGIN
#LISTold @ #LIST @ <> IF WIPE THEN
#LIST @ #LISTold !
#LIST @ 1 = IF  MENCO1  THEN
#LIST @ 2 = IF  MENCO2  THEN
#LIST @ 3 = IF  MENCO3  THEN

INKEY ?DUP 0> IF
TRANSLATE CASE
"U OF #LIST @ 2 = IF  1 #LIST ! THEN
    #LIST @ 3 = IF  2 #LIST ! THEN ENDOF
"D OF #LIST @ 2 = IF  3 #LIST ! THEN
    #LIST @ 1 = IF  2 #LIST ! THEN ENDOF
"1 OF CONF1
    #LIST @ 1 = IF MP1    INTR-F IF MP1  F!!  THEN THEN
    #LIST @ 2 = IF KCAL   INTR-F IF KCAL F!! THEN THEN
    #LIST @ 3 = IF RQMAX  INTR-F IF KCAL F!! THEN THEN
    WIPE ENDOF
"2 OF CONF1
    #LIST @ 1 = IF BP1   INTR-F IF BP1   F!! THEN THEN
    #LIST @ 2 = IF KFL   INTR-F IF KFL   F!! THEN THEN
    #LIST @ 3 = IF %NI   INTR-F IF %NI   F!! THEN THEN
    WIPE ENDOF
"3 OF CONF1
    #LIST @ 1 = IF MT1   INTR-F IF MT1   F!! THEN THEN
    #LIST @ 2 = IF CDAY-NC INTR-F IF CDAY-NC F!! THEN THEN
    #LIST @ 3 = IF %CO2  INTR-F IF %CO2  F!! THEN THEN
    WIPE ENDOF
"4 OF CONF1
    #LIST @ 1 = IF BT1   INTR-F IF BT1   F!! THEN THEN
```

```
    #LIST @ 2 = IF CDAY-C INTR-F IF CDAY-C F!! THEN THEN
    #LIST @ 3 = IF %HI   INTR-F IF %HI   F!! THEN THEN
    WIPE ENDOF

ENDCASE
THEN
100 MS
INKEY TRANSLATE "C = 50 MS UNTIL
;
\ ********************************** LCD-KEY ************************

: ALL
BEGIN
INILCD
MEN REQU
CASE
  "1 OF WIPE L1- MENDI0  1 MODO ! 1000 MS MODO1 ENDOF
  "2 OF WIPE 3 MODO ! MODO3 ENDOF
  "3 OF WIPE 4 MODO ! MODO4 ENDOF
  "0 OF WIPE START ENDOF
ENDCASE

WIPE
KEY? UNTIL
;
```

CAL12.TDS

```
\ VARIABLES DE CALCULATE
\ 1- Variables para conteo de pulsos y Acumulados de flujo
\ RAM ALIGN ROM

2EVAR Told   \ Cuenta de Pulsos al inicio de la Ventana DELTA
2EVAR TNEW   \ Cuenta de Pulsos al final de la Ventana DELTA

FEVAR FREQ  \ Frecuencia equivalente Hz durante la vent DELTA
FEVAR QFNC-HR \ Acumulado de QII calibrados, No Corregidos, en 1 hora

FEVAR QFC-HR  \ Acumulado de QII Corregidos en 1 hora
```

```
\ 2- Variables para promedios de PRE y TEM en una ventana.
\ RAM ALIGN ROM
2EVAR SUM1    \ Suma de lo leido en el periodo por el canal 1
2EVAR SUM2    \ Suma de lo leido en el periodo por el canal 2
\ RAM FALIGN ROM
FEVAR PREAV   \ Promedio de las lecturas de Presion en un periodo
FEVAR TEMAV   \ Promedio de las lecturas de Temp en un periodo

2EVAR POINT-P \ Puntero al ultimo valor promedio de PRES-W
2EVAR POINT-T \ Puntero al ultimo valor promedio de TEMP-W
2EVAR  POINT-PH
2EVAR  POINT-TH

\  Variables para ecuaciones de Calibracion
FEVAR MX-P  300E0 MX-P F!!
FEVAR MX-T  135E0  MX-T F!!

\ FEVAR MP1   0.0097612764E0  MP1 F!!  \ MX-P F@@ 32767E0 F/ MP1 F!!
\ FEVAR BP1   -10E0 BP1 F!! \ -0.32624891E0  BP1 F!!    0E0 BP1 F!!
\ FEVAR MT1   0.0038843574E0 MT1 F!! \ MX-T F@@ 32E0 F- 32767E0 F/ MT1 F!!
\ FEVAR BT1   14.5E0 BT1 F!!  \ 20.813845E0 BT1 F!!  32E0 BT1 F!!

FEVAR MP1   MX-P F@@ 32767E0 F/ MP1 F!!
FEVAR BP1   0E0 BP1 F!!
FEVAR MT1   MX-T F@@ 32E0 F- 32767E0 F/ MT1 F!!
FEVAR BT1   32E0 BT1 F!!

FEVAR MP2   MX-P F@@ 32767E0 F/ MP2 F!!
FEVAR BP2   0E0 BP2 F!!
FEVAR MT2   MX-T F@@ 32E0 F- 32767E0 F/ MT2 F!!
FEVAR BT2   32E0 BT2 F!!

FEVAR CDAY-NC   \ Acumulado no corregido del dia [ACF]
FEVAR CDAY-C    \ Acumulado corregido del dia [ACF]
FEVAR PDAY-NC   \ Acum. no corr. del dia previo
FEVAR PDAY-C    \ Acum. correg. del dia previo

INCLUDE POLA-C.TDS  \ Incluye Rutinas de Interpolacion Presion Constante.
INCLUDE AGA8G2.TDS  \ Algortimo AGA8-Gross2, ajecutar AGA8G-2, sale ZETA
INCLUDE AGA7C.TDS   \ ******* LLAMA A FPV DE GROSS

\ ** Variables para promedios de PRE y TEM en una hora.
EVAR  #MINH   \ Numero de Ventanas en la hora
EVAR  #MIN
EVAR  #HORAS   \ Numero de Horas transcurridas desde START
\ RAM FALIGN ROM
FEVAR SUM11   \  Sumatoria de promedios de ventanas Senal-1
```

```
FEVAR SUM22    \  Sumatoria de promedios de ventanas Senal-2
FEVAR PREAV-HR   \ Promedio de Presion de la hora
FEVAR TEMAV-HR   \ Promedio de Temper. de la hora

: PROMED-HR

\ W@

\ Guarda los Vol No Corr y Corr. en los Vectores de Acumulados correspondientes
QFNC-HR F@@  ACQFNC-HR #HORAS @@  FLOATSD D+  EF!
QFC-HR  F@@  ACQFC-HR  #HORAS @@  FLOATSD D+  EF!

0 #MIN !!   \  Resetea el numero de veces que pasa por MIN
0E0 QFNC-HR F!!  \ Resetea los acumulados de flujo para la siguiente hora
0E0 QFC-HR  F!!  \ tanto no corregido como corregido

0E0 SUM11 F!!  \ Alista sumadores de presion en cero
0E0 SUM22 F!!  \ Alista sumadores de temperat en cero

\ Halla el numero de minutos transcurridos en la hora con la resta de punteros
\ Dividida en 6 bytes que es lo que ocupa un dato flotante
\ No todas las horas son de 60 min, especialmente la primera, cuando se enciende

POINT-P 2@@  PRES-W D-  D>S 6 / 1+ #MINH !!

#MINH @@ 0 DO  \  Halla la suma de todas los promedios de cada minuto
PRES-W I 6 *  M+  EF@  SUM11 F@@  F+   SUM11  F!!
TEMP-W I 6 *  M+  EF@  SUM22 F@@  F+   SUM22  F!!
LOOP

\ Divide en numero de ventanas hallando el promedio de la hora para cada senal
SUM11 F@@  #MINH @@  S>F F/ PREAV-HR  F!!
SUM22 F@@  #MINH @@  S>F F/ TEMAV-HR  F!!

\ Almacena los promedios en los Vectores correspondientes.
PREAV-HR F@@  PRES-HR #HORAS @@ FLOATSD D+ 2DUP POINT-PH 2!!  EF!
TEMAV-HR F@@  TEMP-HR #HORAS @@ FLOATSD D+ 2DUP POINT-TH 2!!  EF!

#HORAS @@ 1+ #HORAS !!   \ Aumenta en 1 el numero de promedios hora hechos
;

\ Variables para los promedios y acumulados del dia. (Quedan en MEM. Externa)
EVAR  #HOUR  \ Numero de horas del dia, en general 24, exepto el primer dia
EVAR  #DIAS  \ Numero de dias transcurridos desde START
2EVAR POINT-DP
2EVAR POINT-DT
2EVAR POINT-Hold  \ Puntero antiguo
```

```
FEVAR PREAV-DAY
FEVAR TEMAV-DAY
FEVAR SUMD1
FEVAR SUMD2

: PROMED-DY

\ Guarda los Vol No Corr y Corr. en los Vectores de Acum DEL DIA
CDAY-NC F@@  ACQFNC-DAY #DIAS @@  FLOATSD D+  EF!
CDAY-C  F@@  ACQFC-DAY  #DIAS @@  FLOATSD D+  EF!

\ Hace Dia Previo = Dia Actual e inicia en cero el Conteo del nuevo dia
CDAY-NC F@@ PDAY-NC F!!   0E0 CDAY-NC F!!
CDAY-C  F@@ PDAY-C  F!!   0E0 CDAY-C  F!!

0E0 SUMD1 F!!  \ Alista sumadores de pres y temp de todo el dia
0E0 SUMD2 F!!

\ Halla el numero de horas del dia, con la resta de punteros
POINT-PH 2@@ POINT-Hold 2@@ D- D>S 6 / 1+ #HOUR !!
POINT-PH 2@@ POINT-Hold 2!!  \ Graba puntero nuevo en antiguo

\ Suma todos lo promedios horarios de PRES y TEMP
#HOUR @@ 0 DO
PRES-HR I FLOATSD D+ EF@  SUMD1 F@@ F+   SUMD1  F!!
TEMP-HR I FLOATSD D+ EF@  SUMD2 F@@ F+   SUMD2  F!!
LOOP
\ Los divide en el numero de horas resultando el promedio del dia
SUMD1 F@@  #HOUR @@  S>F F/ PREAV-DAY  F!!
SUMD2 F@@  #HOUR @@  S>F F/ TEMAV-DAY  F!!

\ Almacena los promedios en los vectores DAY correspondientes y deja un puntero.
PREAV-DAY F@@  PRES-DAY #DIAS @@ FLOATSD D+ 2DUP POINT-DP 2!!  EF!
TEMAV-DAY F@@  TEMP-DAY #DIAS @@ FLOATSD D+ 2DUP POINT-DT 2!!
EF!

#DIAS @@ 1+ #DIAS !!   \ Incrementa en 1 la cuenta de dias.
;

: CEROS
0E0 FREQ F!!  0E0 QFNC-HR F!!  0E0 QFC-HR  F!!
0 #HORAS !!   0 #DIAS !!       0E0 SUM11 F!!    0E0 SUM22 F!!
0E0 CDAY-NC F!!  0E0 CDAY-C F!! 0E0 PDAY-NC F!! 0E0 PDAY-C  F!!
PRES-W  POINT-P 2!!  TEMP-W  POINT-T 2!!  PRES-HR POINT-Hold 2!!
  0 FIRST !!    0 YAH !!      0 YAD !!      0 #INT !!     0 #LOG2 !!
  0 #MIN !!   0. SLOGP 2!!  0. SLOGT 2!!    0. #IRQ1 2!!
;
```

```
: LGPH CR
#HORAS @@ 0  DO
2 SPACES PRES-HR I 6 * M+ EF@ F. 1 SPACES
LOOP ;
: LGTH CR
#HORAS @@ 0 DO
2 SPACES TEMP-HR I 6 * M+ EF@ F. 1 SPACES
LOOP ;
: LGPD CR
#DIAS @@ 0  DO
2 SPACES PRES-DAY I 6 * M+ EF@ F. 1 SPACES
LOOP ;
: LGTD CR
#DIAS @@ 0  DO
2 SPACES TEMP-DAY I 6 * M+ EF@ F. 1 SPACES
LOOP ;
: LGQNCH CR
#HORAS @@ 0  DO
2 SPACES ACQFNC-HR I 6 * M+ EF@ F. 1 SPACES
LOOP ;
: LGQCH CR
#HORAS @@ 0  DO
2 SPACES ACQFC-HR I 6 * M+ EF@ F. 1 SPACES
LOOP ;
: LGQNCD CR
#HORAS @@ 0  DO
2 SPACES ACQFNC-DAY I 6 * M+ EF@ F. 1 SPACES
LOOP ;
: LGQCD CR
#HORAS @@ 0  DO
2 SPACES ACQFC-DAY I 6 * M+ EF@ F. 1 SPACES
LOOP ;
```

POLA-C.TDS

```
\ INCLUDE FPA.TDS
\

DECIMAL

: 6F@@ ( VECTOR INDICE-- ELEMENTO)
```

```
1- 6 * + F@@ ;
: 6F!! ( VECTOR INDICE-- ELEMENTO)
1- 6 * + F!! ;

FEVAR KCAL   1.0E0 KCAL F!!  \ CONSTANTE DE CALIBRACION
FEVAR RQMAX     \ INPUT: Rata de Flujo Maximo (ACFH)
FEVAR 1PULSE    \ INPUT: Equivalencia en ACFH de 1 Pulso de la Turbina
35000E0 RQMAX F!!
100E0 1PULSE F!!

\ INPUT: % de Rata de Flujo de Curva de CalibracioN
FEVAR %QF RAM 15 6 * ALLOT ROM
5E0 %QF 1 6F!!
7E0 %QF 2 6F!!
10E0 %QF 3 6F!!
15E0 %QF 4 6F!!
20E0 %QF 5 6F!!
30E0 %QF 6 6F!!
50E0 %QF 7 6F!!
75E0 %QF 8 6F!!
100E0 %QF 9 6F!!

\ INPUT: Factores de calibracon de Curva de Calibracion
FEVAR 100/K RAM 15 6 * ALLOT ROM
1.001001E0 100/K 1 6F!!
1.003009E0 100/K 2 6F!!
1.003009E0 100/K 3 6F!!
1.003009E0 100/K 4 6F!!
1.003009E0 100/K 5 6F!!
1.003009E0 100/K 6 6F!!
1.001001E0 100/K 7 6F!!
0.990001E0 100/K 8 6F!!
0.998004E0 100/K 9 6F!!

FEVAR %RQFI    \ INPUT: Porcentaje Leido de Rata de Flujo Bruta
FEVAR RQFI     \ Rata de Flujo Bruta
FEVAR RQFNC
0E0 RQFNC F!!  \ OUTPUT: Rata de Flujo Calibrada NO CORREGIDA (ACFH)
FEVAR QFNC
0E0 QFNC F!!  \ OUTPUT: Flujo Calibrado NO CORREGIDO (ACF)
FEVAR 100/KO   \ Valor de Constante de Calibracion
FEVAR QF1      \ Limites de intervalo usado en interpolacion
FEVAR QF2      \ " "
FEVAR 100/K1   \ " "
FEVAR 100/K2   \ " "
FEVAR MTURC    \ Pendiente de la recta de interpolacion Turbina Constante
EVAR NE  9 NE !!   \ Transitorias
```

```
EVAR CON 1 CON !!  \ Transitorias

: INTERP-C

1 CON !!  \
BEGIN
%QF CON @@ 6F@@  %RQFI F@@ F<
WHILE
%QF CON @@ 6F@@  QF1 F!!
%QF CON @@ 1 + 6F@@ QF2 F!!
100/K CON @@ 6F@@ 100/K1 F!!
100/K CON @@ 1 + 6F@@ 100/K2 F!!
CON @@ 1 + CON !!
REPEAT

100/K2 F@@ 100/K1 F@@ F-
QF2 F@@ QF1 F@@ F- F/
MTURC F!!

%RQFI F@@ RQMAX F@@ F* 100E0 F/
RQFI F!!

MTURC F@@ %RQFI F@@ QF1 F@@ F- F*
100/K1 F@@ F+
100/KO F!!

RQFI F@@ 100/KO F@@ F*
RQFNC F!!

\ Saca la Rata de Flujo Calibrado
\ No Corregido (RQFNC)en [ACFH] y luego lo convierte a unidades de
\ volumen de flujo QFNC No Corregido [ACF].

RQFNC F@@ 3600E0 F/ FREQ F@@ F/  KCAL F@@ F*
QFNC F!!

;
```

AGA8G2.TDS

```
\
```

```
\
INCLUDE FPA.TDS
\
INCLUDE #EXTVAR2.TDS

\

FEVAR PDATA  \ Presion de Entrada en **PSIA**  para trabajo de AGA
FEVAR TDATA  \ Tempera de Entrada en **F** para trabajo de AGA

\ 750E0 PDATA F!!
\ 65E0  TDATA F!!

EVAR  DFLAG  0 DFLAG !!  \ Usada en DGROS

FEVAR PATMO   14.519E0     PATMO F!!
FEVAR TBASE   60E0       TBASE F!!  \ Temp de Referencia F
FEVAR PBASE   14.65E0     PBASE F!!  \ Presion de Referencia PSIA
FEVAR SPECG   0.565E0     SPECG F!!  \ Gravedad Especifica
FEVAR %NI     0.01563E0  %NI   F!!
FEVAR %CO2    0.00039E0  %CO2  F!!
FEVAR %HI     0E0        %HI   F!!
FEVAR %CO     0E0        %CO   F!!

FEVAR TBDR
FEVAR HS
FEVAR FPV
FEVAR Df
FEVAR TK
FEVAR PMP
FEVAR ZETA 0E0 ZETA  F!!

INCLUDE #COMMON.TDS
INCLUDE #VIRGS2.TDS
INCLUDE #ZGROS2.TDS
INCLUDE #PGROS.TDS
INCLUDE #DGROS2.TDS
INCLUDE #CHARGS2.TDS

\ ******************************************
: AGA8-G2

\ PREAV F@@ PDATA F!!
\ TEMAV F@@ TDATA F@@

PREAV F@@  PATMO F@@ F+   PDATA F!!
TEMAV F@@  TDATA F!!
```

```
0E0 B011 F!!  0E0 B111 F!!    0E0 B211 F!!
0E0 C011 F!!  0E0 C111 F!!    0E0 C211 F!!

\ TBDR = 5/9 * (TBASE + 459.67) de F a K
TBASE F@@  459.67E0 F+ 5E0 9E0 F/ F* TBDR F!!

\ PB = PBASE * 0.006894757 Pasa de 'psia' a 'MPa'
PBASE F@@  0.006894757E0 F*  PB F!!

0E0 HS F!!

0E0      XIN1 F!!
%NI  F@@ XIN2 F!!
%CO2 F@@ XIN3 F!!
%HI  F@@ XIN4 F!!
%CO  F@@ XIN5 F!!

\ ASIGNACION DE VARIABLES
HS    F@@ HV  F!!
SPECG F@@ GR  F!!
TBDR  F@@ TH  F!!
TBDR  F@@ TD  F!!
PB    F@@ PD  F!!
TBDR  F@@ TGR F!!
PB    F@@ PGR F!!

0E0 MW1 F!!

CHARGS

TDATA F@@ 459.67E0 F+ 5E0 9E0 F/ F* TK F!!
PDATA F@@ 0.006894757E0 F* PMP F!!

PMP F@@ PRE F!!
TK  F@@ TEM F!!
DGROS
DGROSS F@@ DEN F!!

TK F@@ TEM F!!
ZGROS
ZGROSS F@@ ZETA F!!

\ FPV = sqrt (ZB/Z);
ZB F@@ ZETA F@@ F/ FSQRT FPV F!!

\ Df = D* MWX  / 16.01846
```

DEN F@@ MWX F@@ F* 16.01846E0 F/ Df F!!

;

AGA7C.TDS

FEVAR RQFC \ OUTPUT: Rata de Flujo Corregida a Cond. Base [CFH]
0E0 RQFC F!!
FEVAR QFC \ OUTPUT: Volumen de Flujo Corregido a Cond. Base [CF] en la ventana
0E0 QFC F!!

FEVAR FLOWK 0E0 FLOWK F!!
FEVAR KFL 1.019E0 KFL F!!

: AGA7

TBASE F@@ 459.67E0 F+ FPV F@@ F* FPV F@@ F*
TEMPR F@@ 459.67E0 F+ F/
PREPR F@@ PATMO F@@ F+ F* PBASE F@@ F/
KFL F@@ F* FLOWK F!!
FLOWK F@@ QFNC F@@ F*
QFC F!!

\ CR ." ESTE ES RQFC " RQFC F@@ F. CR

QFC F@@ 3600E0 F* FREQ F@@ F* RQFC F!!

;

#COMMON.TDS

\ DECLARACIÓN DE VARIABLES

FEVAR MWX
EVAR PROG

```
FEVAR B011
FEVAR B111
FEVAR B211
FEVAR C011
FEVAR C111
FEVAR C211

FEVAR TOLD
FEVAR BBMIX
FEVAR CCMIX

FEVAR DHIGH
FEVAR PLOW
FEVAR PHIGH
FEVAR THIGH
FEVAR TLOW

FEVAR XIN1
FEVAR XIN2
FEVAR XIN3
FEVAR XIN4
FEVAR XIN5

FEVAR XX1
FEVAR XX2
FEVAR XX3
FEVAR XX4
FEVAR XX5

FEVAR MW1

\ ****  GENERALES

EVAR  IC
FEVAR TEM
FEVAR PRE
FEVAR TOL
FEVAR X1
FEVAR X2
FEVAR X3
FEVAR FFF
FEVAR F1
FEVAR F2
FEVAR F3
FEVAR DGROSS
FEVAR PGROSS
```

```
FEVAR DEN
FEVAR ZGROSS

EVAR  ERRNUM
FEVAR BMIX
FEVAR CMIX
FEVAR TEMP

EVAR FLAG
FEVAR HCH
FEVAR HV
FEVAR GR

FEVAR BCH
FEVAR DOAIR
FEVAR Z0
FEVAR Z0NEW
FEVAR Z0TDPD
FEVAR G1
FEVAR G2
FEVAR HNO
FEVAR MR
FEVAR HTV4
FEVAR HTV5

FEVAR TB  \ TEMP BASE
FEVAR PB  \ PRESION BASE

FEVAR DB
FEVAR ZB
FEVAR TH
FEVAR TD
FEVAR PD
FEVAR TGR
FEVAR PGR
FEVAR VIR
EVAR  OPT
FEVAR EEE
FEVAR X11
FEVAR X22
FEVAR X33
FEVAR X44
FEVAR X55

FEVAR C11
FEVAR C22
```

```
FEVAR C33
FEVAR C23
FEVAR C32
FEVAR C15
FEVAR C44
FEVAR BN2
FEVAR BCO2
FEVAR BH2
FEVAR BCO
FEVAR B12
FEVAR B13
FEVAR B14
FEVAR B15
FEVAR B23
FEVAR B24
```

#VIRGS2.TDS

```
\PROPOSITO: CACULA EL SEGUNTO Y TERCER COEFICIENTE VIRIAL DEL
MODELO GERG
\DESCRIPCION DE ARGUMENTOS:
\T TEMPERATURA EN KELVINS (ENTRADA)
\BMX SEGUNDO COEFICIENTE VIRIAL (SALIDA)
\CMIX TERCER COEFICIENTE VIRIAL (SALIDA)
\BCH COEFICIENTE BINARIO DE INTERACCION CH-CH  (ENTRADA/SALIDA)
\OPT NUMERO DE OPCION
\  OPT=0 CALCULA BCH
\  OPT=1 USA BCH COMO ENTRADA
\ERRNUM BANDERA DE ERROR
\ ERRNUM=0 NO HAY ERROR
\ ERRNUM=1 O 2 LA CONVERGENCIA DE LA INTERACCION FALLO

EVAR FLAKV

: VIRGS

0 FLAKV !!
0 ERRNUM !!

TOLD F@@ TEM F@@ F=
IF
BBMIX F@@ BMIX F!!
CCMIX F@@ CMIX F!!
```

```
1 FLAKV !!
THEN


  FLAKV @@ 0 = IF

XIN1 F@@ XIN1 F@@ F*  X11 F!!
XIN2 F@@ XIN2 F@@ F*  X22 F!!
XIN3 F@@ XIN3 F@@ F*  X33 F!!
XIN4 F@@ XIN4 F@@ F*  X44 F!!
XIN5 F@@ XIN5 F@@ F*  X55 F!!

OPT @@ 0=
IF
B111 F@@ B211 F@@
TEM F@@ F* F+ TEM F@@  F*
B011 F@@ F+
BCH F!!
THEN

0.740910E-3 -0.911950E-6
TEM F@@ F* F+ TEM F@@  F*
-0.144600E0 F+
BN2 F!!

0.403760E-2 -0.516570E-5
TEM F@@ F* F+ TEM F@@  F*
-0.868340E0 F+
BCO2 F!!

0.813385E-4 -0.987220E-7
TEM F@@ F* F+ TEM F@@  F*
-0.110596E-2 F+
BH2 F!!

0.602540E-3 -0.644300E-6
TEM F@@ F* F+ TEM F@@  F*
-0.130820E0 F+
BCO F!!

   THEN

BCO2 F@@ BCH F@@ F* F0<
IF
." RAIZ NEGATIVA "
1 ERRNUM !!
1 FLAKV !!
```

```
THEN

  FLAKV @@ 0 = IF

320E0 TEM F@@ F-
320E0 TEM F@@ F- F*
1.875E-5 F* 0.72E0 F+
BN2 F@@ BCH F@@ F+ F*
2E0 F/
B12 F!!

BCO2 F@@ BCH F@@ F* FSQRT
-0.865E0 F* B13 F!!

-0.250000E-6 TEM F@@ F*
0.271570E-3 F+ TEM F@@ F*
-0.521280E-1 F+ B14 F!!

0.518195E-6 TEM F@@ F*
-0.239381E-5 F+ TEM F@@ F*
-0.687290E-1 F+ B15 F!!

-0.204429E-5 TEM F@@ F*
0.161176E-2 F+ TEM F@@ F*
-0.339693E0 F+ B23 F!!

0.012E0 B24 F!!


X11 F@@ BCH F@@ F*
X22 F@@ BN2 F@@ F* F+
X33 F@@ BCO2 F@@ F* F+
X44 F@@ BH2 F@@ F* F+
X55 F@@ BCO F@@ F* F+

XIN2 F@@ XIN1 F@@ F*
B12 F@@ F* 2E0 F* F+

XIN3 F@@ XIN1 F@@ F*
B13 F@@ F* 2E0 F* F+

XIN4 F@@ XIN1 F@@ F*
B14 F@@ F* 2E0 F* F+

XIN5 F@@ XIN1 F@@ F*
B15 F@@ F* 2E0 F* F+
```

```
XIN3 F@@ XIN2 F@@ F*
B23 F@@ F* 2E0 F* F+

XIN4 F@@ XIN2 F@@ F*
B24 F@@ F* 2E0 F* F+

BMIX F!!

   THEN

OPT @@ 1 =
IF
1 FLAKV !!
THEN

  FLAKV @@ 0 = IF

TEM F@@ TOLD F!!
BMIX F@@ BBMIX F!!

TEM F@@ 270E0 F-
0.0013E0 F*
0.92E0 F+ EEE F!!

1E0 3E0 F/ FFF F!!

TEM F@@ C211 F@@ F*
C111 F@@ F+  TEM F@@ F*
C011 F@@ F+
C11 F!!

TEM F@@ 0.611870E-7 F*
-0.398950E-4 F+  TEM F@@ F*
0.784980E-2  F+
C22 F!!

TEM F@@ -0.837030E-7 F*
0.348880E-4 F+  TEM F@@ F*
0.205130E-2 F+
C33 F!!

TEM F@@  0.467095E-8 F*
-0.364887E-5 F+ TEM F@@ F*
0.104711E-2  F+
C44 F!!

   THEN
```

```
C11 F@@ F0<
IF
." TERMINO INVALIDO EN VIRGS "
1 ERRNUM !!
1 FLAKV !!
THEN

C33 F@@ F0<
IF
." TERMINO INVALIDO EN VIRGS "
1 ERRNUM !!
1 FLAKV !!
THEN

   FLAKV @@ 0 = IF

TEM F@@ 0.343051E-7 F*
-0.276578E-4 F+ TEM F@@ F*
0.736748E-2  F+ 3E0 F*
C15 F!!

TEM F@@ 0.157169E-7 F*
-0.168609E-4 F+ TEM F@@ F*
0.552066E-2  F+ 3E0 F*
C23 F!!

TEM F@@ -0.325798E-7 F*
0.806674E-5 F+ TEM F@@ F*
0.358783E-2 F+ 3E0 F*
C32 F!!

C11 F@@ X11 F@@ F* XIN1 F@@ F*
C22 F@@ X22 F@@ F* XIN2 F@@ F* F+
C33 F@@ X33 F@@ F* XIN3 F@@ F* F+
C44 F@@ X44 F@@ F* XIN4 F@@ F* F+
C23 F@@ X22 F@@ F* XIN3 F@@ F* F+
C32 F@@ X33 F@@ F* XIN2 F@@ F* F+
C15 F@@ X11 F@@ F* XIN5 F@@ F* F+

C11 F@@ C11 F@@ F* C22 F@@ F* FFF F@@ F**
XIN2 F@@ F* X11 F@@ F* EEE F@@ F* 3E0 F* F+

C11 F@@ C22 F@@ F* C22 F@@ F* FFF F@@ F**
XIN1 F@@ F* X22 F@@ F* EEE F@@ F* 3E0 F* F+

C11 F@@ C11 F@@ F* C33 F@@ F* FFF F@@ F**
```

```
XIN3 F@@ F* X11 F@@ F* 0.92E0 F* 3E0 F* F+

C11 F@@ C33 F@@ F* C33 F@@ F* FFF F@@ F**
XIN1 F@@ F* X33 F@@ F* 0.92E0 F* 3E0 F* F+

C11 F@@ C11 F@@ F* C44 F@@ F* FFF F@@ F**
XIN4 F@@ F* X11 F@@ F* 1.2E0 F* 3E0 F* F+

C11 F@@ C22 F@@ F* C33 F@@ F* FFF F@@ F**
XIN1 F@@ F* XIN2 F@@ F* XIN3 F@@ F* 1.1E0 F* 6E0 F* F+

CMIX F!!

CMIX F@@ CCMIX F!!

    THEN
;
```

#ZGROS2.TDS

```
\PROPOSITO: CALCULA EL FACTOR DE COMPRESIBILIDAD COMO FUNCION
DE DENSIDAD Y TEMPERATURA
\DESCRIPCION DE ARGUMENTOS :
\D DENSIDAD MOLAR EN mol/dm^3. (ENTRADA)
\T TEMPERATURA EN KELVINS. (ENTRADA)
\ZGROSS FACTOR DE COMPRESIBILIDAD (SALIDA)

EVAR FLAKZ

: ZGROS

0 FLAKZ !!

0E0 ZGROSS F!!

0 OPT !!
VIRGS

ERRNUM @@ 0<>
IF
1 FLAKZ !!
THEN
```

```
FLAKZ @@ 0= IF

DEN F@@ DEN F@@ F*
CMIX F@@ F*
DEN F@@
BMIX F@@ F* F+
1E0 F+
ZGROSS F!!

THEN
;
```

#PGROS.TDS

```
\PROPOSITO: CALCULA LA PRESION COMO FUNCION DE LA DENSIDAD Y
TEMPERATURA
\DESCRIPCION DE ARGUMENTOS :
\D DENSIDAD MOLAR EN mol/dm^3. (ENTRADA)
\T TEMPERATURA EN KELVINS. (ENTRADA)
\PGROSS PRESION EN MPa. (SALIDA)

: PGROS

ZGROS

ZGROSS F@@ TEM F@@ F*
8.31451E-3  F*
DEN F@@ F*
PGROSS F!!

;
```

#DGROS2.TDS

```
\PROPOSITO: CALCULA DENSIDAD DADAS PRESION Y TEMPERATURA
\LA FUNCION USA EL METODO DE CHAMBERS'S Y PGROSS PARA
\CALCULAR LA DENSIDAD
\DESCRIPCION DE ARGUMENTOS :
\P PRESION EN MPa. (ENTRADA)
\T TEMPERATURA EN KELVINS. (ENTRADA)
\DGROSS DENSIDAD MOLAR A P Y T DADAS; EN mol/dm^3 (SALIDA)
```

```
EVAR FLAK


: DGROS

0 FLAK !!  \ Bandera de RETURN

1E-6 TOL F!!

0E0 DEN F!!
PGROS
PGROSS F@@ X1 F!!

DHIGH F@@ DEN F!!
PGROS
PGROSS F@@ X2 F!!

X1 F@@ DEN F!!
PGROS
PGROSS F@@ PRE F@@ F-
F1 F!!

\ **************

F1 F@@ FABS TOL F@@ F<
IF
  X1 F@@ DGROSS F!!
  1 FLAK !!
THEN

  FLAK @@ 0 = IF  \ ***I1

X2 F@@ DEN F!!
PGROS
PGROSS F@@ PRE F@@ F-
F2 F!!

F2 F@@ FABS TOL F@@ F<
IF
  X2 F@@ DGROSS F!!
  1 FLAK !!
THEN

  THEN     \ ***01

  FLAK @@ 0 = IF     \ ***I2
```

```
F1 F@@ F2 F@@ F* F0<
NOT IF
  ." DGROSS RAIZ NO ENCONTRADA "
  0E0 DGROSS F!!
  1 FLAK !!
THEN
  THEN       \ ***O2


FLAK @@ 0 = IF     \ ***I3

101 1 DO

  FLAK @@ 0 = IF   \ ***I4

0 DFLAG !!

F2 F@@ X1 F@@ F*
F1 F@@ X2 F@@ F* F-
F2 F@@ F1 F@@ F- F/
X3 F!!

X3 F@@ DEN F!!
PGROS
PGROSS F@@ PRE F@@ F-
F3 F!!

   THEN   \ ***O4

   FLAK @@ 0 = IF   \ ***I5

I 6 MOD 0=
IF
  X1 F@@ X2 F@@ F+ 2E0 F/
  DGROSS F!!
ELSE
  F1 F@@ F2 F@@ F-
  F1 F@@ F3 F@@ F- F*
  F2 F@@ F3 F@@ F- F* F0=
  IF
    1 FLAK !!
  THEN

  FLAK @@ 0 = IF      \ ***I6

  X1 F@@ F2 F@@ F* F3 F@@ F*
```

```
  F1 F@@ F2 F@@ F-
  F1 F@@ F3 F@@ F- F* F/

  X2 F@@ F1 F@@ F* F3 F@@ F*
  F2 F@@ F1 F@@ F-
  F2 F@@ F3 F@@ F- F* F/  F+

  X3 F@@ F1 F@@ F* F2 F@@ F*
  F3 F@@ F1 F@@ F-
  F3 F@@ F2 F@@ F- F* F/ F+
  DGROSS F!!

  DGROSS F@@ X1 F@@ F-
  DGROSS F@@ X2 F@@ F- F*
  F0< NOT IF
    X1 F@@ X2 F@@ F+  2E0 F/
    DGROSS F!!
  THEN

  THEN       \ ***O6
THEN

   THEN  \ ***O5

   FLAK @@ 0 = IF   \ ***I7

DGROSS F@@ DEN F!!
PGROS
PGROSS F@@ PRE F@@ F-
FFF F!!

FFF F@@ FABS TOL F@@ F<
IF
  1 FLAK !!
THEN

   THEN    \ ***O7
   FLAK @@ 0 = IF   \ ***I8

FFF F@@ F3 F@@ F*  F0<
IF
  1 DFLAG !!
  DGROSS F@@ X1 F!!
  FFF F@@ F1 F!!
  X3 F@@ X2 F!!
  F3 F@@ F2 F!!
THEN
```

```
F3 F@@ F1 F@@ F* 0E0 F>
IF
  1 DFLAG !!
  DGROSS F@@ X1 F!!
  FFF F@@ F1 F!!
THEN

DFLAG @@ 0=
IF
DGROSS F@@ X2 F!!
FFF F@@ F2 F!!
THEN
   THEN  \ ***O8
LOOP

  THEN  \ ***O3

FLAK @@ 0 = IF
0E0 DGROSS F!! ." DGROS NO CONVERGE"
THEN

;
```

#CHARGS2.TDS

```
\PROPOSITO: DETERMINA EL VALOR CALORIFICO DEL EQUIVALENTE
HIDROCARBONADO
\Y TOMA LOS VALORES PARA CALCULAR LOS PARAMETROS EN EL MODELO
GERG
\USA LA DENSIDAD RELATIVA (Dr) Y LAS FRACCIONES MOLARES DE
NITROGENO Y DIOXIDO DE CARBONO
\DESCRIPCION DE ARGUMENTOS :

\GR DENSIDAD RELATIVA (GRAVEDAD ESPECIFICA)
\X VECTOR DE 5 ELEMENTOS QUE CONTIENEN LAS FRACCIONES MOLARES
DE:
\ X(1) EQUIVALENTE HIDROCARBONADO
\ X(2) NITROGENO
\ X(3) DIOXIDO DE CARBONO
\ X(4) HIDROGENO
\ X(5) MONOXIDO DE CARBONO
\ TH TEMPERATURA DE REFERENCIA PARA EL VALOR CALORIFICO (K)
(ENTRADA)
```

```
\ TD TEMPERATURA DE REFERENCIA PARA LA DENSIDAD MOLAR (K)
(ENTRADA)
\ TGR TEMPERATURA DE REFERENCIA PARA LA DENSIDAD RELATIVA (K)
(ENTRADA)
\ PD PRESION DE REFERENCIA PARA LA DENSIDAD MOLAR (MPa) (ENTRADA)
\ PGR PRESION DE REFERENCIA PARA LA DENSIDAD RELATIVA (MPa)
(ENTRADA)
\ ZB FACTOR DE COMPRESIBILIDAD A 60F Y 14.73psia (SALIDA)
\ DB DENSIDAD MOLAR A 60 F Y 14.73 psia (SALIDA)
\ ERRNUM BANDERA DE FLAG (SALIDA)
\  ERRNUM = 0 NO HAY ERROR
\  ERRNUM = 1 O 2 FALLA LA CONVERGENCIA EN LA ITERACCION
EVAR FLAKC

: CHARGS

0 FLAKC !!

0E0 TOLD F!!
0 ERRNUM !!

-0.12527E0
5.91E-4 TGR F@@ F* F+
6.62E-7 TGR F@@ TGR F@@ F* F* F-
VIR F!!

28.96256E0
8.31451E-3 TGR F@@ F*
PGR F@@ F/  VIR F@@ F+  F/
DOAIR F!!

-2.709328E0 G1 F!!
0.021062199E0 G2 F!!
285.83E0 HTV4 F!!
282.98E0 HTV5 F!!


1E0 Z0 F!!

1E0 XIN2 F@@ F-  XIN3 F@@ F-  XIN4 F@@  F- XIN5 F@@  F- XIN1 F!!


Z0 F@@ GR F@@ F*
8.31451E-3 F* TGR F@@ F*
PGR F@@ F/ DOAIR F@@ F*
MR F!!
```

```
MR F@@ XIN2 F@@ 28.01350E0 F* F-
XIN3 F@@ 44.010E0 F* F-
XIN4 F@@ 2.0159E0 F* F-
XIN5 F@@ 28.01E0  F* F- XIN1 F@@ F/
MW1 F!!


MW1 F@@ G1 F@@ F- G2 F@@ F/
HCH F!!

-0.425468E0
TGR F@@
0.286500E-2 -0.462073E-5
TGR F@@ F* F+ F* F+

0.877118E-3
TGR F@@
-0.556281E-5 0.881510E-8
TGR F@@ F* F+ F* F+
HCH F@@ F* F+

-0.824747E-6
TGR F@@
0.431436E-8 -0.608319E-11
TGR F@@ F* F+ F* F+
HCH F@@ 2E0 F** F* F+
BCH F!!


TGR F@@ TEM F!!
1 OPT !!
VIRGS

BMIX F@@ PGR F@@ F*
8.31451E-3  F/ TGR F@@ F/ 1E0 F+
Z0NEW F!!


BEGIN

Z0 F@@ Z0NEW F@@ F/ 1E0 F-
FABS 0.5E-10 F>
WHILE

  Z0NEW F@@ Z0 F!!
```

```
  Z0 F@@ GR F@@ F*
  8.31451E-3 F* TGR F@@ F*
  PGR F@@ F/ DOAIR F@@ F*
  MR F!!

  MR F@@ XIN2 F@@ 28.01350E0 F* F-
  XIN3 F@@ 44.010E0 F* F-
  XIN4 F@@ 2.0159E0 F* F-
  XIN5 F@@ 28.01E0  F* F- XIN1 F@@ F/
  MW1 F!!

  MW1 F@@ G1 F@@ F- G2 F@@ F/
  HCH F!!

-0.425468E0
TGR F@@
0.286500E-2 -0.462073E-5
TGR F@@ F* F+ F* F+

0.877118E-3
TGR F@@
-0.556281E-5 0.881510E-8
TGR F@@ F* F+ F* F+
HCH F@@ F* F+

-0.824747E-6
TGR F@@
0.431436E-8 -0.608319E-11
TGR F@@ F* F+ F* F+
HCH F@@ 2E0 F** F* F+
BCH F!!


  TGR F@@ TEM F!!
  1 OPT !!
  VIRGS

  BMIX F@@ PGR F@@ F*
  8.31451E-3  F/ TGR F@@ F/ 1E0 F+
  Z0NEW F!!

REPEAT


ERRNUM @@ 0=
NOT IF
```

```
 1 FLAKC !!
THEN

  FLAKC @@ 0= IF


-0.425468E0
0.877118E-3
-0.824747E-6
HCH F@@ F*  F+
HCH F@@ F*  F+
B011 F!!

0.286500E-2
-0.556281E-5
0.431436E-8
HCH F@@ F* F+
HCH F@@ F* F+
B111 F!!

-0.462073E-5
0.881510E-8
-0.608319E-11
HCH F@@ F* F+
HCH F@@ F* F+
B211 F!!

0.646422E-3
0.332805E-6 HCH F@@ F* F-
HCH F@@ F*
-0.302488E0 F+
C011 F!!

-0.422876E-5
0.223160E-8 HCH F@@ F* F+
HCH F@@ F*
0.195861E-2 F+
C111 F!!

0.688157E-8
0.367713E-11 HCH F@@ F* F-
HCH F@@ F*
-0.316302E-5 F+
C211 F!!

XIN1 F@@ XX1 F!!
XIN2 F@@ XX2 F!!
```

```
XIN3 F@@ XX3 F!!
XIN4 F@@ XX4 F!!
XIN5 F@@ XX5 F!!

263E0 TLOW F!!
3.38E0 THIGH F!!
0.5E-9 PLOW F!!
12E0 PHIGH F!!
8E0 DHIGH F!!


PGR F@@  PRE   F!!
TGR F@@  TEM   F!!
DGROS
DGROSS F@@ DB F!!

GR F@@ DOAIR F@@ F*
DB F@@ F/
MWX F!!

60E0 459.67E0 F+ 1.8E0 F/ TB F!!
14.73E0 6894.757E0 F* 1000000E0 F/ PB F!!

PB F@@  PRE   F!!
TB F@@  TEM   F!!
DGROS
DGROSS F@@ DB F!!

DB F@@  DEN  F!!
TB F@@  TEM  F!!
ZGROS
ZGROSS F@@ ZB F!!

  THEN

;
```