INSTITUTO POLITÉCNICO NACIONAL

CENTRO DE INVESTIGACIÓN EN COMPUTACIÓN
(C.I.C)

DISEÑO DE UN MEDIDOR ELÉCTRICO DIGITAL DE PREPAGO

TESIS

QUE PARA OBTENER EL GRADO DE
MAESTRO EN CIENCIAS EN INGENIERÍA DE CÓMPUTO CON ESPECIALIDAD EN SISTEMAS DIGITALES

PRESENTA

ING. JORGE OLVERA ORTEGA

DIRECTOR: M. EN C. AMADEO JOSÉ ARGÜELLES CRUZ

México, D.F. Octubre del 2003

A mis padres Aurora y Otoniel, a mis abuelos Catalina Rosa y Jorge, a mis hermanas Azalia y Elizabeth por su amor y apoyo incondicional.

A Cecilia con amor y a mis amigos.

AGRADECIMIENTOS

Al Centro de Investigación en Computación por darme la oportunidad llevar a cabo mis estudios de posgrado y por permitirme utilizar su infraestructura e instalaciones.

Al Instituto Politécnico Nacional donde curse mis estudios de nivel medio superior, nivel superior y maestría, por proporcionarme los conocimientos y preparación que hoy día tengo

A mi asesor y director de tesis M. en C. Amadeo José Argüelles Cruz por su paciencia y enseñanzas en la electrónica digital así como su asesoramiento para la elaboración de este trabajo de tesis.

A todos los profesores del C.I.C. que imparten los cursos de la maestría en ingeniería de computo que con su trabajo y dedicación me guiaron a lo largo de mis estudios de maestría, en especial al M. en C. Pablo Manrique Ramírez por su disposición y apoyo durante toda la maestría.

.

ÍNDICE

RESUMEN ..... **V**

ABSTRACT ..... **VI**

ÍNDICE ..... **VII**

LISTA DE FIGURAS Y TABLAS ..... **X**

INTRODUCCIÓN **1**

MEDIDORES DE CONSUMO DE ENERGÍA ELÉCTRICA **3**

1.1 Antecedentes 3
1.1.1 Medidores de energía eléctrica 3
1.1.2 Cobro del consumo de energía eléctrica residencial en México 4
1.1.3 Soluciones alternas para la medición y cobro de la energía eléctrica 5
1.1.3.1 Toma de lecturas vía remota 5
1.1.3.2 Sistemas de prepago 6
1.1.4 Medidores de prepago 6
1.1.4.1 Historia del prepago 7
1.1.4.2 Ventajas de los sistemas de medición de prepago 9
1.1.4.3 Comparación entre las tarjetas inteligentes y las fichas con números encriptados 10
1.1.5 Implementación de un sistema de prepago 11
1.1.6 Criptografía 11
1.1.7 Algoritmos de encriptación 12

1.2 Descripción del problema 13

1.3 Definición del problema 14

1.4 Objetivos 14
1.4.1 Objetivo general 14
1.4.2 Objetivos específicos 14

1.5 Justificación 14

1.6 Resumen 14

MEDIDORES DE PREPAGO EXISTENTES **15**

2.1 Soluciones afines 15
2.1.1 Medidor eléctrico de prepago con tecnología de teclado DMS 15
2.1.2 Medidor de prepago de electricidad GEM 16

2.1.3 Liberty Dispensing System 17

2.2 Solución propuesta..... 18

2.3 Resumen..... 18

FUNCIONAMIENTO DEL MEDIDOR ELECTRICO DIGITAL DE PREPAGO 19

3.1 Descripción del funcionamiento 19

3.2 Hardware y software a desarrollar 21
3.2.1 Hardware del medidor..... 21
3.2.2 Software del medidor..... 22

3.3 Recursos de hardware y software a emplear..... 23

3.4 Pruebas de funcionamiento..... 23

3.5 Resumen..... 24

DISEÑO E IMPLEMENTACIÓN DEL MEDIDOR ELÉCTRICO DIGITAL DE PREPAGO..... 25

4.1 Acondicionamiento del canal de corriente..... 26

4.2 Acondicionamiento del canal de voltaje 27

4.3 Procesador de voltaje, corriente y energía CS5460A 28
4.3.1 Operación del CS5460A 28
4.3.2 Configuración del CS5460A en el sistema..... 31

4.4 Microcontrolador AT90S8515..... 32
4.4.1 Configuración del AT90S8515 en el sistema 35

4.5 Codificador de teclas MM74C922 36

4.6 Pantalla de cristal liquido TM16AAC 36

4.7 Implementación del medidor 38

4.8 Resumen..... 41

DISEÑO DEL SOFTWARE DEL MEDIDOR DIGITAL DE PREPAGO 42

5.1 Software del microcontrolador AT90S8515..... 42
5.1.1 Modos de operación 42
5.1.2 Interrupciones..... 44
5.1.2 Subrutinas para el calculo de variables eléctricas..... 50
5.1.2.1 Voltaje..... 50

5.1.2.2 *Corriente* *50*
5.1.2.3 *Potencia aparente* *50*
5.1.2.4 *Potencia real* *51*
5.1.2.5 *Factor de potencia* *51*
5.1.3 Desencriptación 51

5.2 Programa para la encriptación y desencriptación de códigos de 20 dígitos. 55

5.3 Resumen 55

PRUEBAS Y RESULTADOS 56

6.1 Pruebas de funcionamiento del medidor 56

6.3 Resumen 63

CONCLUSIONES 65

7.1 Logros alcanzados 65

7.2 Trabajos a futuro 66

7.3 Resumen 66

BIBLIOGRAFÍA 67

ANEXOS 69

A DIAGRAMAS ELÉCTRICOS DEL MEDIDOR DE PREPAGO 70

B PROGRAMA EN ENSAMBLADOR PARA EL MICROCONTROLADOR 73

C PROGRAMA PARA LA ENCRIPTACION DE CLAVES 136

LISTA DE FIGURAS Y TABLAS

FIGURAS

1.1. Carátula de un medidor electromecánico. 4

1.2. Sistema de recolección remota móvil o de paso. 5

2.1. Medidor de prepago DMS. 16

2.2. Medidor de prepago GEM. 17

2.3. Medidor de prepago Liberty. 18

3.1. Diagrama a bloques del medidor eléctrico digital de prepago. 19

4.1. Detalle del diagrama a bloques del medidor eléctrico digital de prepago. 25

4.2. Transformador de corriente. 26

4.3. Circuito acondicionador del canal de corriente. 26

4.4. Circuito acondicionador del canal de voltaje. 27

4.5. Diagrama a bloques del CS5460A. 28

4.6. Diagrama a flujo del CS5460A. 29

4.7. Registros internos del CS5460A. 30

4.8. Terminales del CS5460A. 31

4.9. Diagrama a bloques del microcontrolador AT90S8515. 33

4.10. Arquitectura interna del microcontrolador AT90S8515. 34

4.11. Terminales del microcontrolador AT90S8515. 35

4.12. Terminales del MM74C922. 36

4.13. Circuito impreso de la tarjeta que contiene el CS5460A. 38

4.14. Tarjeta terminada que contiene el CS5460A. 39

4.15. Diseño en Orcad Layout de la segunda tarjeta. 40

4.16. Imagen de la segunda tarjeta con sus respectivos componentes. 40

5.1. Diagrama de flujo del programa principal.. 43

5.2. Diagrama de flujo de la interrupción TC1M. 46

5.3. Diagrama de flujo de la interrupción EX_INT1. 47

5.4. Diagrama de flujo de la interrupción EX_INT0. 49

6.1. Fecha y hora registradas por el medidor de prepago. 57

6.2. Código introducido por el usuario para abonar crédito en el medidor. 57

6.3. Medidor de prepago con el teclado desactivado. 58

6.4. Crédito abonado en el medidor. 59

6.5. Voltaje registrado por el medidor. 59

6.6. Corriente registrada por el medidor. 60

6.7. Potencia aparente. 60

6.8. Potencia real. 60

6.9. Factor de potencia 61

6.10. Mediciones de voltaje entre un multimetro digital y el medidor digital de prepago. 62

6.11. Mediciones de corriente entre un multimetro digital y el medidor digital de prepago. 63

TABLAS

4.1 Terminales de la pantalla de cristal líquido. 37

6.1 Comparación de mediciones de voltaje y corriente entre un multimetro digital y el medidor digital de prepago.. 61

7.1. Principales especificaciones del medidor de prepago. 66

DISEÑO DE UN MEDIDOR ELÉCTRICO DIGITAL DE PREPAGO

Jorge Olvera Ortega

RESUMEN

Actualmente en el país se emplean en gran medida medidores electromecánicos para la medición del consumo de energía eléctrica residencial, en conjunto con sistemas de facturación y recaudación para el cobro de la energía consumida. Como una alternativa a estos medidores, se diseño un medidor eléctrico que incorpora tecnología digital y de prepago, es decir, un medidor electrónico de energía eléctrica que controla el suministro de electricidad a una residencia en función de la cantidad de energía comprada por el usuario en la forma de una clave encriptada de 20 dígitos. El presente trabajo describe el desarrollo de tal medidor, el cual emplea un microcontrolador, un sensor de energía, un reloj de tiempo real, un teclado, una pantalla de cristal líquido y un relevador en conjunto con un programa de computadora que genera las claves encriptadas. El microcontrolador obtiene del sensor de energía la cantidad de energía consumida por el usuario así como también mediciones de corriente, voltaje y potencia real. En base a estas mediciones calcula los valores de potencia aparente y factor de potencia que son mostrados en la pantalla de cristal líquido a petición del usuario empleando el teclado. Otras funciones realizadas por el microcontrolador son: obtener la fecha y hora del reloj de tiempo real y procesarlas para mostrarlas en la pantalla, desencriptar y verificar la autenticidad de las claves tecleadas por el usuario con el fin de abonar crédito en el medidor y mostrarlo en la pantalla, determinar en base al crédito disponible la conexión o desconexión del relevador que controla el suministro de energía al usuario, mostrar en la pantalla los mensajes que se generan cuando se teclean claves incorrectas o incompletas. Las claves encriptadas son creadas por un programa de computadora empleando el algoritmo de encriptación pública creado por Ron Rivest, Adi Shamir y Leonard Adleman mejor conocido como RSA. Cada clave encriptada contiene el número de serie del medidor así como otros datos que son necesarios para la operación del medidor y seguridad de las claves.

DESIGN OF AN ELECTRICAL DIGITAL PREPAYMENT METER

Jorge Olvera Ortega

ABSTRACT

As of today in the country, electromechanical meters are used for the measurement of residential electric energy consumption altogether with billing and revenue systems to collect for the energy consumption. As an alternative, a meter that incorporates digital and prepayment technology was designed, that is, an electronic meter that measures and controls the flow of energy to a residence in function of the amount of energy bought by the user in the form of an encrypted 20 digit code. This work describes the development of such meter that incorporates the use of a microcontroller, an energy sensor, a real time clock, a keyboard, a liquid crystal display and a relay in conjunction with an encryption code generator computer program. The micro controller reads from the energy sensor the amount of energy consumed by the user and the values of voltage, current and real power. Then using measurements the microcontroller calculates the apparent power and power factor and then displays these values at the users request using the keyboard. Another tasks performed by the micro controller are: reading and processing the time and date from the real time clock, decrypt and verify the authenticity of the codes introduced by the user in order to add credit to the meter and display it, to determine based on the available credit the connection or disconnection of the relay that controls the provision of energy to the user, Display the messages generated when the codes keyed into the meter are incorrect or incomplete. A computer program using the encryption algorithm developed by Ron Rivest, Adi Shamir and Leonard Adleman widely known as RSA creates the encrypted codes. Each encrypted code contains the serial number of the meter and other data that are essential for the meter operation and security of the codes.

INTRODUCCIÓN

Este trabajo tiene como finalidad el desarrollo de un medidor eléctrico digital de prepago. En esta introducción se describe de manera global el contenido de esta tesis.

El capítulo 1, **Medidores de consumo de energía eléctrica**, describe el servicio de energía eléctrica en México, los diferentes tipos de medidores existentes, el cobro de la energía eléctrica en México, las soluciones alternas que existen para realizar la medición y cobro de la energía así como también la descripción de los sistemas de prepago y las ventajas que ofrece un medidor digital de prepago respecto a otros. Se describe la criptografía y algunos algoritmos para la encriptación de datos. Se define el propósito de la tesis, se señalan los objetivos y por ultimo, se justifica la necesidad de diseñar el medidor.

El capítulo 2, **Medidores de prepago existentes**, describe algunos medidores de prepago que existen en el mercado y sus características. Luego, basándose en la información obtenida se proponen las características que debe tener el prototipo a diseñar.

El capítulo 3, **Funcionamiento del medidor eléctrico digital de prepago**, describe cómo debe funcionar el medidor a diseñar, así como el hardware, software y equipo de cómputo a emplear para el diseño del medidor. Al final de este capítulo se mencionan las pruebas de funcionamiento a realizar para verificar el funcionamiento del medidor.

El capítulo 4, **Diseño e implementación del medidor eléctrico digital de prepago**, describe con más detalle los componentes de hardware que componen el medidor, se explica el funcionamiento de cada uno de los componentes principales y la relación e interconexión que existe entre ellos. También se muestra la implementación del hardware del medidor.

El capítulo 5, **Diseño del software del medidor eléctrico digital de prepago**, describe el software creado, el cual permite el funcionamiento del medidor. También se exhibe el software diseñado para el sistema operativo Windows que permite encriptar los códigos que serán introducidos en el medidor y que permiten abonar crédito.

En el capítulo 6, **Pruebas y resultados**, se mencionan las pruebas que se realizaron al medidor para comprobar su funcionamiento y también se presentan los resultados obtenidos de las pruebas.

El capítulo 7, **Conclusiones**, puntualiza los objetivos alcanzados al finalizar el desarrollo del medidor y se proponen las posibles mejoras y trabajos a futuro.

En la **Bibliografía** aparece una lista de las fuentes bibliográficas recopiladas y consultadas para la elaboración de este proyecto.

Por último en los **Anexos**, se incorporan los diagramas esquemáticos del medidor así como también el código fuente de los programas en ensamblador y C.

CAPÍTULO 1

MEDIDORES DE CONSUMO DE ENERGÍA ELÉCTRICA

En este capítulo se describen los medidores de consumo de energía eléctrica y sus características. En base a estas, se plantea y justifica la construcción de un medidor eléctrico digital de prepago.

1.1 Antecedentes

Desde el invento de la electricidad se han desarrollado un gran número de dispositivos para medir el consumo de la energía eléctrica residencial. Al igual que los servicios de agua, gas, teléfono y otros, éstos se cobran por cada una de las compañías que prestan dichos servicios.

En México, desde 1937, la Comisión Federal de Electricidad [CF: 2002] está a cargo de las distintas actividades relacionadas con la generación, transmisión, distribución y comercialización de la energía eléctrica, y en el caso particular de la zona centro del país (Distrito Federal, Estado de México, Hidalgo, Morelos y Puebla) el organismo encargado de esta tarea es la compañía de Luz y Fuerza del Centro [LF: 2002], que es un organismo público descentralizado con personalidad jurídica y patrimonio propios, al igual que la CFE.

1.1.1 Medidores de energía eléctrica

Para determinar el consumo de energía eléctrica realizado por un usuario es necesario contar con un dispositivo que registre dicho consumo, tarea que desempeñan los medidores eléctricos, también conocidos como watthorímetros. Existen varios tipos de medidores, clasificándose en dos grandes grupos: analógicos y digitales. Los medidores analógicos son dispositivos electromecánicos que registran y muestran el consumo de energía eléctrica por hora, medido en “kilowatts-horas” (KWH) en una carátula localizada al frente del medidor, donde se alojan unas manecillas o un contador electromecánico que se incrementa según la cantidad de energía que se esté consumiendo por hora, como se muestra en la figura 1.1. El segundo grupo está integrado por los medidores digitales de estado sólido (sin partes mecánicas móviles) que realizan la misma función que un medidor electromecánico pero que poseen todas las ventajas de un sistema digital como lo es la exactitud, fácil reproducción y estabilidad, entre otras.

Figura 1.1. Carátula de un medidor electromecánico.

La medición se realiza mediante sensores que miden el voltaje y la corriente en la línea de suministro y posteriormente, dichas variables son adquiridas por un procesador o microcontrolador que se encarga de hacer los cálculos correspondientes al consumo, el cual se muestra en un dispositivo de visualización digital (pantallas de cristal líquido).

Además de las ventajas mencionadas, un sistema de tipo digital ofrece una enorme flexibilidad ya que no solo puede registrar y mostrar el consumo sino que puede también registrar y visualizar información adicional como el voltaje, la corriente, la potencia aparente, la potencia real y la potencia reactiva por mencionar algunos parámetros que pueden ser calculados además de la capacidad de comunicaciones vía telefónica, infrarroja o celular.

1.1.2 Problemática a resolver

Para realizar el cobro del consumo de la energía eléctrica residencial en nuestro país, la CFE o Luz y Fuerza del Centro necesitan conocer cuál fue el consumo del cliente en un periodo fijo de tiempo y para dicho efecto cuentan con una infraestructura humana cuya tarea es la de registrar la lectura directamente desde la carátula del medidor, ya que los medidores son de tipo electromecánico, posteriormente los datos recolectados llegan a la gerencia de comercialización para que ésta pueda facturar al cliente por el consumo realizado. Dicha tarea implica un gran gasto en la obtención de las lecturas, en el proceso de facturación y cobro. Otro inconveniente es que debido a que la lectura debe tomarse directamente en el lugar en el que está instalado el medidor y éste no siempre está instalado en un lugar visible en el exterior del inmueble sino dentro del mismo, el recolector tiene que ingresar al interior del inmueble para tomar la lectura, a veces imposible ya que el propietario no se encuentra por lo que el recolector debe regresar una o varias veces hasta encontrar alguien que le permita el acceso al medidor. En el caso de que no sea posible tomar la lectura, el cobro se realiza mediante estimaciones empleando el historial de consumo del cliente por lo que no se cobra el importe exacto por el consumo, además de que resulta incómodo para el usuario permitir el acceso de un extraño a su propiedad dada la creciente inseguridad en el país.

Otro aspecto a considerar son las pérdidas por fraude con los llamados “diablitos” y la técnica de “acostar el medidor” ya que los medidores mecánicos no pueden detectar dichas

condiciones. La única forma de detectar un fraude de esta naturaleza es a través de una inspección visual del medidor y aún así no es del todo confiable ya que el infractor está preparado en caso de una visita del técnico de Luz y Fuerza y esconde cualquier indicio de fraude.

Por último otra causa de gasto la representa la desconexión y reconexión del servicio por falta de pago ya que es necesaria la intervención de un técnico capacitado.

1.1.3 Soluciones alternas para la medición y cobro de la energía eléctrica

En otros países se han implementado otros sistemas que permiten el cobro exacto por el consumo en el servicio y resuelven en su totalidad los problemas descritos anteriormente aunque también cabe mencionar que ningún sistema es 100% eficiente ni seguro.

Entre las soluciones alternas encontramos diferentes tipos de medidores cuyas características se describen a continuación.

1.1.3.1 Toma de lecturas vía remota

La lectura se realiza empleando un medidor que se comunica mediante ondas de radio (radiofrecuencia) [RA: 2002] ya sea con una estación móvil (un vehículo con un receptor/emisor) o con un lector portátil que lleva la persona que va a recolectar las lecturas ver Figura 1.2. Estos lectores tienen la capacidad de acceder a los registros del medidor de manera remota y muy rápida; hay lectores que pueden tomar hasta 500 lecturas en 3 segundos y también tienen la capacidad de desconectar o reconectar al medidor que se accesa así como también detectar fraudes como es el caso de los que fabrica la compañía Ramar.

Otro tipo de medidores tienen la capacidad de comunicarse vía telefónica con la central [MT: 2002] y enviar la información recolectada, esta comunicación es bidireccional por lo que se puede controlar el medidor desde la central.

También encontramos medidores que emplean el principio de la telefonía celular, donde varios medidores se comunican de manera inalámbrica con un nodo que atiende un número determinado de medidores y éste a su vez se comunica ya sea con otro nodo o con la central, vía telefónica.

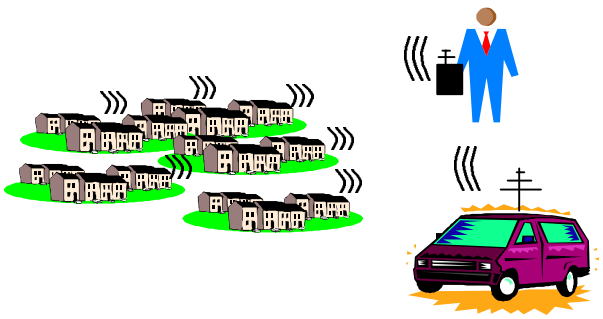


Figura 1.2. Sistema de recolección remota móvil o de paso.

Entre las compañías que ofrecen este tipo de soluciones se encuentran ABB, Siemens, Nams y Ramar, entre otras más

Una vez que alguno de estos sistemas recolecta la información referente al consumo del cliente, la compañía se encarga de realizar el cobro correspondiente al consumo de manera convencional.

1.1.3.2 Sistemas de prepago

Estos sistemas constan de un medidor que a diferencia de los medidores anteriormente expuestos, ofrece una gran flexibilidad ya que al igual que los teléfonos públicos, celulares y la televisión por cable, el usuario tiene la opción de administrar su presupuesto pagando sólo por la energía que tiene pensado consumir en un tiempo determinado. Este tipo de sistemas, por consiguiente, no requiere de un sistema de cobro convencional, eliminando la necesidad por parte de la compañía que presta el servicio de tomar las lecturas y efectuar el cobro correspondiente al consumo. Otra ventaja es que este sistema elimina el costo de desconexión y reconexión y el cliente no tiene que esperar mucho tiempo para la reconexión ya que sólo basta que se introduzca más crédito en el medidor para que éste funcione.

El funcionamiento de este tipo de medidores se explica a continuación.

1.1.4 Medidores de prepago

Las mediciones de prepago [EM: 2002] en su forma más simple se refieren al pago de servicios (electricidad, gas y agua) antes de su uso. El consumidor compra crédito y entonces puede usar la utilidad hasta que se termine dicho crédito o tiempo de uso.

El concepto de mediciones de prepago no es un concepto nuevo ya que fue introducido por primera vez en el Reino Unido en la forma de medidores de gas que operaban con monedas. Este concepto fue refinado en los años 80 a través del uso de transferencias numéricas o electrónicas de crédito y otros tipos de información.

Un sistema electrónico de prepago tradicional opera en tres niveles. En el nivel más bajo están los medidores, los cuales son instalados en la casa del consumidor. El siguiente nivel son las estaciones de venta, las cuales son colocadas en las oficinas del proveedor del servicio o en lugares autorizados. La comunicación entre las estaciones de venta y los medidores se realiza mediante una ficha la cual es empleada para llenar el crédito en el medidor y también para transferir o descargar información al medidor, y en algunos casos, cargar información (dependiendo del tipo de ficha) de regreso a la estación de venta.

En el nivel más alto está el sistema maestro de estaciones o cliente maestro, el cual es necesario para asegurar una base de datos común para reportar y también para proveer un control total gerencial, administrativo, financiero y de ingeniería. El SMS (Stations Master

System) se comunica con las estaciones de venta (clientes vendedores) vía modem u otro enlace de datos.

La información de los consumidores, cambios de tarifa, etc. Son comunicados a la estación de venta y entonces los detalles de las ventas al consumidor se comunican de regreso al SMS. Se debe recordar que un sistema de medición de prepago no es solo una alternativa a las mediciones convencionales sino un sistema completo que incluye el cobro y el control del sistema.

1.1.4.1 Historia del prepago

A través de los años se han desarrollado varios métodos diferentes para transferir crédito desde un punto de venta hacia el dispositivo que controla el suministro de un bien, típicamente un medidor de agua, gas o electricidad.

Estas tecnologías se pueden dividir en dos tipos: fichas y autónomos (sin ficha), por ejemplo, aquellos que necesitan un medio físico para convertir el crédito y aquellos que no. La tecnología de las tarjetas inteligentes (smart card) está recibiendo mucha atención en algunos sectores pero no es necesariamente la tecnología más apropiada para las mediciones de prepago. Por otro lado, los sistemas sin ficha como lo es la transferencia numérica o de "teclado" (keypad) han ido ganando terreno en el ámbito comercial.

A continuación se describen las principales características de los diferentes sistemas de prepago a lo largo de los años:

Medidores de monedas

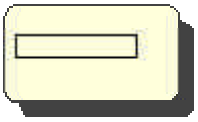
- Las monedas podían ser utilizadas en cualquier medidor
- Robo de monedas de los medidores
- Robo del efectivo en tránsito
- Se requería de un servicio frecuente
- Costo alto de los medidores
- No había información administrativa disponible de una estación de venta

Medidores de boleto

- No eran específicos de un medidor
- Fichas codificadas de varios valores con cintas magnéticas
- Inseguridad inherente - encriptación y administración de claves
- Transferencia de datos en un solo sentido

Medidores de tarjetas magnéticas actuales

- Desarrollados inicialmente en Sudáfrica (1989)
- Pobre seguridad en primeros modelos



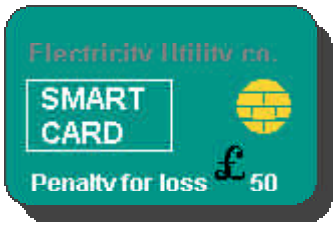
- Medidores de bajo costo
- Fichas específicas para el medidor
- Costo razonable de las fichas desechables (pero aún significativo)
- Interoperabilidad posible entre equipo de varios fabricantes
- Transferencia de datos en un solo sentido

Medidores de llave

- Usa un circuito no volátil (memoria) dentro de una llave de plástico
- La ficha es re-utilizable
- Las fichas son caras
- Los medidores y las llaves están expuestos a daños por electricidad estática
- Transferencia de datos en dos sentidos

Medidores de tarjeta inteligente (smart card)

- Llamadas tarjetas de memoria, embeben un chip en una tarjeta de crédito con funcionalidad comparable
- Algunas versiones mejoradas de este tipo de ficha incluyen tarjetas inteligentes con microprocesadores, los cuales son capaces de desarrollar funciones complejas y proveer niveles más altos de seguridad.
- Hay otro tipo menos popular con dispositivos de almacenamiento óptico, los cuales no requieren de contacto físico (tarjetas inteligentes sin contactos)
- Medidores y fichas caras
- Sistema de dos sentidos que puede transferir mucha información hacia y desde el medidor



Medidores de teclado

- Medidores de bajo costo
- Fichas de bajo costo
- Convenientes para el cliente
- No se requiere físicamente de una ficha (número encriptado)
- Las ventas por Internet, teléfono y radio son posibles
- Seguridad muy alta
- Transferencia de datos en un solo sentido

Otras tecnologías

Otras alternativas incluyen la transferencia de crédito mediante las líneas de transmisión eléctrica (power line carrier) [HP: 2002]. Esta solución aún no ha encontrado una aplicación extensa ya que la tecnología requerida para hacer viable el sistema de prepago es aún muy caro y poco confiable.

1.1.4.2 Ventajas de los sistemas de medición de prepago

Existen varias razones por las cuales un prestador de servicios puede considerar la instalación de un sistema de medición de prepago. Los sistemas de medición de prepago ofrecen las siguientes ventajas:

- Calidad de servicio.
- Pago por adelantado (flujo de efectivo de la utilidad mejorado).
- Los medidores no requieren personal para tomar las lecturas (se emplean en otras funciones).
- No se requiere del envío del estado de cuenta o de un sistema de cobro adicional.
- Eliminación de deudas.
- Eliminación de costos por desconexión y reconexión del servicio.
- Administración completa del ingreso.
- Control de fraudes.
- Cobro a morosos.
- Cobro de tarifas mensuales y de cargos pendientes.
- Fácil de instalar.
- Tarifas por tiempo de uso.
- No se necesita que el prestador del servicio guarde las claves (keys) del consumidor.
- No se necesita acceder a la propiedad del consumidor.
- Eliminación de la toma de lecturas incorrectas del medidor.

El servicio mejorado se obtiene debido a que los sistemas de prepago ofrecen al cliente las siguientes ventajas:

- Administración del presupuesto.
- Control del uso de la energía.
- Conveniencia de compra (a la hora y lugar que convenga al cliente).
- No hay costo de desconexión/reconexión y no hay esperas para la reconexión.
- No se requiere de depósitos.
- Habilidad de pagar deudas.

1.1.4.3 Comparación entre las tarjetas inteligentes y las fichas con números encriptados

La tarjeta inteligente en su forma más sofisticada provee muchas opciones. Puede llevar varios kilobytes de información, lo cual es ciertamente suficiente para la mayoría de las aplicaciones de servicios utilitarios y lo puede hacer en ambos sentidos, hacia el medidor y de regreso al prestador de servicios.

Con dichas características es conveniente su uso en aplicaciones donde las lecturas del medidor u otra información necesiten ser regresadas del consumidor hacia el prestador del servicio para propósitos de estadística o de administración del sistema. También puede realizar tareas como pequeños reportes de las lecturas del consumo o ejecutar comandos de reprogramación o cambio de tarifa con la simple inserción de la tarjeta.

Una posible desventaja es el hecho de que la ficha debe regresarse al punto de venta para así realizar sus tareas o ser reprogramada. Los clientes pueden tener la inconveniencia de ir al supermercado o a algún punto de venta y posiblemente preferirían usar el teléfono o el Internet para adquirir crédito.

La experiencia indica que los usuarios sofisticados de "primer mundo" son receptivos al prepago ya que saben por seguro que no necesitan dejar de hacer sus actividades en algún momento inconveniente.

La funcionalidad de las tarjetas inteligentes tiene su costo. La tarjeta por sí sola es cara y no es inmune a daños físicos, comparada con otras tecnologías de fichas. Se requiere de un equipo especial en las estaciones de venta y también se requiere de un sistema de tecnología de información (por ejemplo, nuevas tarifas para cada tipo de medidor) disponible en todos los puntos de venta para que puedan ser cargados en la tarjeta respectiva.

La ventaja más grande de los medidores de prepago tipo teclado la representan las estructuras flexibles de venta que ofrecen. Las infraestructuras de venta pueden ser establecidas con abastecimiento para todas las condiciones. Dada la naturaleza sin ficha del sistema es posible hacer uso de redes telefónicas, redes de cajeros automáticos, radio e Internet así como computadoras personales existentes.

Dicha flexibilidad en el sistema, adecua el "impulso de compra" del cliente: no se necesita de una ficha de identificación, sólo el número del medidor, un nombre o una dirección y por supuesto una cuenta bancaria o un vínculo a una tarjeta de crédito.

Los números encriptados también ofrecen la increíble ventaja de costos operacionales bajos haciendo uso de fichas de papel desechables de bajo costo (el número también puede ser escrito a mano en un trozo de papel). No se requiere de un hardware de encriptación especial en el punto de venta. Todo lo que se requiere normalmente es una impresora, la cual es un equipo esencial en la mayoría de las tiendas y cajeros. Ya que no se requiere que la ficha de papel sea legible por una máquina, los medidores por sí mismos no requieren de lectores mecánicos de tarjeta. Los medidores se pueden sellar por completo para evitar el ingreso de polvo, humedad o insectos por lo que son robustos. También, si una ficha de

papel se pierde o es destruida, puede ser reemplazada sin riesgo de duplicidad o pérdida financiera.

La seguridad de la ficha está completamente definida por el algoritmo de encriptación y las claves de encriptación y no depende de la inteligencia misma de la ficha.

Desarrollos recientes en esta tecnología han mostrado que los números encriptados se pueden emplear con éxito para sistemas de prepago con tarifas complejas, a pesar que normalmente se requiere de más de un número de transferencia para transportar el nuevo cambio de tarifa al medidor. Esto se debe principalmente a la cantidad restringida de información que puede ser llevada en una sola transferencia.

Para una comunicación bidireccional, la tecnología de teclado puede depender de la visita de un técnico con una herramienta de campo tipo lector para proveer de una retroalimentación hacia el prestador del servicio.

1.1.5 Implementación de un sistema de prepago

La implementación de un sistema de prepago no se limita a la selección de una marca de medidor. Los sistemas de prepago reemplazan no solo el medidor sino también el sistema de cobro, la lectura de los medidores y la administración de la recolección de los ingresos.

La implementación de un sistema de medición de prepago implica un cambio de mentalidad, de la forma en que se administra la recolección de ingresos, de los procedimientos de TI (tecnologías de información), un cambio en el servicio al cliente, en la medición y en el comportamiento del consumidor.
Para poder cosechar los beneficios mencionados, todas las partes necesitan creer en el sistema y apreciar los beneficios que ellos mismos recibirán.

Es también necesario planear la implementación del proyecto por adelantado: la necesidad de programas de actividades detallados, incluyendo la descentralización de recursos y distribución de responsabilidades así como también metas realistas son el factor clave para asegurar el éxito del proyecto.

1.1.6 Criptografía

Criptografía [DE: 1983], del griego *kryptos* que significa escondido y *graphein* que significa escritura, es la ciencia y arte de hacer las comunicaciones ininteligibles a todos excepto a los destinatarios. Los orígenes de la escritura secreta se documentan desde hace cuatro milenios hasta el sistema de escritura jeroglífica de los egipcios. El rompe códigos de David Kahn es la historia de la criptografía desde los primeros intentos hasta su concepción durante la segunda guerra mundial.

Se le llama cripto-sistema al método utilizado para encriptar información. El criptoanálisis es la práctica de derrotar cualquier intento para esconder información. La criptología incluye a la criptografía y al criptoanálisis.

La información original que será escondida se le llama plaintext (texto llano). A la información escondida se le llama ciphertext (texto encriptado). La encriptación es cualquier procedimiento para convertir plaintext en ciphertext. La desencriptación es cualquier procedimiento para convertir ciphertext en plaintext.

Un cripto-sistema está diseñado de tal forma que la desencriptación pueda ser realizada únicamente bajo ciertas condiciones es decir, sólo por las personas que poseen una máquina de desencriptación (en estos días, una computadora) y una pieza particular de información llamada llave de desencriptación, la cual es suministrada a la maquina de desencriptación durante el proceso de desencriptación.

En este proceso la llave de encriptación y de desencriptación pueden ser o no las mismas. Cuando son iguales el cripto-sistema es llamado un sistema de llave simétrica y cuando son diferentes se le llama sistema de llave asimétrica. El sistema mejor conocido del tipo simétrico es el DES (Data Encryption Standard) y el sistema más conocido de llave asimétrica es el RSA (Rivest-Shamir-Adleman).

Hay varias razones para usar la encriptación y el cripto-sistema a emplear es aquel que mejor se ajusta al propósito particular del usuario y que satisface los requerimientos de seguridad, confiabilidad y facilidad de uso.

- Facilidad de uso significa que el cripto-sistema es fácil de comprender
- La confiabilidad significa que el cripto-sistema empleado en la forma que el diseñador pretende, revelará exactamente la información escondida cuando sea necesario.
- La seguridad significa que el cripto-sistema mantendrá la información escondida para todos, excepto para aquellas personas que se pretende puedan ver la información.

1.1.7 Algoritmos de encriptación

Rivest-Shamir-Adleman (RSA)

El algoritmo Rivest-Shamir-Adleman (RSA) [BS: 1996] es uno de los métodos más seguro y popular del tipo llave asimétrica. La seguridad del algoritmo se basa en la dificultad para factorizar números grandes.

El algoritmo de encriptación es el siguiente:

Las llaves públicas son E y N, la llave secreta es D, el dato original es M y el dato encriptado es C.

P y Q son números primos
N = (P)(Q)
L = (P-1)(Q-1)
E = Cualquier número excepto los factores de L
D cumple con la condición (E)(D) mod L = 1

Para encriptar un mensaje se emplea la ecuación:

$$C=M^E \bmod N$$

Para desencriptar un mensaje se emplea la ecuación:

$$M=C^D \bmod N$$

Cualquier técnica criptográfica que pueda resistir un ataque concertado es considerada segura. En este momento el algoritmo RSA es considerado seguro.

Algoritmo DES (Digital Encription Standar)

El algoritmo DES (Estándar de encriptación de datos) es uno de los algoritmos mas populares a nivel mundial.

DES funciona encriptando grupos de mensajes de 64 bits y regresando bloques encriptados del mismo tamaño, realizando permutaciones entre los 64 bits. Es decir, puede realizar 2^{64} permutaciones diferentes. Sin embargo, aunque este método de encriptación ha sido considerado como seguro durante muchos años, es precisamente su popularidad lo que ha llevado al desarrollo de algoritmos y computadoras que comprometen su seguridad.

1.2 Descripción del problema

Existen varias alternativas para medir el consumo de energía eléctrica residencial, siendo el medidor electromecánico el más empleado en el país. Este tipo de medidor no ha cambiado significativamente en muchos años y a pesar de que los sistemas de facturación y recaudación actuales han experimentado algunas mejoras tecnológicas, no resuelven por sí mismos la problemática y las desventajas de emplear tal tipo de medidores.

La tecnología actual permite la implementación de nuevos equipos y sistemas de medición más eficientes, precisos y menos costosos, que a corto plazo tienen efectos positivos para las compañías que prestan el servicio de energía.

Por estas razones y las ventajas que presentan los medidores de prepago del tipo teclado, se propone el desarrollo de un medidor de este tipo como una alternativa a los medidores electromecánicos que se emplean actualmente.

1.3 Definición del problema

Construcción de un medidor eléctrico digital de prepago del tipo teclado que funcione empleando el algoritmo de desencriptación RSA.

1.4 Objetivos

1.4.1 Objetivo general

Diseñar y construir un medidor eléctrico digital de prepago del tipo teclado que emplee el algoritmo de encriptación RSA para abonar crédito.

1.4.2 Objetivos específicos

- Diseñar y construir el hardware del medidor
- Desarrollar el software para el funcionamiento del medidor
- Desarrollar el software para generar las fichas (códigos encriptados) que abonan crédito al medidor.

1.5 Justificación

El desarrollo del medidor eléctrico digital de prepago que se propone, representa una alternativa a los medidores electromecánicos existentes. La implementación de un medidor digital de prepago implica también cambios favorables en los sistemas de facturación y amplia las opciones de los sistemas de recaudación ya que las fichas para abonar crédito al medidor pueden ser vendidas mediante cajeros automáticos, vía Internet, vía telefónica o en centros de atención al cliente. Además, el empleo de este tipo de sistemas crea en el usuario una conciencia de ahorro de energía.

1.6 Resumen

En este capítulo se describieron los medidores de consumo de energía eléctrica, los diferentes tipos y tecnologías empleadas por estos equipos, las diferentes formas en que se cobra el consumo de electricidad y las ventajas de los medidores de prepago. Se describió el proceso de encriptación de datos junto con algunos algoritmos y con base en estos antecedentes, se plantearon los objetivos a alcanzar con la realización de esta tesis, y la justificación de su desarrollo. En el siguiente capítulo se exponen las características de algunos medidores comerciales y se describen las características de la solución propuesta.

CAPÍTULO 2

MEDIDORES DE PREPAGO EXISTENTES

En este capítulo se describen las características de algunos medidores de prepago para, que con base en estas características se propongan las especificaciones del medidor a diseñar.

2.1 Soluciones afines

Para el diseño del medidor eléctrico de prepago se tomarán en cuenta las características y especificaciones técnicas de tres medidores comerciales de prepago que cumplen con las normas internacionales especificadas por la IEC (International Electrotechnical Comission), que es una organización global que prepara y publica estándares internacionales para las tecnologías relacionadas con la electricidad y la electrónica [IE:2002].

2.1.1 Medidor eléctrico de prepago con tecnología de teclado DMS

Fabricante:

Direct Metering (Reino Unido) [DM:2002].

Características:

- Emplea encriptación de 20 dígitos utilizando el algoritmo DES (Data Encryption Standard). Ver figura 2.1.
- Registra eventos tales como los intentos de abrir el medidor y flujo de energía en sentido contrario.
- Cuenta con circuitos de control de carga para encender y apagar el medidor.
- Cumple con la norma internacional IEC 1036.

Especificaciones técnicas:

- Cuenta con un reloj de tiempo real para ajuste de tarifas y estaciones.
- Carga de perfiles de 5, 10, 15, 30 ó 60 minutos.
- Corriente de entrada de 60A ó 100A (máxima).
- Clase de precisión 1 ó 2 para mediciones watts-horas.
- Cuenta con un puerto óptico infrarrojo para programación y lectura de los registros internos del medidor.
- Pantalla de cristal líquido de 10 dígitos.

- Teclado de 12 caracteres.
- Almacenamiento de datos en una memoria no volátil.

Figura 2.1. Medidor de prepago DMS.

2.1.2 Medidor de prepago de electricidad GEM

Fabricante:

Energy Measurements (unión de Siemens Ltd y Spescom, Sudáfrica). [EM: 2002].

Características:

- Tecnología de teclado. Ver figura 2.2.
- Fácil instalación.
- Cumple con la norma internacional IEC 1036.
- Límite de carga programable.
- Encriptación de datos de 16 y 20 dígitos.

Especificaciones técnicas:

- Voltaje 230 V AC (-20%, +15%) 115 V AC (-20%, +15%).
- Corriente 0.1 A hasta 60ª.
- Frecuencia de 45 hasta 65 Hz.
- Monofásico.

- Precisión clase 2.
- Algoritmo de encriptación de 16 ó 20 dígitos.
- Indicador de pantalla de cristal líquido y led´s.
- Información detallada accesible vía el panel de control.

Figura 2.2. Medidor de prepago GEM.

2.1.3 Liberty Dispensing System

Fabricante:

Polimeters Response International Limited, U.K (Reino Unido).

Características:

- Tecnología de teclado. Ver figura 2.3.
- Encriptación de datos de 20 dígitos.
- Desconexión automática y remota.
- Cumple con la norma internacional IEC 1036.

Especificaciones técnicas:

- Voltaje 230 V AC.
- Corriente básica 20 y 40A.
- Corriente máxima 80 y 100A.
- Frecuencia de 40 hasta 60 Hz.
- Visualización en pantalla de cristal líquido.
- Indicador de actividad.

- Comunicaciones vía telefónica
- Monofásico

Figura 2.3. Medidor de prepago Liberty.

2.2 Solución propuesta

Basado en la información previamente expuesta, se tomaron en cuenta aquellas características y especificaciones importantes para proponer las características del medidor que son las siguientes:

Características principales:

Tecnología de teclado
Encriptación de datos de 20 dígitos empleando el algoritmo RSA
Desconexión automática
Desplegado de voltaje, corriente, potencia aparente, potencia real y factor de potencia

Especificaciones técnicas

Monofásico
Voltaje nominal 120 VAC
Corriente nominal 2.5 A
Corriente máxima 7.5 A
Frecuencia 60 Hz
Visualización en pantalla de cristal líquido

2.3 Resumen

En este capítulo se describieron las características de tres medidores comerciales de prepago que cumplen con los estándares internacionales de la IEC. Las especificaciones del medidor a diseñar están basadas en estas características. En el siguiente capítulo se expone de manera general el funcionamiento del medidor a diseñar.

CAPÍTULO 3

FUNCIONAMIENTO DEL MEDIDOR ELÉCTRICO DIGITAL DE PREPAGO

En este capítulo se describe de manera general el funcionamiento del medidor mediante un diagrama de bloques. También se describen el hardware y el software que se necesitan implementar durante el desarrollo del medidor.

3.1 Descripción del funcionamiento

En la figura 3.1 se muestra el diagrama a bloques con los elementos que componen el medidor. Primero, las señales de voltaje y corriente pasan por una etapa de acondicionamiento para ajustar su amplitud de tal forma que puedan ser procesadas por la siguiente etapa. Posteriormente, un procesador convierte las señales analógicas a un formato digital y realiza una serie de cálculos entre los cuales se encuentran el Voltaje RMS, Corriente RMS y Energía, entre otros, que son requeridos por el microcontrolador.

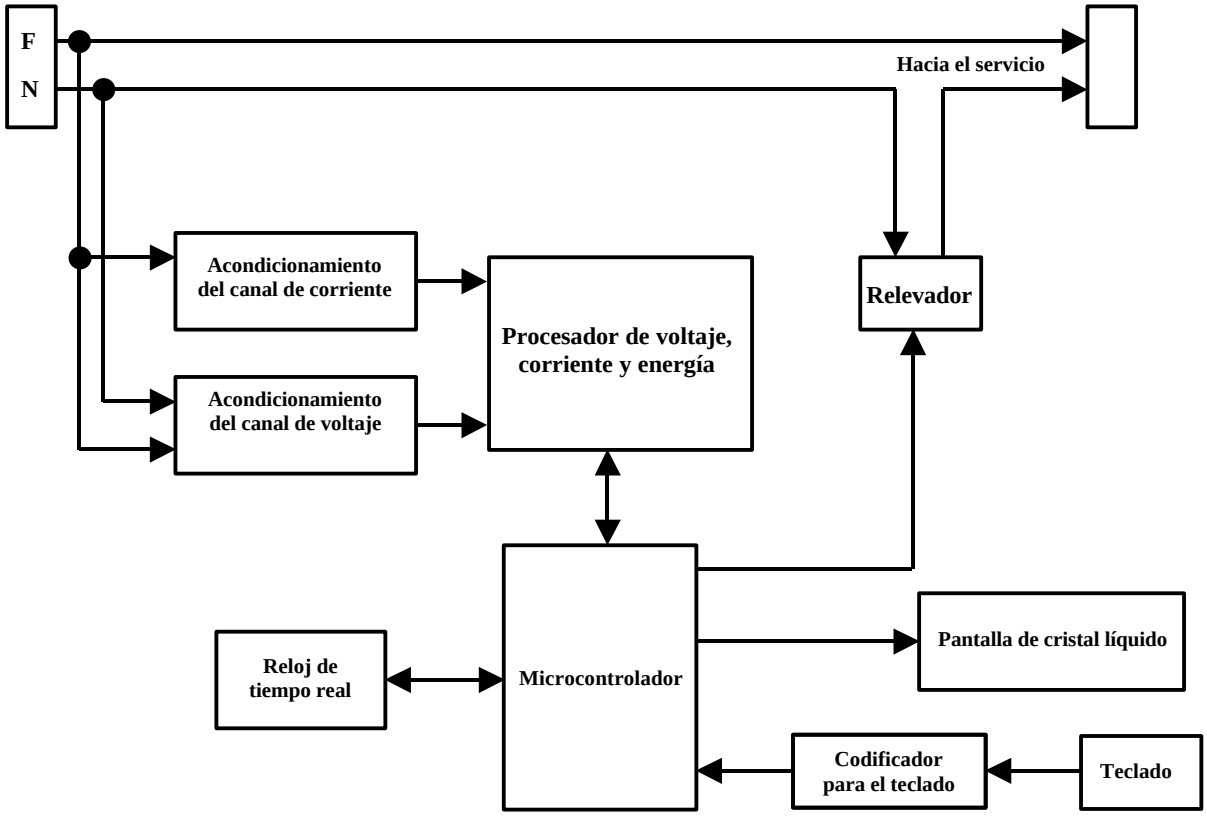


Figura 3.1. Diagrama a bloques del medidor eléctrico digital de prepago.

El microcontrolador es la parte central del sistema, este dispositivo es el encargado del procesamiento de los mensajes a mostrar en la pantalla de cristal líquido y de los comandos recibidos a través del teclado, de los cuales depende en gran parte el funcionamiento de todo el sistema.

Una vez que el sistema está en funcionamiento, se despliega la fecha y hora en la pantalla de cristal líquido así como el crédito disponible (siendo éste el modo normal de despliegue). En caso de que el crédito disponible sea cero, el relevador quedará desactivado de tal forma que no se suministrará energía al usuario.

Para activar o reactivar el relevador, el usuario debe teclear un número de veinte dígitos encriptado previamente por un programa de computadora, que contiene el número de serie del medidor y la cantidad de crédito a abonar entre otros datos que conforman la seguridad del sistema y que sirven para validar la autenticidad del número encriptado.

Una vez que el número ha sido suministrado, se teclea el comando para que el microcontrolador comience la desencriptación del número y valide la transacción, y en el caso de que ésta sea exitosa, se abonará el crédito embebido en el número de veinte dígitos y activará el servicio de energía al usuario, al mismo tiempo que se actualiza en la pantalla la nueva cantidad de crédito disponible. Si la transacción es inválida sólo se permiten dos intentos más y en caso de ser inválidos también, el medidor desactiva el teclado durante 12 horas. Después de transcurrido este tiempo, se reactiva el teclado y se permiten nuevamente tres intentos para abonar crédito.

Para la desencriptación del número de veinte dígitos el microcontrolador emplea el algoritmo conocido como RSA, que es un algoritmo de encriptación público y cuya seguridad recae en la dificultad de factorizar números primos grandes.

Otras funciones del medidor son el despliegue en la pantalla de cristal líquido de las mediciones de voltaje RMS, corriente RMS, potencia aparente, potencia real y factor de potencia (algunas previamente obtenidas del procesador de voltaje, corriente y energía). El despliegue de cada una de estas mediciones depende de la tecla de función presionada y la duración del despliegue de cada medición será de por lo menos diez segundos, con una actualización de las mediciones cada segundo. Después de transcurridos diez segundos, regresará al modo de despliegue normal mostrando nuevamente la fecha, hora y crédito disponible.

El microcontrolador obtiene del procesador de señales de voltaje y corriente pulsos que representan una cantidad fija de energía real consumida o suministrada al usuario, de tal manera que cada vez que se ha consumido un kilowatt-hora el microcontrolador se encarga de decrementar el crédito en base al precio unitario del kilowatt-hora. Dicho valor se encuentra almacenado en la memoria de programa del microcontrolador, lo cual da la flexibilidad de modificar dicho valor según se necesite o cambie la tarifa del kilowatt-hora.

La fecha y la hora del sistema se obtienen desde el reloj de tiempo real cada vez que se inicializa el sistema y después de manera periódica, cada doce horas para asegurar que estos valores sean correctos (varias de las operaciones del microcontrolador están basadas

en rutinas dependientes del tiempo) ya que la desencriptación requiere la fecha exacta para funcionar correctamente.

El reloj de tiempo real incluye 60 bytes de memoria RAM no volátil (NVRAM) que se emplea para almacenar periódicamente los valores de energía consumida, crédito restante y otras variables necesarias para que funcione el sistema correctamente después de una interrupción en la energía del medidor.

3.2 Hardware y software a desarrollar

El medidor eléctrico digital de prepago está constituido en el hardware por los componentes físicos como lo son el microcontrolador, el procesador de corriente, voltaje y energía, el teclado, la pantalla de cristal líquido y la tarjeta sobre la cual se montan estos componentes. El software consta de dos partes, la primera es el programa, que reside en el microcontrolador y permite que funcionen los distintos componentes como un medidor y la segunda parte, es una aplicación en C que permite encriptar los datos de crédito y número de serie, entre otros, en un número de veinte dígitos.

3.2.1 Hardware del medidor

El hardware del medidor consiste en el diseño de la tarjeta de circuito impreso donde irán montados los dispositivos. El microcontrolador a emplear es un AVR AT90S8515 de Atmel, el cual fue escogido por su bajo costo y alto desempeño, su arquitectura RISC (conjunto de instrucciones reducidas) que le permite que casi todas sus instrucciones se ejecuten en un solo ciclo de reloj a diferencia de otros microcontroladores que tardan más de un ciclo en ejecutar una sola instrucción, como es el caso de los microcontroladores 8051 de Intel, los COP 8 de National y los PICs de microchip. Otra de las ventajas de este microcontrolador es que integra en el mismo encapsulado 512 bytes de memoria EEPROM y 512 bytes de RAM, además de que la memoria de programa es del tipo FLASH e ISP (programable en el sistema), lo cual permite programar al microcontrolador sin la necesidad de costosas herramientas y sin tener que quitar el circuito integrado de la tarjeta para ser programado.

El procesador de corriente, voltaje y energía es el CS5460A de Cirrus Logic. Se trata de un convertidor analógico digital (ADC) $\Delta\Sigma$ (delta-sigma) altamente integrado que combina dos convertidores analógico digital $\Delta\Sigma$, funciones para el cálculo de potencia de alta velocidad y una interfaz del tipo SPI (interfaz de tipo serie) en un solo encapsulado. Está diseñado para medir y calcular con precisión: energía, potencia instantánea, voltaje RMS y corriente RMS en aplicaciones monofásicas de medición de potencia. El CS5460A posee una interfaz serie bidireccional para comunicaciones con el microcontrolador y una salida de frecuencia programable que es proporcional a la energía consumida por parte del usuario.

La pantalla de cristal líquido a emplear es la TM162AAC del fabricante Tianma y puede desplegar hasta dieciséis caracteres por línea en las dos que posee, lo que da un total de treinta y dos caracteres máximo para poder desplegar los resultados de los cálculos realizados por el microcontrolador, así como los mensajes para el usuario. Esta pantalla

LCD (pantalla de cristal líquido) contiene un circuito controlador que es el HD44780 de Hitachi, el cual permite desplegar caracteres enviados desde un microcontrolador empleando una interfaz de siete bits como mínimo y once como máximo.

El teclado a emplear es de tipo matricial y contiene dieciséis teclas (dieciséis interruptores en un arreglo matricial de cuatro filas por cuatro columnas). Diez de estas teclas representan los dígitos del cero al nueve y las seis restantes representan letras de la A a la F; los dígitos se emplean para introducir los números a desencriptar y las letras se emplean para acceder a las diferentes funciones del medidor.

Entre el teclado y el microcontrolador se encuentra un circuito integrado, encargado de codificar en cuatro bits cualquiera de las 16 teclas oprimidas en el teclado y de eliminar los rebotes generados al oprimir y soltar una tecla. Este circuito, el MM74C922, fabricado por Fairchild Semiconductor, es un codificador para teclados que contiene la lógica y circuitos necesarios para eliminar rebotes y generar un código binario de cuatro bits que representa la tecla oprimida en el teclado, también genera una señal cada vez que se oprime y suelta una tecla. Este circuito codifica las dieciséis teclas en BCD (binario codificado en decimal), lo cual quiere decir que por cada tecla oprimida se genera un número de cuatro bits de cero al quince en binario, siendo los números del diez al quince la representación de las letras A a la F. Una vez que el usuario ha oprimido y soltado una tecla, este circuito envía una señal al microcontrolador para indicar que se oprimió una tecla, y el microcontrolador responde leyendo el dígito tecleado directamente desde este circuito.

El reloj de tiempo real a utilizar es el DS1307 de Dallas Semiconductor y es el circuito encargado de llevar la cuenta de los segundos, minutos, horas, asi como la fecha en tiempo real. Dichos valores se leen por el microcontrolador de manera periódica sólo para actualizar la cuenta que realiza por sí mismo y mantener la exactitud del tiempo. Otra función del reloj de tiempo real es el almacenamiento periódico de variables que emplea el microcontrolador para funcionar correctamente. El reloj de tiempo real cuenta con una batería de níquel-cadmio de tal forma que su operación no se ve interrumpida aún en ausencia de energía en el medidor.

El acondicionamiento de las señales de corriente y voltaje se realiza mediante un transformador de corriente y un transformador de voltaje, respectivamente, además de varios arreglos de resistencias y capacitores que adecuan las señales para poder ser procesadas por el CS5460A.

Para la conexión-desconexión del servicio al usuario se emplea un relevador y un circuito que permite que el microcontrolador pueda activar y desactivar el relevador de manera segura y directa.

3.2.2 Software del medidor

Para que el medidor pueda operar, se requiere la elaboración de dos programas; el primero, consiste en un programa escrito en ensamblador que será grabado en la memoria FLASH del microcontrolador y es el encargado de la operación de todas las partes del medidor. A manera general este programa debe contener las rutinas necesarias para desplegar

información en la pantalla de cristal líquido, leer información desde el teclado, leer información desde el reloj de tiempo real, leer y enviar comandos al CS5460A, llevar a cabo la desencriptación y la cuenta del crédito disponible, y activar y desactivar el relevador. El segundo programa es una aplicación escrita en C y que corre en Windows; este programa es el encargado de crear y encriptar un numero de veinte dígitos que contiene la información del número de serie del medidor, la fecha y la cantidad de crédito a introducir al medidor, junto con dos dígitos verificadores para evitar la generación de mas de un número igual.

3.3 Recursos de hardware y software a emplear

- Computadora con procesador Intel o AMD que funcione a más de 350 MHz, con un puerto serie y uno paralelo disponibles.
- Emulador ICE200 de Atmel para emular el funcionamiento del microcontrolador en tiempo real.
- Adaptador paralelo para programar el microcontrolador.
- Microcontrolador AT90S8515 de Atmel.
- Circuito integrado CS5460A de Cirrus Logic.
- Reloj de tiempo real DS1307 de Dallas Semiconductor.
- Codificador de 16 teclas MM74C922 de Fairchild Semiconductor.
- Orcad 9 para Windows. [CA: 2003]
- Visual C++ 6 para el desarrollo de la aplicación de encriptación.
- AVR Studio 4 para el desarrollo del código del microcontrolador y la emulación.
- Robot LPKF Protomat 95s/II para la fabricación de la tarjeta.
- Componentes discretos (resistencias y capacitores).
- Conectores, headers y cables.
- Teclado de matriz de 16 teclas.
- Relevador.
- Transformador de voltaje 120V/12V/1A.
- Transformador de corriente 120V/5A.

3.4 Pruebas de funcionamiento

Las pruebas para verificar el correcto funcionamiento del medidor son las siguientes:

- Verificar que la corriente RMS desplegada por el medidor coincida con la mostrada por un multímetro
- Verificar que el voltaje RMS desplegado por el medidor coincida con el mostrado por un multímetro
- Tras conectar una carga resistiva de valor conocido, verificar que la potencia aparente es correcta.
- Tras conectar una carga resistiva de valor conocido, verificar que la potencia real es correcta.
- Calcular el factor de potencia con base en las potencias aparente y real, y comparar con el resultado mostrado por el medidor.

- Verificar que el tiempo y la fecha del sistema son correctos, comparando con un reloj durante varios días a intervalos aleatorios.
- Verificar que el medidor desactiva el relevador cuando el crédito es cero.
- Comprobar que el medidor está registrando apropiadamente el consumo, conectando una carga de valor conocido y tomando el tiempo que permanece conectada. Con esto será posible calcular la energía consumida y compararla con la que registra el medidor.
- Introducir un número encriptado válido de veinte dígitos varias veces para comprobar que no puede ser utilizado más de una vez para abonar crédito en el medidor.
- Introducir varios números encriptados no válidos para comprobar que éstos no son aceptados por el medidor.

3.5 Resumen

En este capítulo se describió, mediante un diagrama a bloques, el funcionamiento general del medidor y los componentes que lo conforman, el hardware a desarrollar y de manera global el papel que desempeña cada uno de sus componentes. Se mencionaron también las características principales del software a diseñar y cuales son las herramientas de hardware y software a emplear para llevar a cabo la realización del medidor. Por último, se describieron las pruebas de funcionamiento que se deben realizar al medidor para comprobar que opera correctamente.

CAPÍTULO 4

DISEÑO E IMPLEMENTACIÓN DEL MEDIDOR ELÉCTRICO DIGITAL DE PREPAGO

En este capítulo se describe el funcionamiento del medidor eléctrico digital de prepago especificando de forma más detallada los componentes que conforman el medidor y su implementación en hardware.

En la figura 4.1 se muestran a detalle los componentes principales que conforman al medidor digital de prepago, especificando los dispositivos de hardware a utilizar en su diseño.

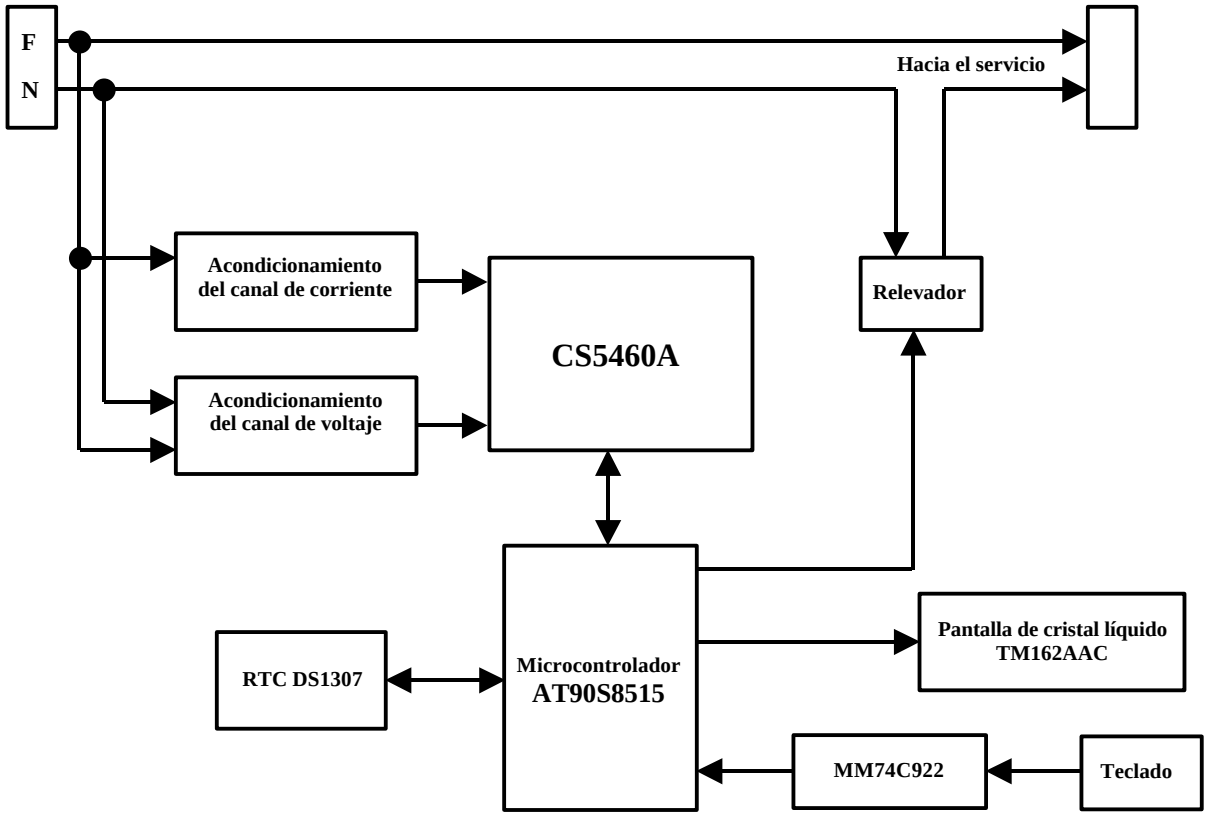


Figura 4.1. Detalle del diagrama a bloques del medidor eléctrico digital de prepago.

4.1 Acondicionamiento del canal de corriente

La señal de corriente se obtiene de la línea de alimentación mediante un transformador de corriente. Un transformador de corriente está conformado por un embobinado de alambre (alambre enrollado con un espacio circular en el centro) por cuyo centro pasa un conductor através del cual fluye una corriente alterna, que genera un campo magnético pulsante y éste a su vez, induce un flujo de corriente en el embobinado del transformador, el cual es proporcional a la corriente que fluye por el conductor. En la figura 4.2 se muestra el transformador de corriente que se utilizará.

Figura 4.2. Transformador de corriente.

La corriente que se induce en el transformador de corriente se debe convertir en un voltaje para que la señal sea procesada por el CS54060A, el cual tiene una entrada diferencial que opera en el intervalo de ±250mVDC. o 250mVAC. Para tal efecto se diseño un circuito que convierte esta señal en voltaje y que se muestra en la figura 9.
El circuito de la figura 4.3 muestra el transformador de corriente conectado a un arreglo de resistencias, de tal forma que una corriente de 4.5 amperes que pase a través del transformador generará un voltaje de ±150mV en la entrada del canal de corriente del CS5460A.

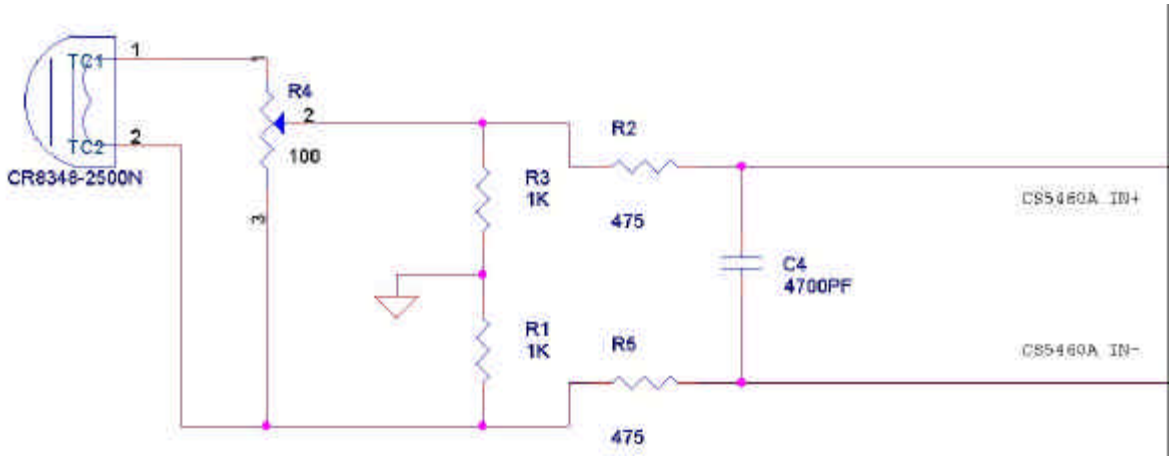


Figura 4.3. Circuito acondicionador del canal de corriente.

4.2 Acondicionamiento del canal de voltaje

La señal de voltaje se obtiene de la línea de alimentación mediante un transformador de voltaje cuya relación de transformación es de 10:1 es decir, si el primario del transformador es alimentado con 120 VAC, en el secundario habrá 12 VAC. Esta señal debe ser acondicionada por un arreglo de resistencias para poder ser procesada por el CS5460A, de tal forma que se diseñó el circuito de la figura 4.4 para que un voltaje de la línea de 180 VAC se convierta en un voltaje de ± 150 mV a la entrada del canal de voltaje del CS5460A.

Como se describió anteriormente, las señales provenientes del canal de corriente y de

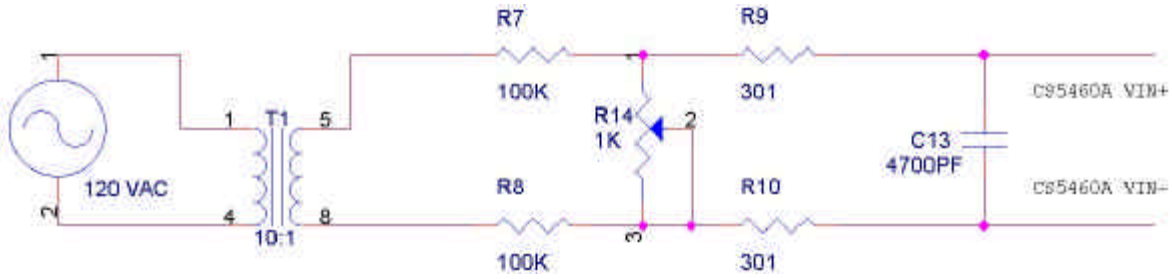


Figura 4.4. Circuito acondicionador del canal de voltaje.

voltaje se deben acondicionar antes de ser empleadas por el CS5460A. Como se explicó con anterioridad, el CS5460A puede procesar señales con amplitudes de hasta ± 250 mV. Sin embargo, nótese que los circuitos de acondicionamiento están diseñados para convertir señales de 4.5 A y 180 V en señales de 150 mV, debido a que dichos valores se consideraron como la corriente y voltaje máximos con los que puede funcionar el sistema, es decir, estos valores representan el 60% de la escala máxima a la que puede operar el CS5460A, lo cual permite que el sistema pueda funcionar con corrientes y voltajes hasta un 40% mayores de lo especificado. Por lo tanto, el medidor puede procesar voltajes de hasta 300 volts y corrientes de hasta 7.5 amperes.

4.3 Procesador de voltaje, corriente y energía CS5460A

El procesador empleado es el CS54060A de Cirrus Logic [CL: 2002], el cual es un convertidor analógico digital altamente integrado (ADC) que combina dos convertidores ΣΔ, un amplificador de ganancia programable, funciones de alta velocidad para el cálculo de energía y una interfaz serie en un solo circuito integrado. Está diseñado para medir y calcular exactamente: energía, potencia instantánea, corriente RMS (I_{RMS}) y voltaje RMS(V_{RMS}) para aplicaciones de medición de energía monofásicas de dos o tres conductores. En CD se encuentra información más detallada del funcionamiento de este circuito.

En la figura 4.5 se muestra el diagrama a bloques del circuito integrado.

4.3.1 Operación del CS5460A

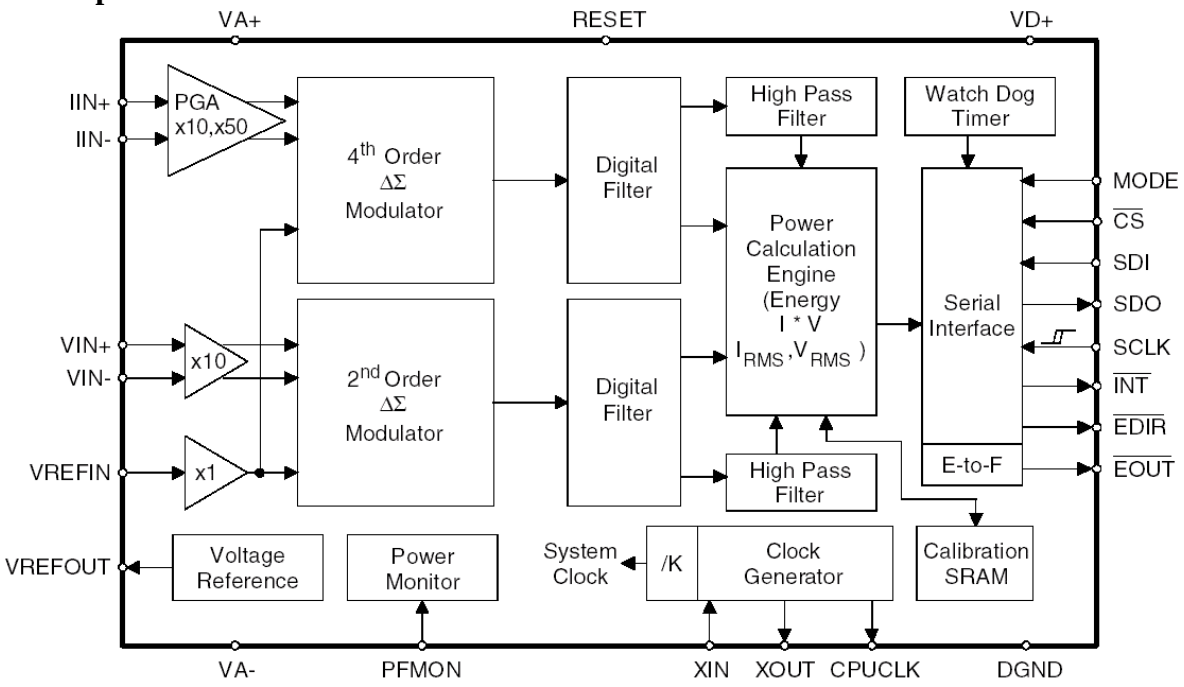


Figura 4.5. Diagrama a bloques del CS5460A.

En la figura 4.6 se muestra el diagrama de flujo que describe el procesamiento de los datos en los canales de voltaje y corriente.

Las señales analógicas en las entradas de los canales de voltaje y corriente son modificadas por los amplificadores de ganancia programable y después son muestreadas por los moduladores delta-sigma. A continuación, los datos son procesados por un filtro pasa bajas para eliminar cualquier ruido de alta frecuencia que se haya podido generar a la salida del modulador. Haciendo referencia a la figura 4.6, nótese que los datos digitales en el canal de voltaje son modificados por un filtro de retardo variable. La cantidad de retraso depende del valor de siete bits de compensación de fase y pueden ser programados por el usuario para compensar el retraso que generan los sensores de corriente (transformadores de corriente).

Los datos de ambos canales pasan entonces por dos filtros digitales de compensación tipo FIR (respuesta finita al impulso) cuyo propósito es compensar cambios en la magnitud provocados por la operación del filtro pasa bajas en el paso anterior.
Ambos canales poseen filtros pasa altas adicionales (HPF) que se pueden habilitar para eliminar cualquier contenido de DC (corriente directa) que pudiese encontrarse en los canales de voltaje y corriente antes de realizar los cálculos de energía y valores RMS. También se dispone de dos filtros pasa todo (APF), los cuales se emplean en caso de que sólo se active uno de los filtros pasa altas en cualquier canal para mantener la sincronía entre ambos canales.

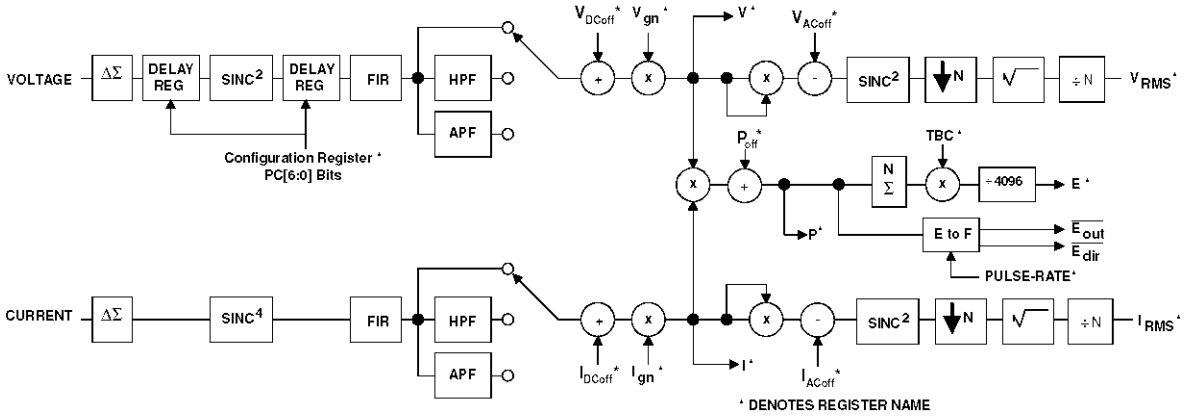


Figura 4.6. Diagrama a flujo del CS5460A.

Después del filtrado, los códigos digitales de voltaje y corriente instantáneos pasan por una etapa de ajuste de nivel/ganancia que depende del valor de los registros de nivel y de ganancia. Estos registros se emplean para calibrar el dispositivo.

Después de los ajustes de nivel y ganancia, las muestras de datos instantáneos de 24 bits se almacenan en los registros de voltaje y corriente instantáneos, desde los cuales el usuario puede leer dichos datos a través de la interfaz serie.

Los datos instantáneos de corriente y voltaje son multiplicados para formar muestras instantáneas de potencia real.

Las muestras de potencia real instantáneas son agrupadas en conjuntos de **N** muestras. La suma acumulada de estas muestras se emplea para calcular el resultado almacenado en el Registro de Energía, el cual es proporcional a la cantidad de energía real suministrada por el dispositivo. Los resultados de voltaje y corriente RMS se calculan empleando las últimas muestras instantáneas de voltaje y corriente, y los resultados pueden ser leídos desde los registros de corriente y voltaje RMS a través de la interfaz serie.

El CS5460A convierte las mediciones a un formato de datos de 24 bits con y sin signo que representan un porcentaje de la escala completa. Esto significa que las palabras de datos de 24 bits en los registros de salida del CS5460A representan valores entre 0 y 1 (para registros sin signo) o entre –1 y +1 (para registros con signo). Un valor de uno en cualquier registro representa el máximo valor posible.

Mediante la interfaz serie se pueden enviar comandos al CS5460A y también se puede acceder a los registros internos, algunos de estos registros son de solo lectura como es el caso de aquellos que contienen los resultados de los cálculos realizados por el dispositivo.

En el caso de los registros de configuración del CS5460A, es necesario escribir datos en ellos para que el dispositivo funcione acorde con los requerimientos de la aplicación en la cual se empleará el circuito.

En la figura 4.7 se muestran los registros internos del CS5460A.

Como una opción alterna a la lectura del registro de energía, el CS5460A cuenta con las terminales /EOUT y /EDIR que conforman una interfaz simple mediante la cual se puede acumular energía (positiva o negativa). Cada pulso en la terminal /EOUT representa una cantidad predeterminada de energía. La cantidad de energía representada en un pulso se puede ajustar programando dicho valor en el registro de pulsos. Los pulsos correspondientes a la terminal /EDIR indican el signo de la energía y por lo tanto, determinan si aumenta o disminuye la energía total acumulada.

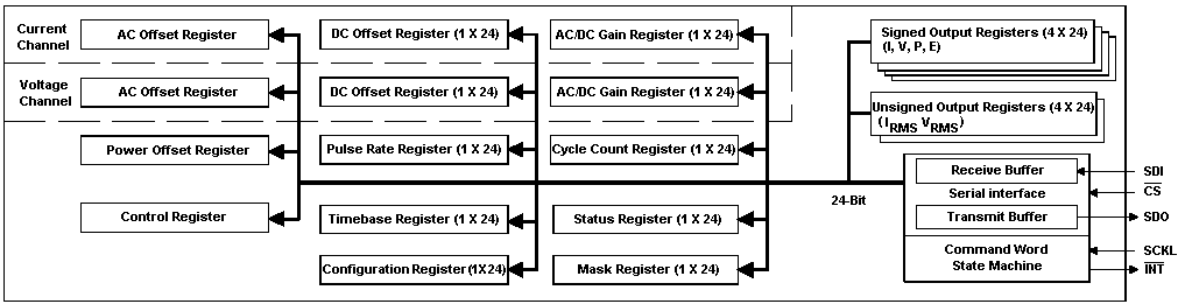


Figura 4.7. Registros internos del CS5460A.

El CS5460A cuenta también con varios registros de configuración que le permiten variar el modo de operación del dispositivo.

Otras características adicionales son un Watchdog Timer, una terminal PFMON que genera una interrupción en el caso de que la energía del sistema falle y una terminal /INT de interrupción que sirve para indicar que ha ocurrido un evento dentro del dispositivo que requiere atención.

4.3.2 Configuración del CS5460A en el sistema

En la figura 4.8 se muestran las terminales del CS5460A y a continuación se describe su interconexión con los demás dispositivos del sistema:

- Las terminales /CS, SDI, SDO, SCKL conforman la interfaz serie del dispositivo y se conectan a las terminales /SS, MOSI, MISO y SCK, respectivamente, en el microcontrolador para poder establecer la comunicación serie entre ambos dispositivos.
- La terminal MODE se conecta a tierra para indicar que el dispositivo está operando en el modo de microcontrolador.
- Las terminales /EDIR y /EOUT están conectadas a las terminales INT0 y PB3 del microcontrolador, respectivamente.
- Las terminales IIN+ e IIN- se conectan al circuito acondicionador de corriente.
- Las terminales VIN+ y VIN- se conectan al circuito acondicionador de voltaje.
- La terminal de /RESET se mantiene a un nivel alto y está conectada a la terminal PB1 del microcontrolador para poder forzar un reset por hardware desde el microcontrolador.
- La terminal PFMON se conecta mediante un divisor de voltaje a la alimentación del sistema para indicar un eventual corte en la energía de alimentación.
- La terminal /INT se conecta a la terminal PB0 del microcontrolador para poder monitorear el estado del CS5460A desde el microcontrolador.

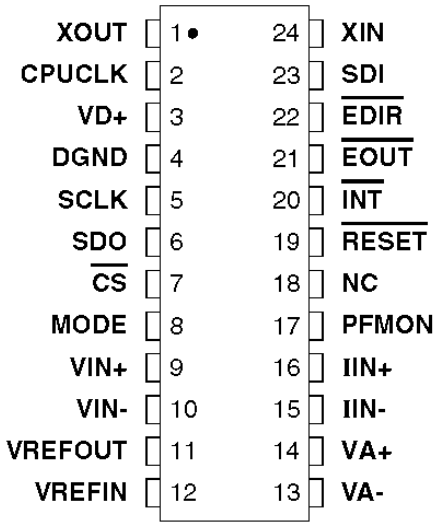


Figura 4.8. Terminales del CS5460A.

4.4 Microcontrolador AT90S8515

El dispositivo seleccionado para controlar el funcionamiento de todas las partes que conforman al medidor es el microcontrolador AT90S8515 de la compañía ATMEL [AT: 2002], el cual es un microcontrolador con arquitectura RISC (Reduced Instruction Set Computer). A continuación se da una breve descripción de este dispositivo.

El AT90S8515 es un microcontrolador CMOS de 8 bits de bajo consumo de energía, con una arquitectura del tipo RISC que le permite ejecutar instrucciones en un ciclo de reloj. En la figura 4.9 se muestra el diagrama a bloques de este dispositivo.

El AT90S8515 cuenta con 32 registros de propósito general que están directamente conectados a la unidad aritmético lógica (ALU), permitiendo el acceso a dos registros independientes en una sola instrucción que se ejecuta en un ciclo de reloj. La arquitectura resultante es más eficiente en cuanto al código empleado y alcanza un desempeño hasta diez veces mayor que los microcontroladores convencionales CISC (Compact Instruction Set Computer) [AT: 2002].

Este microcontrolador incluye además, las siguientes características:

- 8K bytes de memoria FLASH programable en el sistema (ISP)
- Memoria EEPROM de 512 bytes
- SRAM (RAM estática) de 512 bytes
- 32 líneas de entrada/salida de propósito general
- 32 registros de propósito general
- 2 contadores con modos de comparación
- Interrupciones internas y externas
- Una UART (Universal Asyncronous Receiver Transceiver) programable
- Watchdog Timer programable con oscilador interno
- Una interfaz serie del tipo SPI compatible con Microwire
- Dos modos de ahorro de energía que pueden ser seleccionados por software

Seis de los 32 registros pueden emplearse como tres registros apuntadores indirectos de direcciones de 16 bits, para direccionar el espacio de datos. Estos registros de 16 bits son el registro X, el registro Y y el registro Z.

La unidad aritmético lógica soporta operaciones aritméticas y lógicas entre registros o entre constantes y registros. Las operaciones de un solo registro también se ejecutan en la ALU. La figura 4.10 muestra la arquitectura del microcontrolador. Además de la operación de registros, los modos de direccionamiento convencionales se pueden emplear también en el espacio de registros. Esta capacidad se permite ya que el espacio de registros tiene asignadas las 32 direcciones más bajas del espacio de datos ($00 – $1F), permitiendo el acceso de estas localidades como si fueran direcciones de memoria ordinarias.

El espacio de memoria de entrada/salida (I/O) contiene 64 direcciones para las funciones de los periféricos del CPU (unidad central de procesamiento), como lo son los registros de

control, los contadores y otras funciones de entrada/salida. La memoria de entrada/salida se puede acceder directamente al igual que el espacio de datos inmediatamente después del espacio de registros en las localidades de memoria $20 – $5F.

El microcontrolador emplea el concepto de la arquitectura Harvard con buses y memorias de datos y programa separados. La memoria de programa se ejecuta en un pipeline de dos etapas. Mientras una instrucción está siendo ejecutada, la siguiente instrucción se precarga desde la memoria de programa. Este concepto permite que las instrucciones se ejecuten en cada ciclo de reloj.

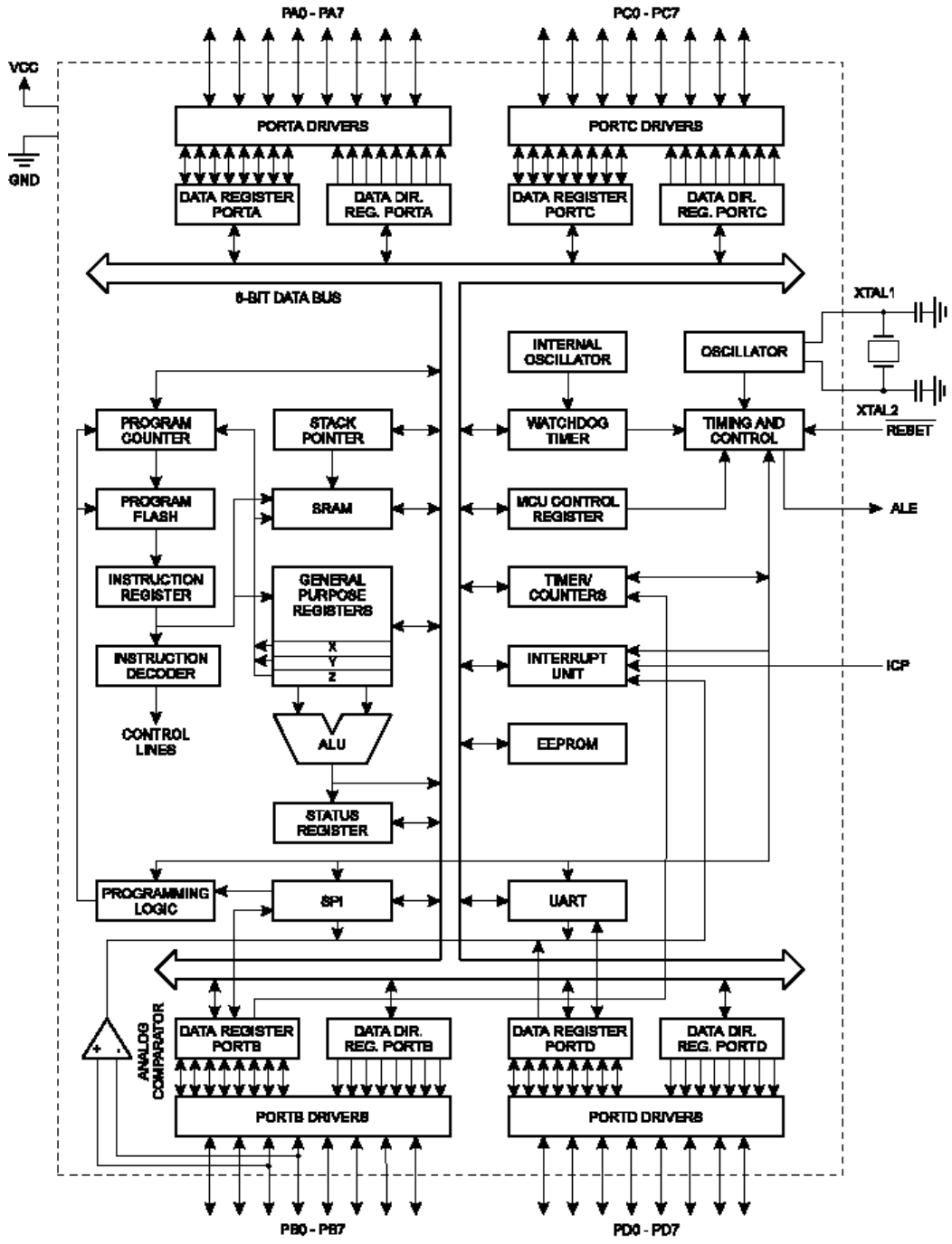


Figura 4.9. Diagrama a bloques del microcontrolador AT90S8515.

Con las instrucciones de salto y llamada relativas (RJMP RCALL) se puede direccionar de forma inmediata todo el espacio de direcciones de 4Kbytes.

Durante las interrupciones y las llamadas a subrutinas, el contador de la dirección de regreso de programa (PC) es almacenado en la pila. La pila está situada en la SRAM de datos generales y por lo tanto el tamaño de la pila sólo está limitado por el tamaño total de la SRAM y el uso de la misma.

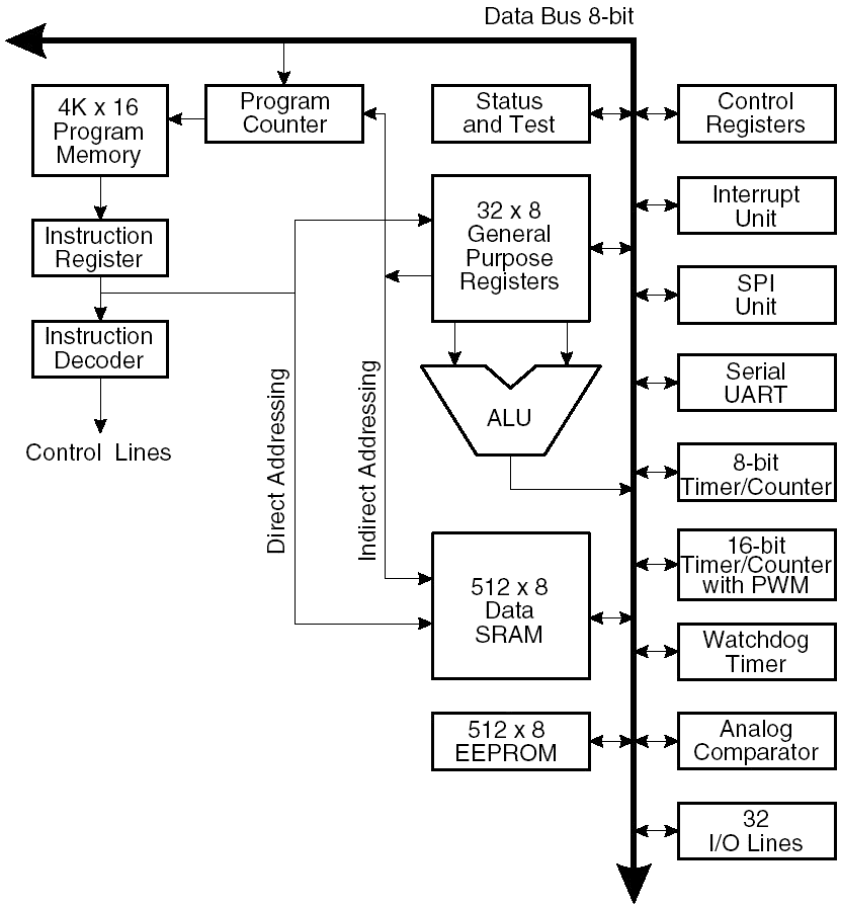


Figura 4.10. Arquitectura interna del microcontrolador AT90S8515.

Los 512 bytes de memoria SRAM se pueden acceder utilizando cualquiera de los cinco diferentes modos de direccionamiento con que cuenta el microcontrolador: directo, indirecto con desplazamiento, indirecto, indirecto con predecremento e indirecto con preincremento.

Las 32 líneas de entrada/salida del microcontrolador se dividen en 4 puertos de 8 bits, el puerto A (PA0 –PA7), el puerto B (PB0 –PB7), el puerto C (PC0 –PC7) y el puerto D (PD0 –PD7). Cada línea se puede programar independientemente como entrada o salida y algunas de estas líneas tienen funciones adicionales como es el caso del puerto SPI (dentro del puerto B) y la UART (en el puerto D), entre otras.

El módulo de interrupciones tiene sus registros de control en el espacio de entrada/salida con un bit de habilitación de interrupción global en el registro de estado. Todas las interrupciones tienen vectores de interrupción separados en la tabla de vectores de interrupción al principio de la memoria de programa.

4.4.1 Configuración del AT90S8515 en el sistema

En la figura 4.11 se muestran las terminales del AT90S8515 y a continuación se describe su interconexión con los demás dispositivos del sistema:

- Las terminales del puerto B /SS, MISO, MOSI, SCKL, PB0 y PB1 están conectadas al CS5460A para poder establecer la comunicación bidireccional y tener el control de este dispositivo.
- Las terminales del puerto D PD0 y PD1 están conectadas al circuito MAX232 para así contar con una interfaz RS232 opcional (no se utiliza en este trabajo). La terminal PD2 está conectada a la terminal /EOUT del CS5460A, ya que se emplea para generar la interrupción externa 0. La terminal PD3 se conecta a la terminal DAVAL del codificador MM74C922 y se emplea para generar la interrupción externa 1. Las terminales PD4 y PD5 están conectadas a las terminales SCL y SDA del reloj de tiempo real, respectivamente, para establecer la comunicación con este dispositivo usando una interfaz I^2C.
- Las terminales del puerto C PC0 a PC4 están conectadas respectivamente, a las terminales DOUTA, DOUTB, DOUTC, DOUTD y /OE para controlar la lectura de teclas codificadas por el MM74C922.
- La terminal PC6 se emplea para controlar el circuito del relevador para la desconexión y conexión del servicio de energía al usuario.
- Las terminales del puerto A PA0 a PA6 están conectadas a las terminales de la pantalla de cristal líquido DB4, DB5, DB6, DB7, E, R/W y RS respectivamente, para controlar la operación y el despliegue de mensajes en la pantalla.
- La terminal /RESET se mantiene conectada a un nivel alto mediante un circuito de reset que cuenta con un botón del tipo PUSHBUTTON para forzar un reset por hardware cuando este botón es oprimido.
- Las terminales MOSI, MISO, SCK y /RESET están conectadas también a un circuito que permite la programación del microcontrolador en el sistema (ISP).

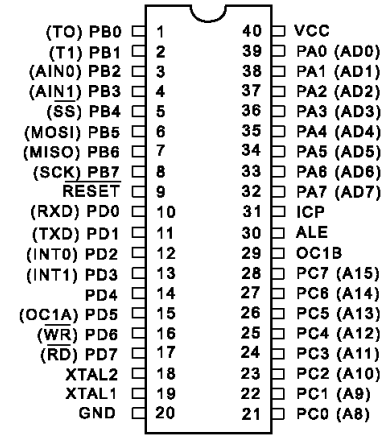


Figura 4.11. Terminales del microcontrolador AT90S8515.

4.5 Codificador de teclas MM74C922

El MM74C922 es un codificador de 16 teclas del tipo CMOS [FS: 2001] que contiene toda la lógica necesaria para codificar un arreglo de 16 interruptores SPST (Single Pole Single Trigger). Los circuitos internos del codificador requieren únicamente de un capacitor para eliminar los efectos de rebote que se generan al oprimir y soltar una tecla.

La tecla oprimida es codificada y traducida a un formato binario de 4 bits que está disponible para ser leído en las terminales DATA OUT A – DATA OUT D. El codificador indica que una tecla fue oprimida poniendo a nivel alto la terminal DATA AVAILABLE, sin embargo, para poder hacer la lectura de la tecla codificada se debe poner a nivel bajo la terminal /OUTPUT ENABLE, ya que las terminales donde se encuentran los datos están en alta impedancia.

La figura 4.12 muestra las terminales del codificador.

La conexión de este dispositivo con el microcontrolador ya fue descrita anteriormente. La conexión con el teclado de matriz se muestra en los diagramas eléctricos que se verán más delante.

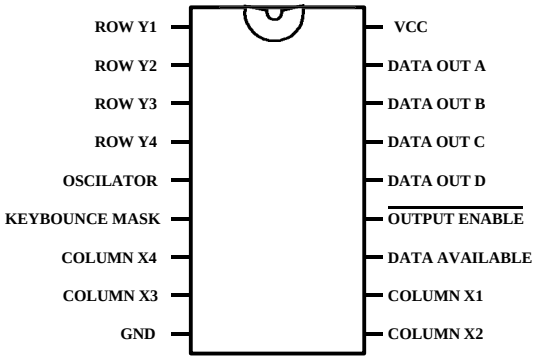


Figura 4.12. Terminales del MM74C922.

4.6 Pantalla de cristal líquido TM16AAC

La pantalla de cristal líquido TM16AAC [TA: 2001] está controlada por el circuito integrado HD44780 de la compañía Hitachi. Este dispositivo debe ser controlado por el microcontrolador para poder así acceder a las diferentes funciones de despliegue y poder mostrar los mensajes del usuario en la pantalla.

En la tabla 1 se muestran las terminales de la pantalla de cristal líquido.

Como se explicó anteriormente, la pantalla de cristal líquido está conectada al puerto A del AT90S8515, de tal forma que todas las funciones de la pantalla son controladas vía software.

Como puede observarse en la tabla 1, la pantalla de cristal líquido tiene un bus de datos de 8 bits, pero puede funcionar empleando los 4 bits más significativos, con lo que se reduce el número de líneas necesarias para controlar el dispositivo. Además del bus de datos se necesitan 3 líneas más de control; éstas son RS, /RW y E.

La pantalla de cristal líquido puede desplegar caracteres almacenados en una tabla de datos dentro del HD44780. Para poder mostrar en pantalla cualquier carácter almacenado en esta tabla, se debe escribir en el bus de datos la dirección que ocupa dicho carácter en la tabla, que para fines prácticos corresponde exactamente al código ASCII (American Standard Code for International Interchange). Por lo tanto, no es necesario conocer la dirección de cada carácter, sólo basta conocer su código ASCII y enviarlo al controlador de la pantalla para poder mostrarlo.

Tabla 4.1 Terminales de la pantalla de cristal líquido.

Número de terminal	Nombre	Función
1	**Vss**	**Tierra**
2	**Vdd**	**Alimentación +Ve**
3	**Vee**	**Contraste**
4	**RS**	**Register Select**
5	**R/W**	**Read/Write**
6	**E**	**Enable**
7	**D0**	**Data bit 0**
8	**D1**	**Data bit 1**
9	**D2**	**Data bit 2**
10	**D3**	**Data bit 3**
11	**D4**	**Data bit 4**
12	**D5**	**Data bit 5**
13	**D6**	**Data bit 6**
14	**D7**	**Data bit 7**

4.7 Implementación del medidor

Para efecto de pruebas y seguridad, debido a que el medidor opera con voltajes de hasta 300 Volts y corrientes de hasta 7.5 Amperes, se implemento el medidor en dos tarjetas separadas.

En la figura 4.13 se muestra el diseño en Orcad Layout de la primera tarjeta en una sola cara, que contiene los circuitos y elementos necesarios para el funcionamiento del CS5460A y su interconexión con la segunda tarjeta mediante un conector de diez líneas.

Esta tarjeta contiene los siguientes dispositivos:

- Transformador de corriente
- Transformador de voltaje
- Circuito integrado CS5460A
- Fuente de voltaje integrada que se energiza desde la línea de alimentación de 120 VAC
- Acondicionador para el canal de voltaje
- Acondicionador para el canal de corriente
- Conector tipo “poste” (header) para la conexión con la segunda tarjeta

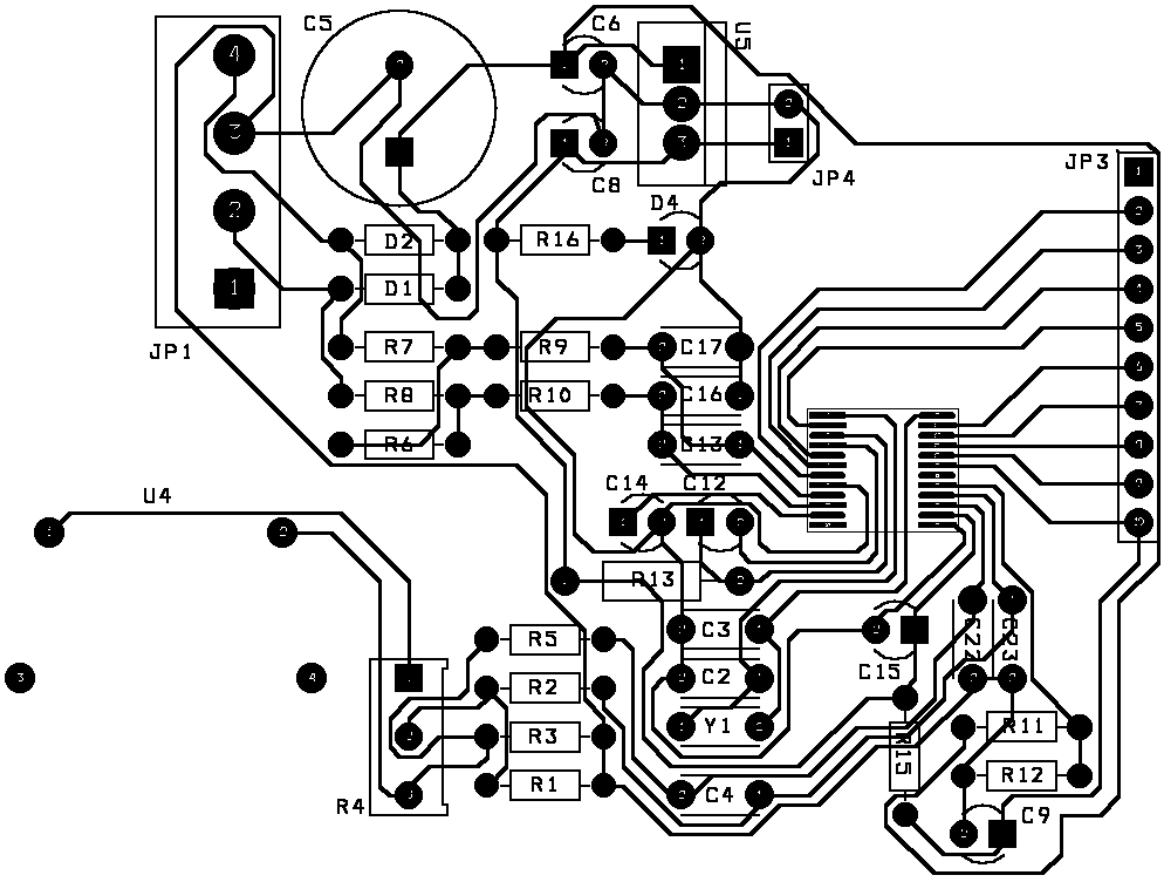


Figura 4.13. Circuito impreso de la tarjeta que contiene el CS5460A.

En la figura 4.14 se muestra la tarjeta terminada con sus respectivos componentes por la cara superior. El CS5460A se encuentra en la cara inferior.
El diagrama eléctrico de esta tarjeta se encuentra en el anexo A.

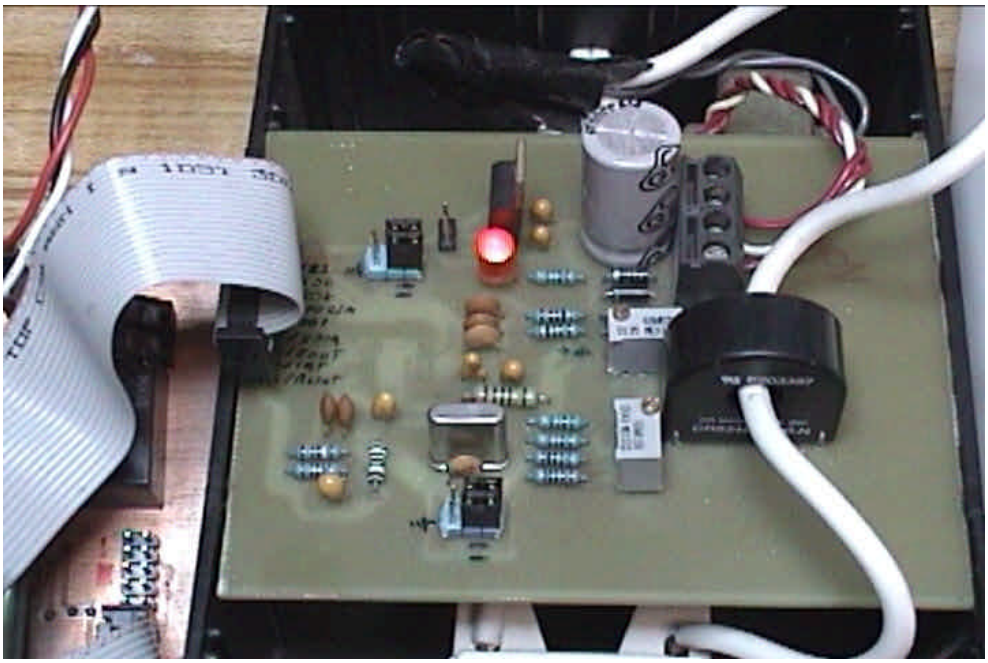


Figura 4.14. Tarjeta terminada que contiene el CS5460A.

La segunda tarjeta contiene los demás componentes que conforman al medidor. Esta tarjeta se diseñó en dos caras para ahorrar espacio. Las interconexiones entre ambas capas fueron hechas mediante through hole. El diseño en Orcad Layout de esta tarjeta se muestra en la figura 4.15.

Esta tarjeta contiene los siguientes dispositivos:

- Microcontrolador AT90S8515
- Reloj de tiempo real DS1307
- Codificador de teclado MM74C922
- Relevador para la conexión/desconexión del servicio
- Circuito regulador de voltaje para conectar a una fuente de alimentación de corriente directa o a la alimentación de la tarjeta que contiene al sensor.
- Pantalla de cristal líquido
- Circuito de reset por hardware
- Circuito y conector RS232 opcional

En la figura 4.16 se muestra la tarjeta terminada con sus respectivos componentes por la cara superior.

El diagrama eléctrico de esta tarjeta se encuentra en el anexo A.

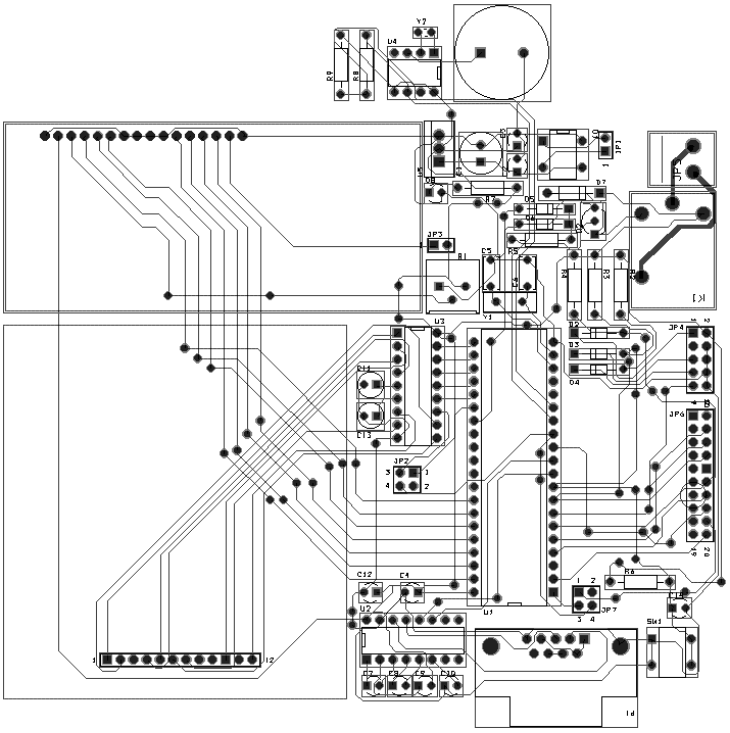


Figura 4.15. Diseño en Orcad Layout de la segunda tarjeta.

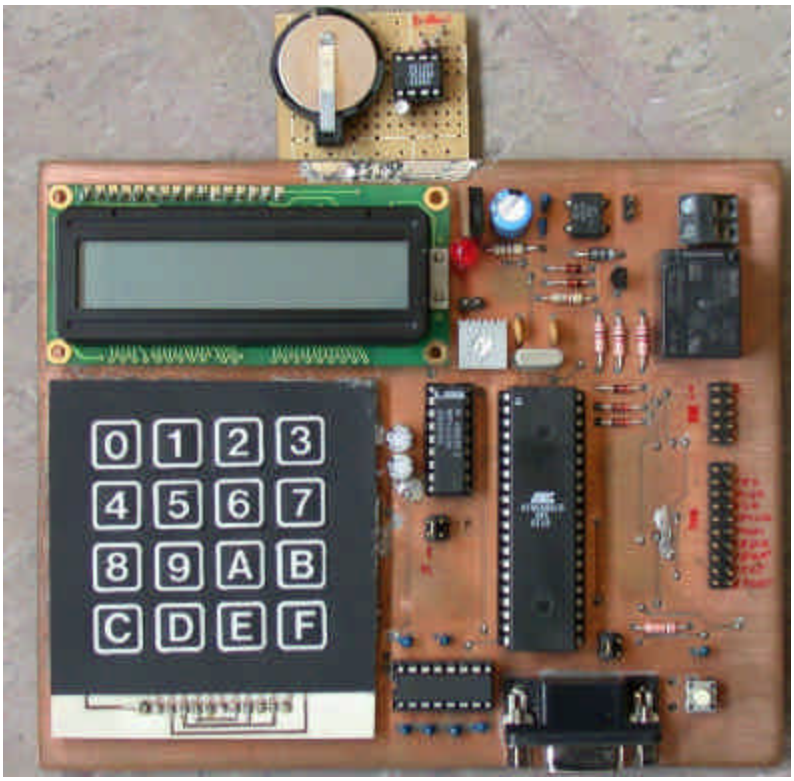


Figura 4.16. Imagen de la segunda tarjeta con sus respectivos componentes.

4.8 Resumen

En este capítulo se describieron los componentes que forman parte del medidor y se explicó el funcionamiento de los mismos. Después se explicó la implementación en hardware de las dos tarjetas que conforman al medidor. En el capítulo siguiente se hablará de la implementación del software necesario para la operación del medidor y del programa de computadora necesario para encriptar datos de 20 dígitos.

CAPÍTULO 5

DISEÑO DEL SOFTWARE DEL MEDIDOR DIGITAL DE PREPAGO

En este capítulo se describe el funcionamiento del software necesario para la operación del medidor, el cual se divide en dos partes. La primera es el programa que reside en el microcontrolador y el cual le permite controlar todas las funciones del medidor. La segunda parte comprende un programa en C que se ejecuta en una computadora personal para generar y encriptar números de 20 dígitos que requiere el medidor para abonar crédito.

5.1 Software del microcontrolador AT90S8515

Para el diseño del software del medidor se empleó el entorno de desarrollo de ATMEL conocido como AVR Studio. Este programa funciona en conjunto con el emulador ICE200 también de ATMEL para diseñar y comprobar el funcionamiento del software en el medidor sin necesidad de programar al microcontrolador ya que el ICE200 emula el funcionamiento de un microcontrolador real montado en la tarjeta del medidor. La ventaja de emplear un emulador es que éste permite la ejecución de código paso a paso, a la vez que se puede observar el estado y el contenido de los diferentes registros y puertos del microcontrolador, entre otras características que permiten la depuración del código para obtener un programa funcional y libre de errores.

La programación se realizó en el lenguaje ensamblador, propio del microcontrolador, empleando el AVR Studio. De manera global el programa se divide en subrutinas, rutinas de servicio de interrupciones y un programa principal. El programa principal es el encargado de determinar el modo de operación del medidor y mostrar en la pantalla la información correspondiente a cada uno de estos modos, el diagrama de flujo del programa principal se muestra en la figura 5.1.

5.1.1 Modos de operación

- Modo normal: despliega en la pantalla de cristal líquido la fecha y hora, así como el crédito disponible. En caso de que se hayan introducido tres códigos inválidos consecutivos para abonar crédito, se mostrará el mensaje de “TECLADO INACTIVO” en lugar de mostrar el crédito disponible, provocando la inhabilitación automática del teclado por un periodo de 12 horas.
- Modo de entrada de datos: en este modo se muestran en la pantalla de cristal líquido los números introducidos por el usuario, con el fin de abonar crédito al medidor, los cuales se mostrarán por un periodo de tiempo definido y reprogramable.

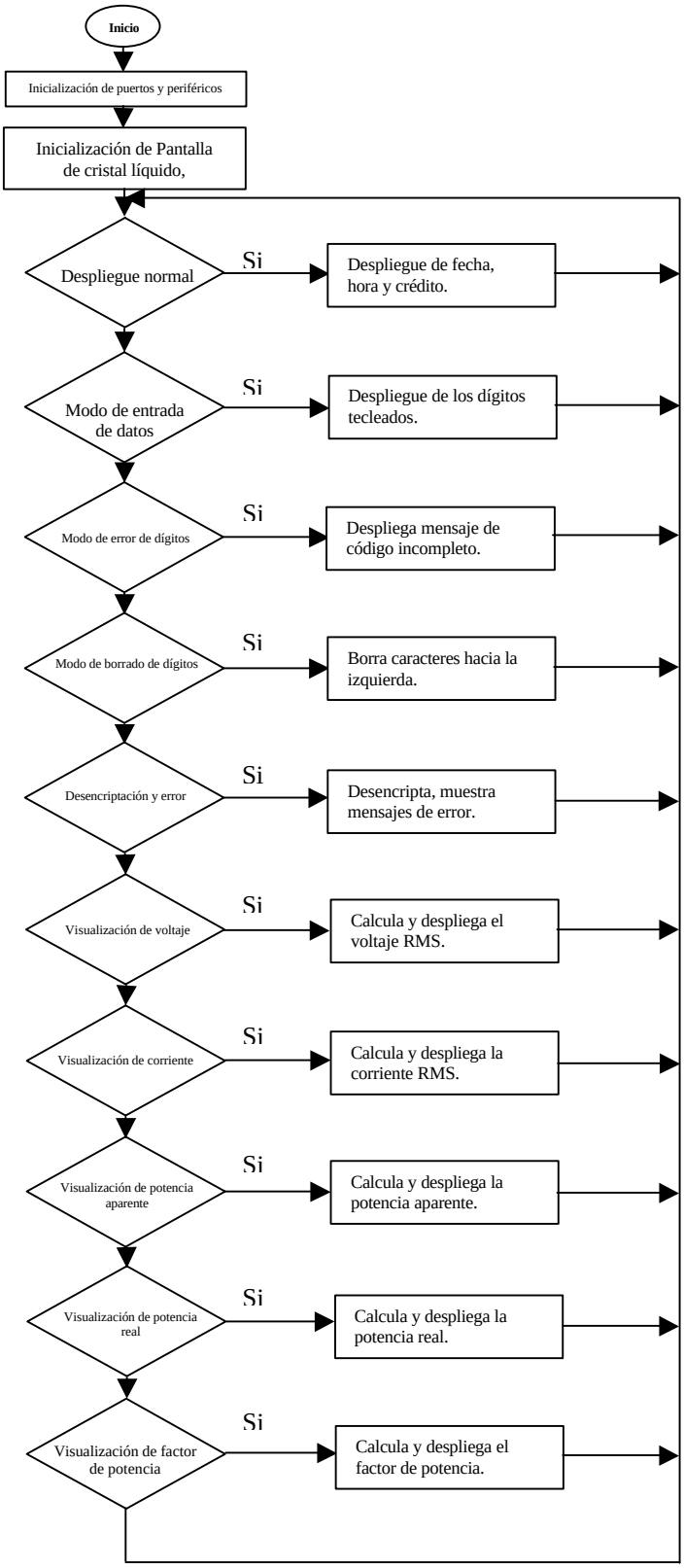


Figura 5.1. Diagrama de flujo del programa principal

- Modo de visualización de voltaje: en este modo se muestra en la pantalla de cristal líquido el promedio del voltaje registrado en el segundo anterior. Modo de visualización de corriente: en este modo se muestra en la pantalla de cristal líquido el promedio de la corriente registrada durante el segundo anterior.
- Modo de visualización de potencia aparente: en este modo se muestra en la pantalla de cristal líquido el promedio de la potencia aparente registrada durante el segundo anterior.
- Modo de visualización de potencia real: en este modo se muestra en la pantalla de cristal líquido el promedio de la potencia aparente registrada durante el segundo anterior.
- Modo de borrado de dígitos: este modo permite borrar de la pantalla de cristal líquido los números erróneamente introducidos al teclear el código de 20 dígitos para abonar crédito.
- Modo de error de dígitos: en el caso de que el número de dígitos tecleado por el usuario para abonar crédito sea menor de 20, se mostrará en la pantalla de cristal líquido el mensaje de "CÓDIGO INCOMPLETO" y retornará al modo normal de visualización.
- Modo de de desencriptación y error de código: en el caso de que el código de 20 dígitos introducido por el usuario una vez desencriptado resulte inválido, se mostrará en la pantalla de cristal líquido el mensaje de "CÓDIGO INCORRECTO". Después de mostrar este mensaje, el medidor automáticamente comienza a contar el número de códigos incorrectos tecleados de tal forma que sólo permite la introducción de tres códigos incorrectos por día.

Al energizar el medidor, el microcontrolador ejecuta las instrucciones de inicialización de los puertos e interrupciones, carga en la memoria de datos las variables necesarias para la operación del medidor que incluyen las llaves de desencriptación, la fecha y hora que provienen del reloj de tiempo real así como los valores de crédito y Kilowatts – horas disponibles. También inicializa la pantalla de cristal líquido, escribe las constantes de calibración en el CS5460A y determina en base al crédito disponible la conexión o desconexión del relevador.

5.1.2 Interrupciones

El programa contiene tres interrupciones, la primera es una interrupción por software que se genera cada segundo y que sirve para llevar la cuenta del tiempo y actualizar las mediciones mostradas en la pantalla de cristal líquido. el diagrama de flujo de esta interrupción se muestra en la figura 5.2. Las dos interrupciones restantes son interrupciones por hardware; la primera se genera cada que se oprime una tecla. El diagrama de flujo de esta interrupción se muestra en la figura 5.3. y la segunda, cada vez que el CS5460A genera un pulso para indicar al microcontrolador que debe incrementar o decrementar la cuenta de energía registrada por el medidor. El diagrama de flujo de esta interrupción se muestra en la figura 5.4.

La interrupción TC1M corresponde a la interrupción por software que se genera aproximadamente cada segundo. Su función es la de llevar la cuenta de los segundos, minutos y horas que se muestran en la pantalla. Otra de sus funciones es mostrar la fecha en pantalla así como los cálculos de corriente, voltaje, potencia aparente, potencia real, factor de potencia, códigos introducidos para la desencriptación, y el crédito disponible, dependiendo del modo de visualización en que se encuentre el medidor, el cual está determinado por la última tecla oprimida y que se describe en la interrupción EX_INT1, la cual es la interrupción externa uno.

Otras funciones de la interrupción TC1M son:

- Llevar la cuenta del tiempo que dura la visualización en pantalla de valores como el voltaje, la corriente, la potencia aparente, la potencia real, el factor de potencia y el código tecleado por el usuario.
- Indicar al programa el almacenamiento de los valores de energía y crédito restante en el reloj de tiempo real.
- Indicar al programa la actualización de la cuenta del tiempo y la fecha desde el reloj de tiempo real para mantener la exactitud de la fecha y hora.
- Llevar la cuenta de las horas que permanecerá desactivado el teclado debido a la introducción consecutiva de tres códigos no válidos.

La interrupción EX_INT1 es una interrupción externa por hardware que se genera cada vez que el usuario oprime una tecla del medidor. Su función es la de registrar y determinar cuál tecla fue oprimida para así ejecutar las subrutinas que corresponden a dicha tecla.

La función de cada tecla se describe a continuación:

- Las teclas 0 al 9 sirven para introducir el código a ser desencriptado que permite abonar crédito al medidor. Cada que se oprime una de estas teclas la rutina de interrupción EX_INT1 almacena el valor de cada tecla oprimida en la memoria de datos, lleva la cuenta del número de dígitos tecleados e indica al programa que se debe mostrar en pantalla el o los dígitos introducidos por el usuario, los cuales no pueden ser más de 20.
- La tecla A sirve para visualizar los valores de voltaje, corriente y potencia aparente, cada vez que es oprimida. La rutina de interrupción EX_INT1 determina que valor será mostrado en pantalla dependiendo del número de veces que ha sido oprimida la tecla A; e indica a la rutina TC1M cuántos segundos se mostrará en pantalla cada valor.
- La tecla B sirve para visualizar los valores de potencia real y factor de potencia cada vez que es oprimida. La rutina de interrupción EX_INT1 opera de la misma forma que con la tecla A.
- La tecla C sirve para borrar uno a uno los dígitos tecleados por el usuario, la rutina de interrupción EX_INT1 se encarga de decrementar la cuenta de los dígitos tecleados e indica al programa que borre el dígito de la pantalla.
- La tecla D sirve para iniciar la desencriptación del código de 20 dígitos teclado por el usuario para abonar crédito al medidor. La rutina de interrupción EX_INT1 indica al programa que se debe de iniciar el proceso de desencriptación.

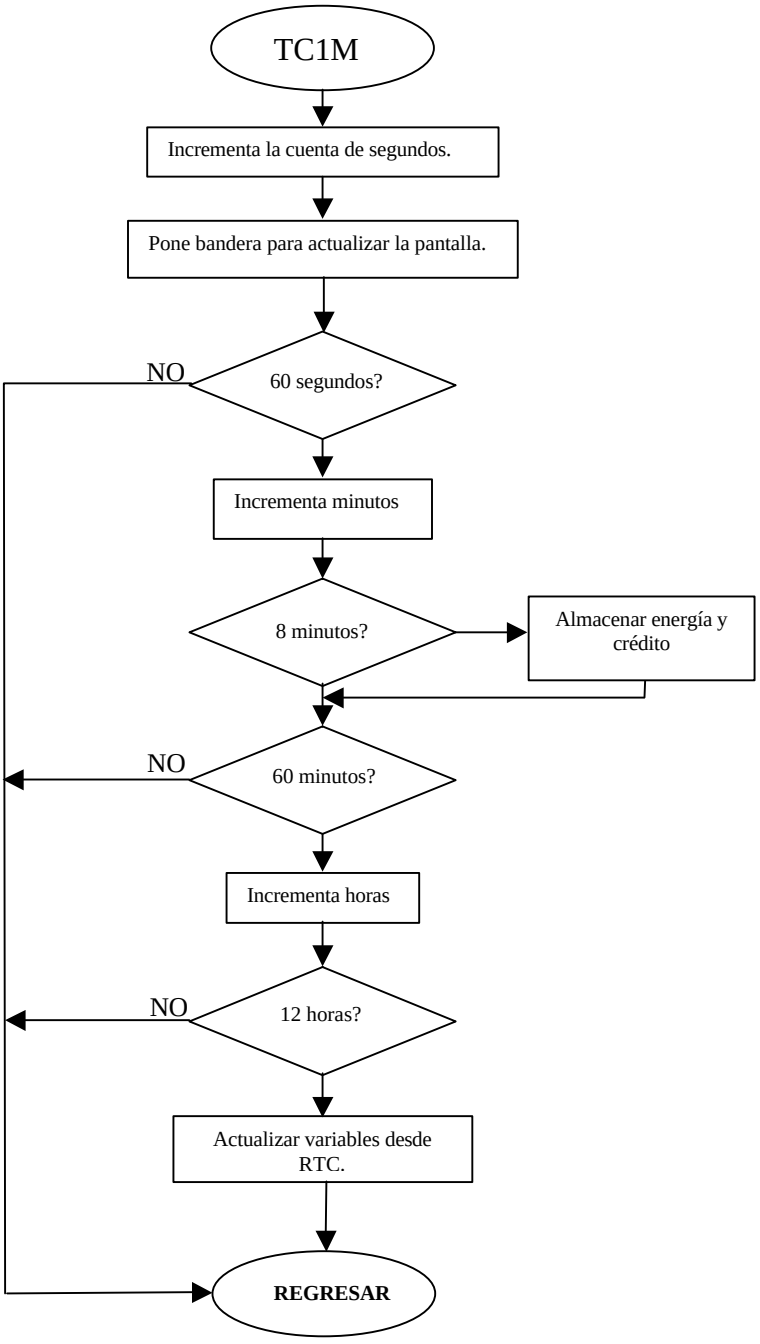


Figura 5.2. Diagrama de flujo de la interrupción TC1M.

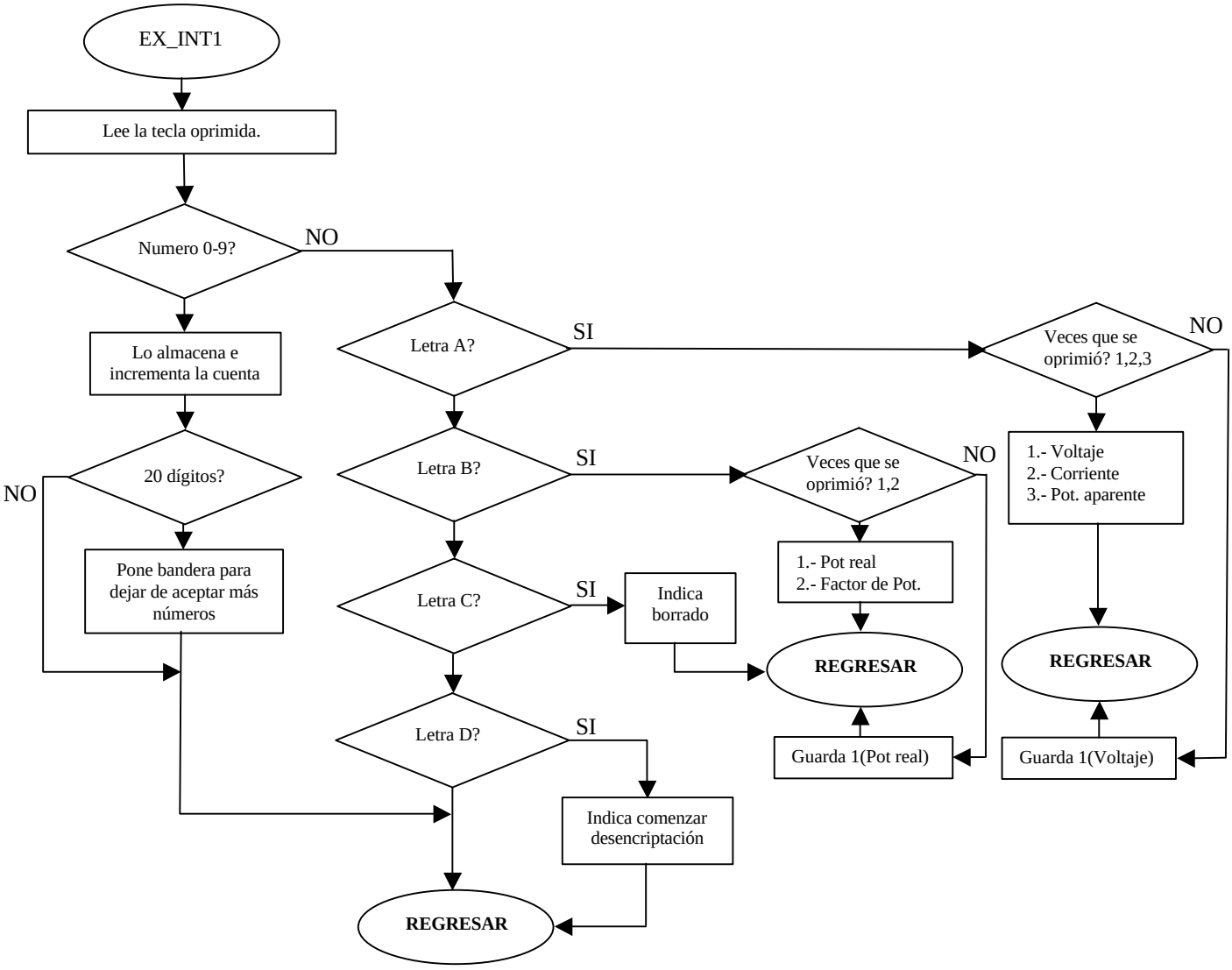


Figura 5.3. Diagrama de flujo de la interrupción EXT_INT1.

La interrupción EX_INT0 es una interrupción externa por hardware que es generada cada que el CS5460A envía un pulso al microcontrolador para indicarle que debe incrementar o decrementar la cuenta de la energía registrada por el dispositivo.

El CS5460A fue programado para generar 3600 pulsos por cada Kilowatt – hora consumido por el medidor, de tal manera que cada pulso generado por este dispositivo debe ser registrado y contado por el microcontrolador. La energía puede fluir hacia el servicio (consumida por el usuario) o desde el servicio (es el caso de cargas altamente inductivas), en cuyo caso, la cuenta general de energía que lleva el microcontrolador debe ser incrementada o decrementada según el estado de la terminal EDIR del CS5460A, que indica el signo de la energía registrada.

Para entender el funcionamiento de esta interrupción es necesario saber que cuando el medidor no cuenta con crédito, esta interrupción se encuentra deshabilitada. La existencia de crédito en el medidor implica que esta rutina se encuentra habilitada y que lleva la cuenta de los kilowatts – horas y dinero abonados al medidor. Una vez que el medidor cuente 3600 pulsos de energía positiva (consumida por el medidor), la rutina resta un kilowatt – hora de la cuenta que lleva y resta su equivalente en dinero de la cuenta que lleva del dinero abonado.
En caso de que la energía sea negativa, es decir, que la energía fluya desde la instalación del usuario hacia la red de distribución, la cuenta de los pulsos es decrementada. Cuando esta cuenta llega a ser menor que cero, la rutina incrementa en uno la cuenta de kilowatts-horas y su equivalente en dinero a la cuenta que lleva del dinero abonado, al mismo tiempo que la cuenta de los pulsos generados por el CS5460A es modificada para que contenga el valor de 3599.

Las funciones que realiza la interrupción externa EX_INT0 se describen a continuación:

- La interrupción se activa cada vez que detecta un flanco de bajada desde la terminal EOUT del CS5460A, lo cual indica al microcontrolador que debe incrementar o decrementar la cuenta de la energía registrada.
- La interrupción incrementa o decrementa la cuenta de los pulsos registrados desde el CS5460A dependiendo del estado de la terminal EDIR el cual indica si la energía fue consumida o generada por la instalación eléctrica del usuario.
- La interrupción decrementa en uno la cuenta de kilowatts – horas cuando la cuenta de pulsos desde el CS5460A llegue a 3600.
- La interrupción resta al crédito abonado el equivalente en dinero de un kilowatt –hora cada que cuente 3600 pulsos desde el CS5460A.
- La interrupción incrementa en uno la cuenta de kilowatts – horas cuando la cuenta de pulsos desde el CS5460A sea menor que cero y modifica esta cuenta para que almacene un valor de 3599 pulsos.
- La interrupción suma al crédito abonado el equivalente en dinero de un kilowatt – hora cada que la cuenta de pulsos desde el CS5460A sea menor de cero.
- La interrupción se desactiva a sí misma y al relevador cuando la cuenta de kilowatts – horas es cero (la cuenta del dinero abonado es cero también) y avisa al programa que no

hay crédito disponible para que mantenga este estado hasta que se abone crédito nuevamente.

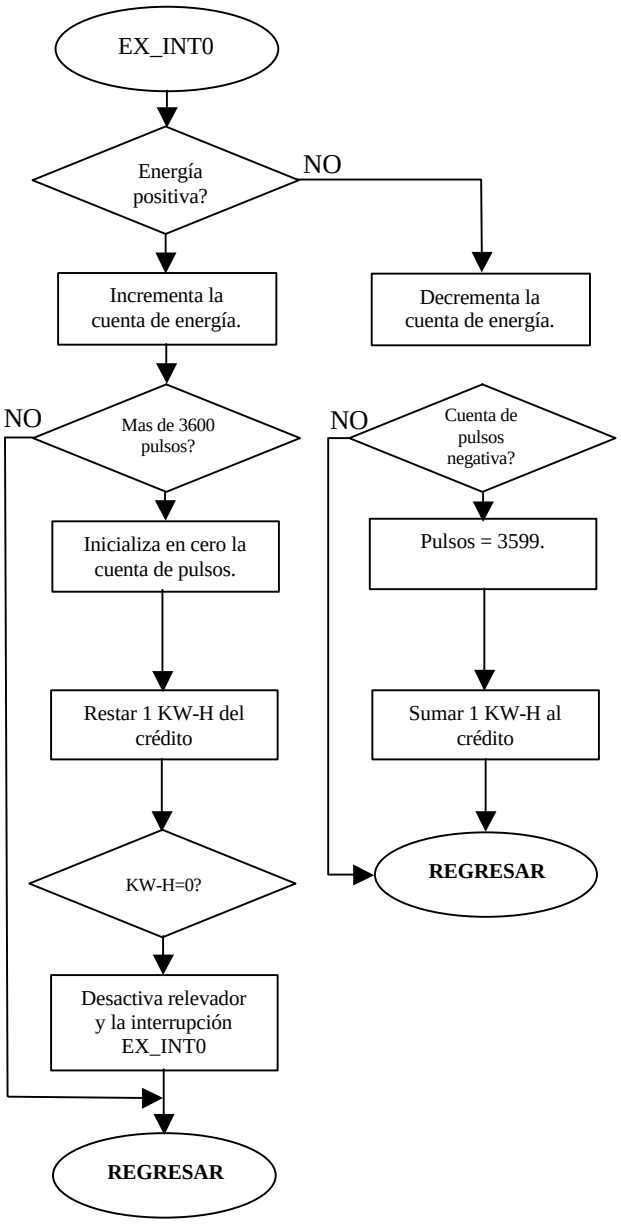


Figura 5.4. Diagrama de flujo de la interrupción EXT_INT0.

5.1.2 Subrutinas para el cálculo de variables eléctricas

5.1.2.1 Voltaje

Para mostrar en pantalla el cálculo del voltaje se siguen los siguientes pasos:

- El microcontrolador mediante el puerto SPI obtiene desde el CS5460A el voltaje registrado durante el último segundo, el cual está contenido en 24 bits que representan un valor entre cero y uno de la escala total del voltaje máximo que es 300 Volts.
- A continuación, solo se toman los 16 bits más significativos de la lectura del voltaje desde el CS5460A y se multiplican por 300 (voltaje máximo) para obtener el voltaje en binario (escalamiento).
- El siguiente paso consiste en modificar el valor binario del voltaje para poder ser mostrado en pantalla, lo cual implica convertir dicho valor a ASCII y añadir el correspondiente punto decimal, además de eliminar los ceros a la izquierda del punto decimal.

5.1.2.2 *Corriente*

Para mostrar en pantalla el cálculo de la corriente se siguen los siguientes pasos:

- El microcontrolador, mediante el puerto SPI, obtiene desde el CS5460A la corriente registrada durante el último segundo, la cual está contenida en 24 bits que representan un valor entre cero y uno de la escala total de la corriente máxima, que es 7.5 Amperes.
- A continuación, solo se toman los 16 bits más significativos de la lectura de la corriente desde el CS5460A y se multiplican por 7.5 (corriente máxima) para obtener la corriente en binario (escalamiento).
- El siguiente paso consiste en modificar el valor binario de la corriente para poder ser mostrada en pantalla, lo cual implica convertir dicho valor a ASCII y añadir el correspondiente punto decimal además de eliminar los ceros a la izquierda del punto decimal.

5.1.2.3 *Potencia aparente*

Para el cálculo de la potencia aparente se siguen los siguientes pasos:

- El microcontrolador realiza las lecturas de la corriente y el voltaje registrados durante el último segundo empleando el puerto SPI, es decir, lee 24 bits correspondientes al voltaje y 24 bits más que corresponden a la corriente.

- El microcontrolador multiplica los 24 bits de voltaje con los 24 bits de la corriente y del resultado sólo toma los 16 bits más significativos que posteriormente son multiplicados por los 16 bits más significativos del resultado de multiplicar los valores máximos de voltaje y corriente, cuyo producto es 2250 (escalamiento).

- Los 16 bits más significativos de la última operación son procesados para convertirlos a ASCII, añadir el punto decimal y eliminar los ceros a la izquierda y con ello poder mostrar la potencia aparente en la pantalla.

5.1.2.4 *Potencia real*

Para el cálculo de la potencia real se siguen los siguientes pasos:

- El microcontrolador, mediante el puerto SPI, obtiene desde el CS5460A la potencia real consumida durante el último segundo, la cual está contenida en 24 bits que representan un valor entre –1 y 1 (complemento a 2).
- Se procesan los 24 bits para obtener el signo y el valor absoluto de la potencia real.
- Los 16 bits más significativos son multiplicados por los 16 bits más significativos del resultado de multiplicar los valores máximos de voltaje y corriente, cuyo producto es 2250 (escalamiento).
- Los 16 bits más significativos de la última operación son procesados para convertirlos a ASCII, añadir el punto decimal y eliminar los ceros a la izquierda y asi poder mostrar la potencia real en la pantalla.

5.1.2.5 *Factor de potencia*

El factor de potencia es la relación entre la potencia real y la potencia aparente, de tal forma que para su cálculo se emplean las rutinas mencionadas anteriormente con el fin de obtener el valor binario de ambas variables. Después se procede a dividir la potencia real entre la potencia aparente, operación de la cual se obtiene el resultado y el residuo, que en conjunto son procesados para poder mostrar el factor de potencia en pantalla. El cálculo del factor de potencia mostrará un valor de cero en el caso de que la potencia aparente sea cero y así evitar una división entre cero. Cuando el factor de potencia sea mayor de uno, sólo se mostrará en pantalla el valor de uno ya que esta condición se presenta cuando las cargas conectadas al medidor cambian rápidamente, por lo que el valor mostrado en la pantalla del medidor será correcto una vez que las cargas conectadas al mismo se estabilicen.

5.1.3 Desencriptación

Para poder abonar crédito al medidor es necesario introducir un número de 20 dígitos previamente encriptado que contiene la información del número de serie del medidor, la cantidad de dinero que será abonada, la fecha en que fue generado el número y otros dígitos que en conjunto conforman la seguridad del medidor.

El número de serie es un campo de 9 dígitos, lo cual permite tener hasta un billón de medidores con números de serie diferentes.

El crédito a abonar es un campo de 4 dígitos, de tal forma que se puede abonar desde un peso hasta 9999 en una sola transacción.

Los 7 dígitos restantes contienen la fecha y otros dígitos necesarios para verificar la autenticidad del código de 20 dígitos y asegurar que el código generado solo será valido en una transacción.

Gran parte de la seguridad del sistema recae en el hecho de que la fecha es un valor continuo y cambiante que no puede repetirse, por lo que es imposible generar códigos iguales a partir de fechas distintas.

El proceso de desencriptación funciona de la siguiente manera:

- El proceso de desencriptación comienza una vez que el usuario ha tecleado 20 dígitos y oprimido la tecla D sucesivamente. Si el numero de dígitos es menor que 20, el medidor mostrará en pantalla el mensaje "CÓDIGO INCOMPLETO" durante 3 segundos.
- Si el numero de dígitos es correcto, el programa desencripta el código introducido por el usuario y posteriormente verifica, en primera instancia, el número de serie, la fecha y los "dígitos verificadores" (estos aumentan la seguridad del sistema). Si cualquiera de estos valores es incorrecto aparecerá en la pantalla el mensaje "CÓDIGO INCORRECTO" durante 3 segundos y se comenzará a contar el número de códigos incorrectos. Cuando el número de códigos incorrectos sea igual a 3 el medidor desactivará el teclado durante 12 horas para evitar que el usuario siga intentando introducir números al azar. El teclado permanecerá inactivo durante 12 horas de funcionamiento del medidor, es decir, si el medidor es desenergizado no contará el tiempo que permanezca inactivo.
- Como se explicó el usuario tiene 3 oportunidades de introducir un código válido. Al introducir un código válido el contador de códigos incompletos es inicializado en cero.
- Una vez que se introduce un código válido, el programa calcula en base a la tarifa el numero de Kilowatts – hora a ser abonados, así como también la cantidad de dinero correspondiente a esos Kilowatts – hora.
- En caso de que se haya abonado crédito al medidor antes de que se terminara el crédito anterior, el nuevo crédito será sumado al crédito existente.
- En caso de que se abone crédito y el medidor haya estado funcionando sin crédito, automáticamente se activarán el relevador y la interrupción externa EX_INT0 para, de esta forma, restablecer el servicio hacia el usuario.
- Después de validar o descartar un código esta rutina automáticamente almacena en la memoria no volátil del reloj de tiempo real las variables de energía y algunas banderas para asegurar que los procesos realizados no se verán afectados por fallas en la alimentación de energía al medidor.

El código introducido por el usuario fue previamente encriptado empleando el algoritmo de encriptación público RSA.
Para obtener el código original (antes de ser encriptado) se debe desencriptar dicho código empleando el mismo algoritmo con algunas variantes. Los datos necesarios para desencriptar el código introducido por el usuario son la llave pública llamada N y la llave secreta llamada D, como se describió en el capítulo 1.

La fórmula para llevar a cabo la desencriptación es la siguiente:

$$M = C^D \bmod N$$

Donde:

M es el mensaje desencriptado
C es el mensaje encriptado (código introducido por el usuario)
D es la llave secreta
N es la llave pública

Por lo que, considerando los siguientes valores:

C = 45869875698532124568
D = 05130908430267877761
N = 189765432191559752533

La fórmula puede re-escribirse como:

$$M= 45869875698532124568^{05130908430267877761} \bmod 189765432191559752533$$

Como puede observarse para obtener el valor de M se deben realizar operaciones de multiplicación y división con números enteros muy grandes, los cuales quedan fuera del alcance de varios lenguajes de programación comerciales, por lo menos de manera directa.

El microcontrolador que estamos empleando puede realizar operaciones aritméticas de suma y resta, de hasta 8 bits y los valores mostrados de C, D y N son de por lo menos 72 bits cada uno, por lo que se tuvieron que implementar rutinas de multiplicación de 72 por 72 bits y divisiones de 144 entre 72 bits aunado a un algoritmo conocido como “Russian Peasant Algorithm” [RP: 2002], el cual simplifica las operaciones de multiplicación de números grandes realizadas por el microcontrolador ya que el algoritmo emplea multiplicaciones y divisiones entre dos para obtener el producto de dos factores.

Para multiplicar dos números empleando el Russian Peasant Algorithm se procede de la siguiente manera:

- Se escribe cada numero en la parte superior de una columna
- El número escrito en la primer columna se multiplica por dos y el número en la segunda columna se divide entre dos.
- Si el número en la segunda columna es impar, se divide entre dos y se descarta el residuo.
- Si el número de la segunda columna es par, se elimina toda la fila
- Se continua el procedimiento hasta que el numero de la segunda columna sea uno.
- Por último se suman los números de la primer columna que no fueron eliminados.
- El resultado de la suma es el producto que se buscaba calcular.

A continuación se muestra como se multiplican los números 99999 y 88888 empleando el algoritmo:

~~99999~~	~~88888~~
~~199998~~	~~44444~~
~~399996~~	~~22222~~
799992	11111
1599984	5555
3199968	2777
~~6399936~~	~~1388~~
~~12799872~~	~~694~~
25599744	347
51199488	173
~~102398976~~	~~86~~
204797952	43
409595904	21
~~819191808~~	~~10~~
1638383616	5
~~3276767232~~	~~2~~
6553534464	1
8888711112	**Suma**

Como se mencionó anteriormente, la ventaja de utilizar este algoritmo reside en el hecho de que las multiplicaciones y divisiones entre dos son realizadas por el microcontrolador en binario, e implican tan solo corrimientos a la izquierda o a la derecha.

Las llaves para llevar a cabo la desencriptación están almacenadas en la memoria de programa del microcontrolador, al igual que el número de serie.

En el anexo B se encuentra el programa completo en ensamblador para el microcontrolador.

5.2 Programa para la encriptación y desencriptación de códigos de 20 dígitos.

Para la generación de las llaves de encriptación y desencriptación se diseñó un pequeño programa en C, de tal forma que este sencillo programa permite generar la llave pública N y la llave secreta D.

El funcionamiento del programa se describe a continuación:

Primero se generan 2 números primos para obtener la llave pública N.
Se determinó que el valor de la llave pública E seria 65537, ya que es un valor comúnmente empleado que no representa ningún problema de seguridad [BS: 1996].
Se emplea el algoritmo euclidiano de Euler [EA: 2002] para calcular el valor de D.
Se muestra un mensaje en pantalla para que el usuario introduzca el código a encriptar.
El código es encriptado y se muestra en pantalla el resultado.
Para verificar que el número fue encriptado correctamente se realiza la operación de desencriptación y se muestra en pantalla.
Para las operaciones de encriptación y desencriptación se utilizó el algoritmo "Russian peasant algorithm".
Este programa fue desarrollado empleando el programa Visual C++ de Microsoft y la librería de funciones de WinNTL [NT: 2002].

En el anexo C se encuentra el programa en C para encriptar códigos de hasta 20 dígitos.

5.3 Resumen

En este capítulo se describió el funcionamiento del programa que reside en el microcontrolador y que se encarga de la operación de todas las funciones del medidor. También se menciona el funcionamiento del programa que encripta los códigos de 20 dígitos con los cuales se puede abonar crédito al medidor. En el capítulo siguiente se describirán las pruebas de funcionamiento y los resultados obtenidos.

CAPÍTULO 6

PRUEBAS Y RESULTADOS

En este capítulo se describen las pruebas realizadas durante el desarrollo del medidor para verificar su funcionalidad y las mediciones realizadas para comprobar su precisión.

6.1 Pruebas de funcionamiento del medidor

Durante el desarrollo del medidor se realizaron varias pruebas. Cada una de estas pruebas se realizó para verificar el funcionamiento de las rutinas que se programaron para así avanzar en el desarrollo del medidor una vez que cada rutina fuera probada y se verificara que no tuviera errores.

A continuación se describen algunas de las pruebas realizadas en orden cronológico:

- Se probaron las rutinas que permiten controlar y enviar mensajes a la pantalla de cristal líquido.
- Se probó el funcionamiento de la rutina que procesa las teclas oprimidas en el teclado.
- Se probó la rutina que procesa la interrupción que se genera cada segundo y que lleva la cuenta del tiempo en el microcontrolador.
- Se diseñó y probó un programa para verificar la comunicación del microcontrolador con el CS5460.
- Se diseñó y probó un programa para calibrar el CS5460A.
- Se probaron las rutinas que realizan los cálculos de voltaje y corriente.
- Se probaron las rutinas que calculan y muestran la potencia aparente, la potencia real y el factor de potencia.
- Se probaron las rutinas para programar, leer y escribir datos en reloj de tiempo real.
- Se probaron las rutinas que realizan la desencriptación, las cuales incluyen rutinas de multiplicación, división, conversión de decimal a binario y viceversa.
- Se probaron las rutinas que muestran en pantalla la fecha y hora, así como el crédito disponible en el medidor.

Por último se estructuró el programa principal y fueron incorporadas las rutinas descritas anteriormente. Durante este proceso se depuró el programa en repetidas ocasiones y se obtuvieron 21 versiones del programa.

Con la última versión del programa se realizaron varias pruebas; la primera consistió en programar el reloj de tiempo real con la hora y fecha actuales y posteriormente se verificó que la cuenta de la hora y fecha que se muestra en la pantalla del medidor coincidiera con

la fecha y hora actuales. Estas comparaciones se realizaron repetidamente durante más de un mes y se pudo concluir que en el mes de pruebas la cuenta de la fecha y hora que lleva el medidor fue correcta. En la figura 6.1 se muestran la fecha y hora registradas por el medidor.
Posteriormente, se puso a prueba la seguridad del sistema. Estas pruebas consistieron en la

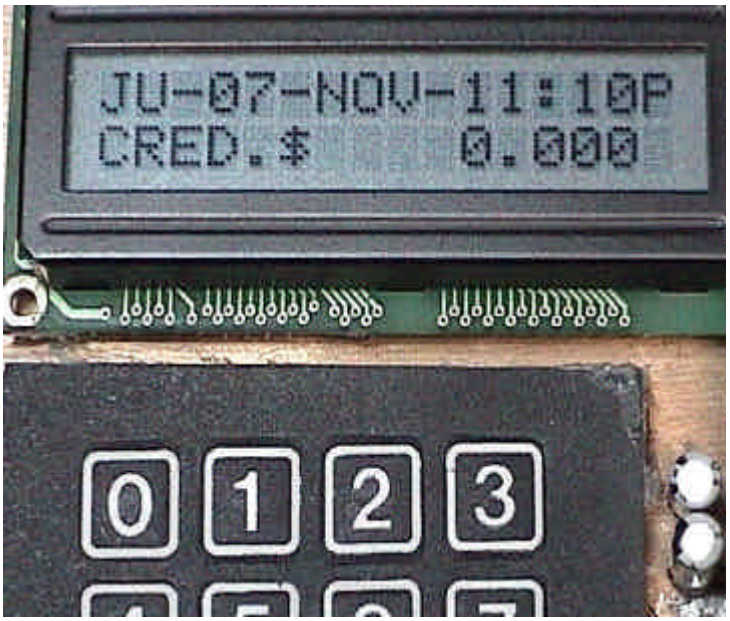


Figura 6.1. Fecha y hora registradas por el medidor de prepago.

introducción de códigos válidos e inválidos en el medidor para comprobar que sólo la introducción de un código válido permite abonar crédito en el medidor. La figura 6.2 muestra la pantalla del medidor que contiene un código antes de ser desencriptado.
Estas pruebas consideraron los siguientes casos:

Figura 6.2. Código introducido por el usuario para abonar crédito en el medidor.

- La introducción de un mismo código válido en repetidas ocasiones para comprobar que un código valido puede emplearse una sola vez.
- La introducción de un código válido que contenía un número de serie de medidor diferente para comprobar que el código sólo puede ser empleado en el medidor cuyo número de serie corresponde con el embebido en el código.
- La introducción en el medidor de 3 códigos inválidos consecutivos para comprobar que el teclado del medidor es bloqueado durante 12 horas y después de transcurrido este tiempo reactiva el teclado y permite nuevamente 3 oportunidades para introducir un código válido. La figura 6.3 muestra la pantalla del medidor cuando el teclado ha sido bloqueado.

Figura 6.3. Medidor de prepago con el teclado desactivado.

- La introducción de varios códigos válidos que contenían diferentes cantidades de crédito para comprobar que el dinero contenido en cada uno de los códigos era abonado por el medidor. La figura 6.4 muestra la pantalla del medidor cuando éste tiene crédito.

Las siguientes pruebas que se realizaron consistieron en comparar las mediciones de voltaje y corriente realizadas por el medidor con aquellas que se registraron con un multímetro digital. Para realizar estas pruebas se conectaron el medidor y el multímetro a un arreglo de lámparas incandescentes conectadas a un reóstato (resistencia variable) y a un VARIAC (autotransformador variable) para modificar los valores de voltaje y corriente de tal forma que se pudieran registrar ambos cambios en el medidor y en el multímetro. En las figuras 6.5 y 6.6 se muestran los valores de voltaje y corriente registrados por el medidor. Otra prueba consistió en conectar el medidor a una carga conocida de 500 Watts y medir el tiempo que tardaba en consumir un kilowatt – hora. En aproximadamente 2 horas se había consumido

un kilowatt – hora y el medidor había restado del crédito que tenia abonado la cantidad en dinero correspondiente a un kilowatt – hora.

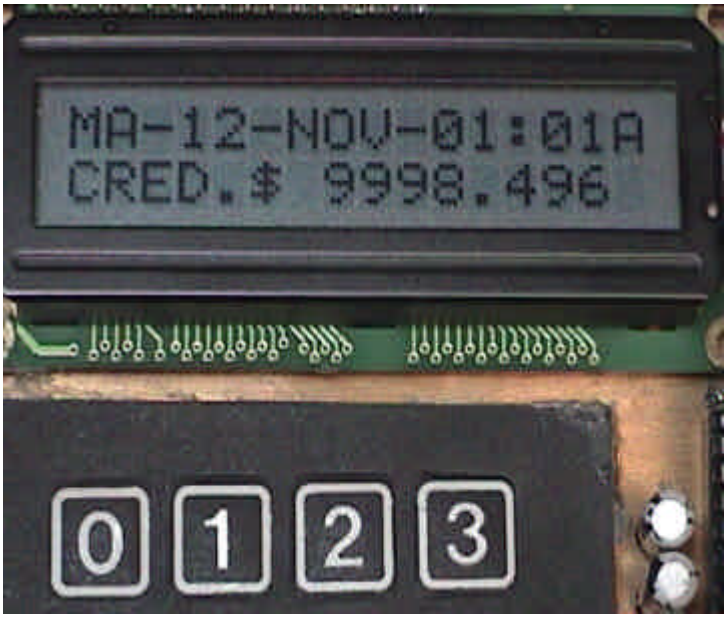


Figura 6.4. Crédito abonado en el medidor.

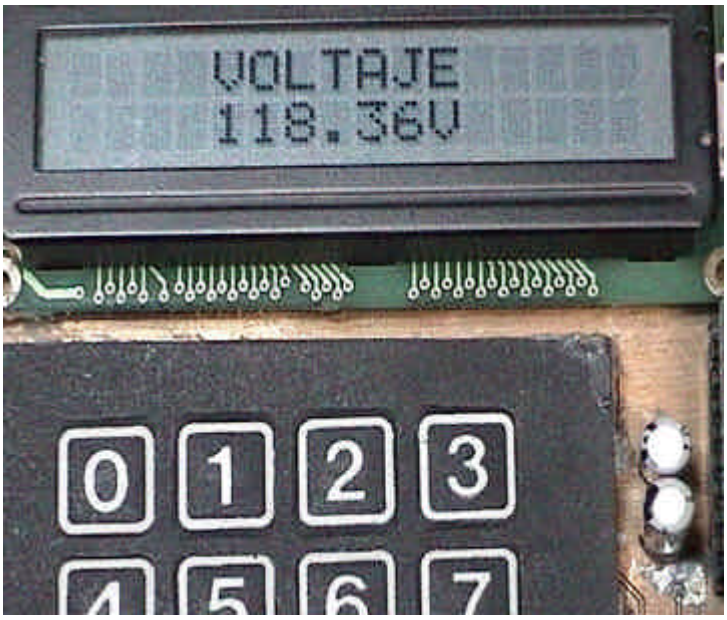


Figura 6.5. Voltaje registrado por el medidor.

Figura 6.6. Corriente registrada por el medidor.

Las mediciones registradas por el medidor, correspondientes a la potencia aparente, la potencia real y el factor de potencia, dependen directamente de la exactitud de las mediciones de voltaje y corriente. Para comprobar los cálculos de potencia aparente, potencia real y factor de potencia se tomaron varias lecturas de voltaje y corriente, y se compararon los resultados mostrados en la pantalla del medidor con los cálculos realizados en una hoja de cálculo. En las figuras 6.7, 6.8 y 6.9 se muestran las variables calculadas por el medidor y desplegadas en la pantalla de cristal líquido.

Figura 6.7. Potencia aparente.

Figura 6.8. Potencia real.

Figura 6.9. Factor de potencia.

En la tabla 6.1.se muestran lecturas de voltaje y corriente tomadas con un multimetro digital y las registradas con el medidor. En la figura 6.10 se muestra la gráfica comparativa de las muestras de voltaje y en la figura 6.11 se muestra la gráfica comparativa de las muestras de corriente.

Tabla 6.1 Comparación de mediciones de voltaje y corriente entre un multimetro digital y el medidor digital de prepago.

	Voltaje multimetro	Voltaje medidor	Corriente multimetro	Corriente medidor
Muestras				
1	118.235	118.24	3.391	3.389
2	118.281	118.27	3.387	3.388
3	118.302	118.31	3.388	3.386
4	118.206	118.21	3.391	3.388
5	118.254	118.25	3.387	3.389
6	118.273	118.27	3.385	3.385
7	118.258	118.26	3.385	3.384
8	118.248	118.24	3.384	3.384
9	118.243	118.24	3.386	3.389
10	118.265	118.26	3.395	3.398
11	118.309	118.31	3.396	3.394
12	118.289	118.28	3.394	3.394
13	118.293	118.29	3.389	3.392
14	118.278	118.28	3.387	3.387
15	118.285	118.28	3.384	3.387
16	118.279	118.28	3.386	3.387
17	118.264	118.28	3.389	3.39
18	118.257	118.26	3.392	3.391
19	118.264	118.26	3.389	3.392
20	118.271	118.27	3.389	3.392
21	118.228	118.23	3.387	3.389
22	118.208	118.21	3.384	3.385

	Voltaje multimetro	Voltaje medidor	Corriente multimetro	Corriente medidor
Muestras				
23	118.225	118.23	3.385	3.385
24	118.219	118.22	3.387	3.388
25	118.211	118.22	3.391	3.389
26	118.236	118.24	3.393	3.395
27	118.224	118.25	3.389	3.392
28	118.236	118.24	3.389	3.392
29	118.244	118.23	3.386	3.385
30	118.233	118.23	3.389	3.387

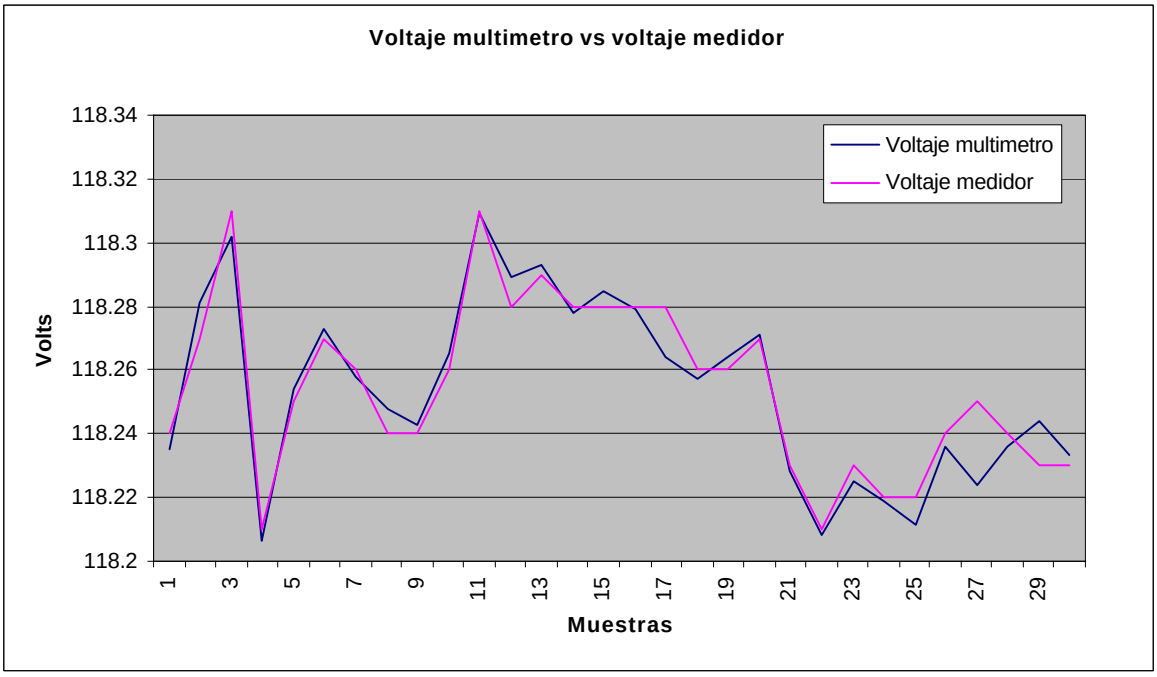


Figura 6.10. Mediciones de voltaje entre un multimetro digital y el medidor digital de prepago.

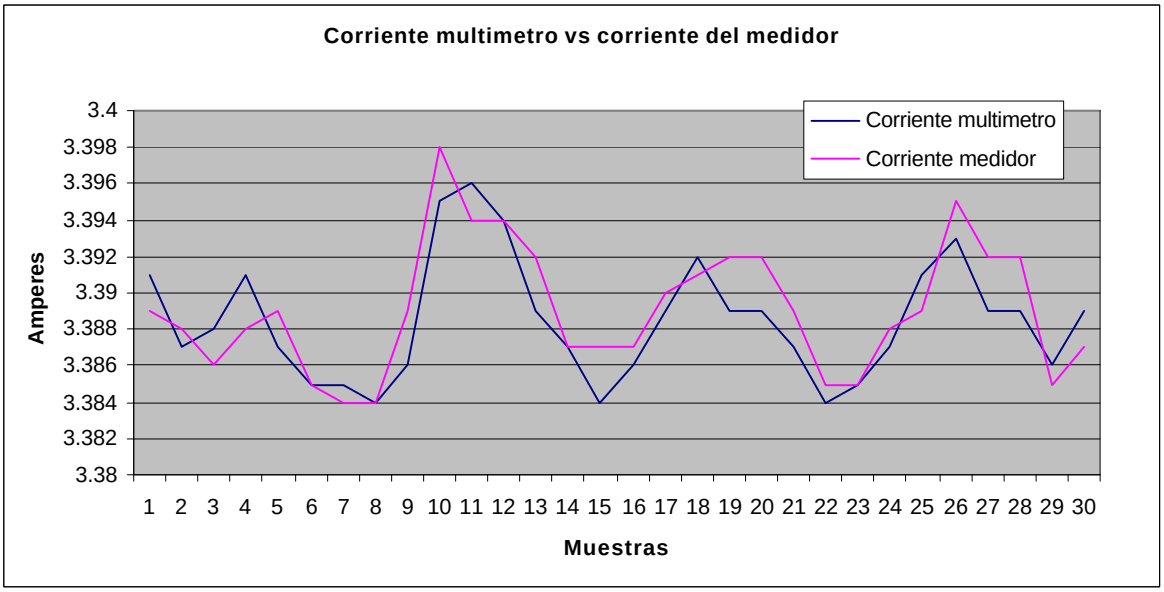


Figura 6.11. Mediciones de corriente entre un multimetro digital y el medidor digital de prepago.

6.2 Resultados

Las mediciones realizadas por el multímetro digital y las realizadas por el medidor no son exactamente iguales debido a las siguientes razones:

La línea de alimentación varía de manera significativa, por ejemplo, en menos de un segundo puede cambiar de 115 VAC a 120 VAC.
Los multímetros empleados llevan ya más de 5 años de servicio sin haber sido calibrados, por lo cual no puede asegurarse su exactitud.
Las cargas resistivas generan calor, el cual provoca variaciones en su resistencia conforme pasa el tiempo, por lo cual no es posible asegurar que una carga de 100 Watts consumirá la misma cantidad de energía durante una hora o más.

Cualquier equipo de medición debe ser calibrado y sometido a pruebas bajo condiciones específicas y controlables para asegurar su exactitud. Para poder calibrar el medidor diseñado e incrementar su exactitud se requiere de equipo especializado y costoso.

En términos generales, las mediciones realizadas por el medidor varían en poco menos de un 1% respecto a los multímetros digitales con que se cuenta en el laboratorio de electrónica.

6.3 Resumen

En este capítulo se describieron las pruebas realizadas a lo largo del desarrollo del medidor para comprobar su funcionamiento. Después se describieron los resultados obtenidos al

comparar las mediciones realizadas por el medidor con aquellas realizadas por un multímetro digital y se explicaron las posibles razones por las cuales ambas mediciones difirieron. En el capítulo siguiente se describirán las conclusiones del trabajo de tesis realizado.

CAPÍTULO 7

CONCLUSIONES

En este capítulo se describen las metas alcanzadas al finalizar el desarrollo del medidor eléctrico digital de prepago. Además, se describen las mejoras y posibles trabajos a futuro basados en este proyecto de investigación.

7.1 Logros alcanzados

La primer meta alcanzada fue el desarrollo de un medidor eléctrico que incorpora la tecnología de prepago del tipo teclado con un nivel de seguridad por lo menos igual al que ofrecen varias compañías a nivel mundial (encriptación de 20 dígitos). Otro logro alcanzado fue la implementación del algoritmo de desencriptación en el microcontrolador ya que su arquitectura es de 8 bits y no cuenta con instrucciones avanzadas para realizar operaciones matemáticas de multiplicación y división por lo que fue necesario diseñar y programar las rutinas para la desencriptación de la manera más eficiente aprovechando al máximo los recursos limitados del microcontrolador. En la actualidad las compañías que generan y distribuyen energía eléctrica se ven en la necesidad de implementar sistemas de generación, distribución y cobro más eficientes debido a la creciente demanda y a la limitada capacidad de generación de energía. Un sistema de prepago de teclado plantea una solución a corto plazo para incrementar el flujo de capital ya que su principal ventaja es que la energía se cobra antes de ser consumida. Otras de sus ventajas incluyen la eliminación de costosos sistemas de facturación y recaudación además de crear una conciencia de ahorro de energía en el consumidor ya que puede conocer cuanta energía consume en un día o en unas horas en vez de recibir una factura dos meses después desconociendo de qué forma gastó esa energía. El efecto que produce esta tecnología es similar al de la telefonía celular donde los consumidores tratan de limitar el uso de sus teléfonos para ahorrar lo más posible, incluso aquellas personas con posibilidades económicas por encima de la media.

El medidor diseñado también se puede emplear en aplicaciones de subarrendamiento o simplemente para medir el consumo de equipos eléctricos y electrodomésticos así como para medir el voltaje, la corriente y la potencia suministrados a dichos equipos o subinstalaciones eléctricas.

La tabla 7.1 muestra las principales especificaciones técnicas del medidor eléctrico digital de prepago.

Tabla 7.1. Principales especificaciones del medidor de prepago

Voltaje medido	50 VAC – 300 VAC
Corriente medido	0 A – 7.5 A
Frecuencia de operación	50 Hz – 70 Hz
Precisión de las mediciones	+/- 1%
Desencriptación	20 dígitos @ 500ms

El medidor digital fue concebido como una alternativa a los medidores eléctricos mecánicos que se emplean hoy día en la mayoría de las residencias del país y para demostrar que las nuevas tecnologías implementadas en los institutos y centros de investigación del país pueden tener un impacto positivo en la industria y la economía.

7.2 Trabajos a futuro

El medidor desarrollado puede ser mejorado o modificado para responder a varias necesidades las cuales incluyen:

- El desarrollo de medidores trifásicos que son ampliamente empleados en la industria y que pudieran incluir comunicaciones inalámbricas, ópticas o vía TCP/IP (Internet) con la compañía que presta el servicio de energía eléctrica.
- Medidores que pueden funcionar en la banda ISM (inalámbrica) para monitorear en tiempo real el consumo, fallas en el servicio eléctrico, periodos de mayor consumo, variaciones de consumo dependiendo de la estación del año y hábitos de consumo por parte de los usuarios.
- Medidores para registrar la eficiencia y el consumo de aparatos eléctricos.
- Medidores para el subarrendamiento en edificios de departamentos o de difícil acceso.

El trabajo realizado no limita su implementación en microcontroladores sino también en computadoras personales, procesadores digitales de señales, combinaciones de todos o de dispositivos más avanzados.

7.3 Resumen

En este capítulo se describieron las metas alcanzadas al desarrollar el medidor, su impacto tecnológico y las posibles mejoras o trabajos a futuro para desarrollar medidores más complejos y con más funciones.

ANEXOS

A DIAGRAMAS ELÉCTRICOS DEL MEDIDOR DE PREPAGO
B PROGRAMA EN ENSAMBLADOR DEL MICROCONTROLADOR
C. PROGRAMA PARA LA ENCRIPTACIÓN DE CLAVES

ANEXO A

DIAGRAMAS ELÉCTRICOS DEL MEDIDOR DE PREPAGO

A continuación se muestran los diagramas eléctricos de las dos tarjetas del medidor de prepago.

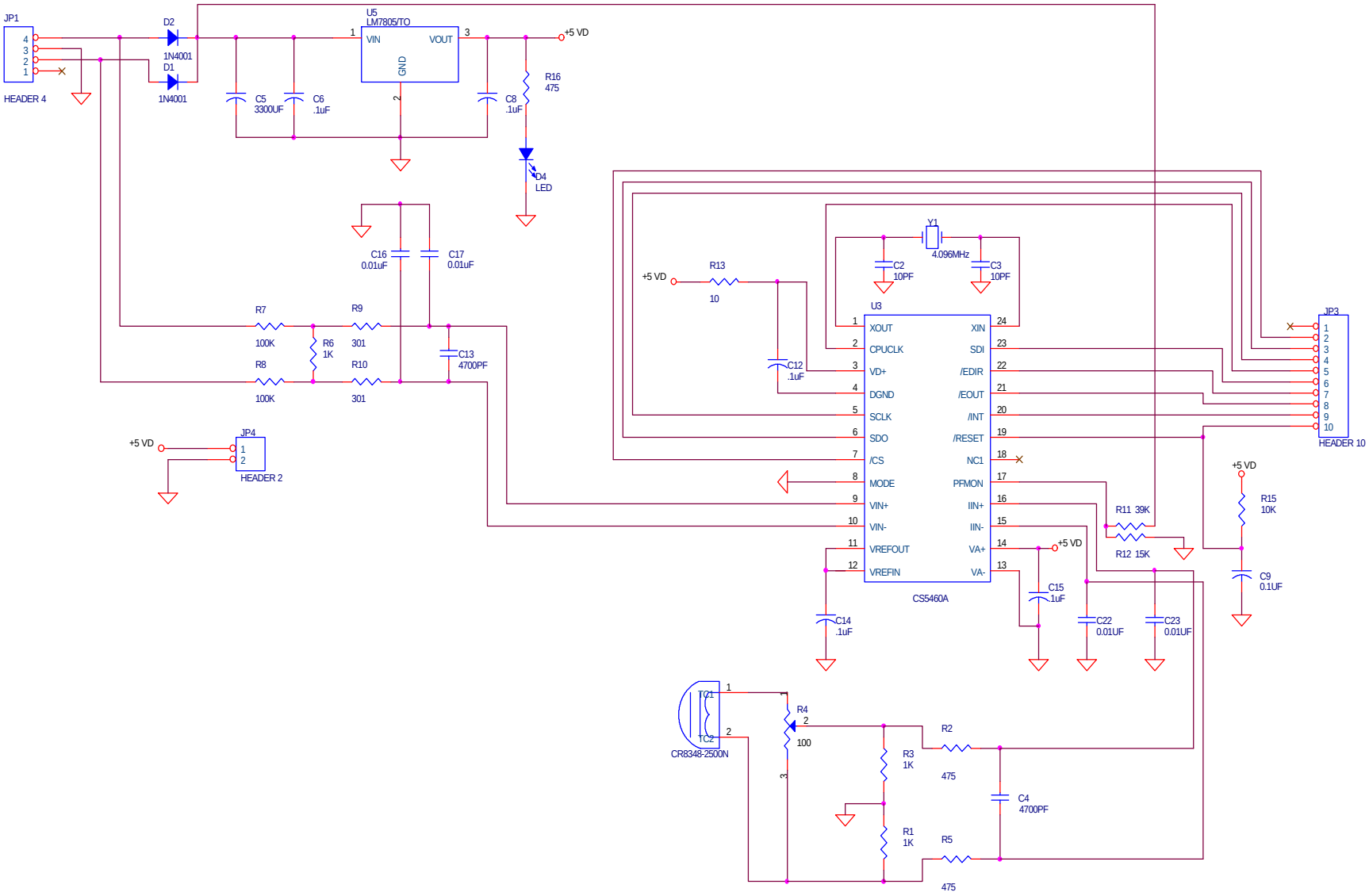


Diagrama en ORCAD de la primer tarjeta que contiene el circuito CS5460A

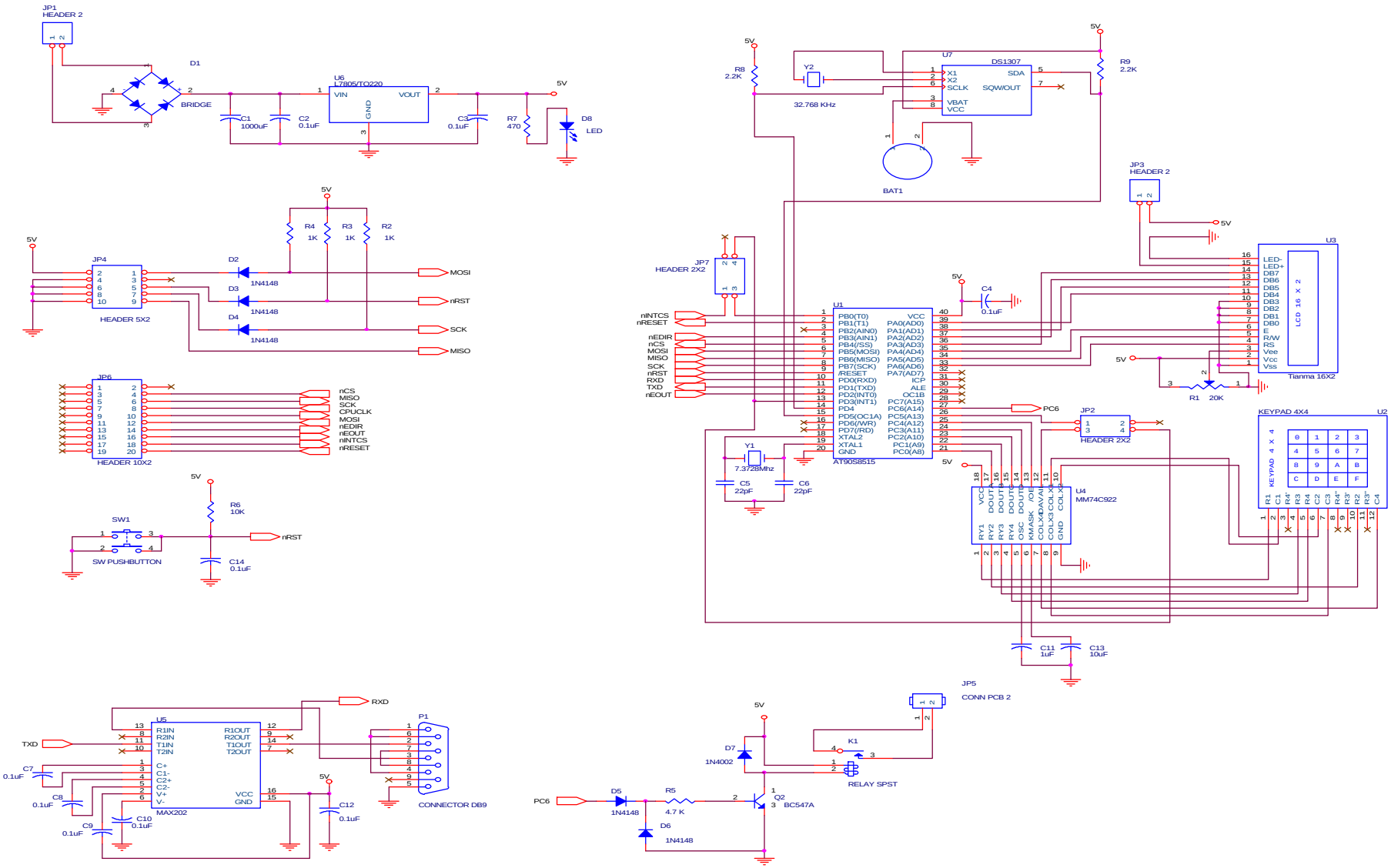


Diagrama en ORCAD de la segunta tarjeta que contiene el microcontrolador AT90S8515

ANEXO B

PROGRAMA EN ENSAMBLADOR PARA EL MICROCONTROLADOR

A continuación se muestra el listado del programa embebido en el microcontrolador.

```
.include "8515def.inc"
.list
.listmac

.DSEG                   ;Segmento de datos

SERVICIO:       .byte   1       ;Cero indica que no hay credito disponible y uno lo contrario
BANDERA:        .byte   1       ;Indicador del modo de despliegue
ENERSAVE:       .byte   1       ;Indicador para guardar la energia consumida en NVRAM
SDECRYPT:       .byte   1       ;Indicador para verificar el comienzo de la desencriptacion
LAPSEDHR:       .byte   1       ;Indicador de las horas transcurridas despues de 3 intentos para poder volver a introducir numeros
ABONAR:         .byte   1       ;Indicador para codigos desencriptados validos. 1=valido, 0=invalido

HORAS:          .byte   1       ;Valor de horas en binario
MINUTOS:        .byte   1       ;Valor de minutos en binario
SEGUNDOS:       .byte   1       ;Valor de segundos en binario
AMPM:           .byte   1       ;Indicador AM (0) o PM (1)
FECHA:          .byte   1       ;Valor del día del mes en binario
MES:            .byte   1       ;Valor del mes en binario
ANNO:           .byte   1       ;Valor del año en binario

ANNREF:         .byte   1       ;Referencia de año debe de ser cargada desde la NVRAM al inicializar y cambiar cada que cambie el año
valor:          .byte   8       ;
 codigo:        .byte   22      ;Variable para almacenar los dígitos tecleados por el usuario en ASCII necesario para desplegar en pantalla
CODNASCII:      .byte   21      ;Numero tecleado por el usuario en decimal
cuantos:        .byte   1       ;Numero de dígitos tecleados por el usuario en binario
VIP:            .byte   1       ;Indica el despliegue del voltaje o corriente o potencia aparente
WFP:            .byte   1       ;Indica el despliegue de potencia real o factor de potencia
TIMEOUT:        .byte   1       ;Indicador del tiempo que se despliega un mensaje

REALTIME:       .byte   1       ;Variable para actualizar valores del RTC
REFRESH:        .byte   1       ;Indicador para actualizar los mensajes en la pantalla

RTCS:           .byte   1       ;Segundos leidos desde el RTC
RTCM:           .byte   1       ;Minutos leidos desde el RTC
RTCH:           .byte   1       ;Horas leidos desde el RTC
RTCDY:          .byte   1       ;Dia de la semana leido desde el RTC
RTCDATE:        .byte   1       ;Fecha leida desde el RTC
RTCMES:         .byte   1       ;Mes leido desde el RTC
RTCANI:         .byte   1       ;Año leido desde el RTC

PAPTEMP1:       .byte   1       ;Registros auxiliaries para calcular la potencia aparente
PAPTEMP2:       .byte   1       ;

ENERGYH:        .byte   1       ;Energia abonada y almacenada en RTC
ENERGYL:        .byte   1       ;
ENER1:          .byte   1       ;Cuenta de pulsos de energia registrados
ENER2:          .byte   1
```

```
NVMESES:        .byte   12      ;Contadores para los meses

;*************** RAM asignada para la desencriptacion*******************************************

REMINDER:       .byte 9         ;Resultado temporal de la desencriptacion
VALOR_D:        .byte 9         ;Almacena llave D para desencriptacion
D_TEMP:         .byte 9         ;Almacena temporalmente una copia de la llave D
VALOR_N:        .byte 9         ;Valor de N cargado desde la EEPROM a RAM
VALOR_M:        .byte 9         ;Resultado de la desencriptacion en binario
VALOR_C:        .byte 9         ;Valor introducido desde el teclado y que tiene que ser desencriptado
VALOR_MDEC:     .byte 21        ;Numero desencriptado en decimal*****A partir de aqui cargar apuntadores H y L usando loadmens
SERIALN:        .byte 9         ;Numero de serie almacenado en decimal
DECMES:         .byte 1         ;Numero de mes desencriptado en hexadecimal
VERIFD:         .byte 1         ;Digito verificador en hexadecimal

TARIFH:         .byte 1         ;Byte alto de la tarifa en binario
TARIFL:         .byte 1         ;Byte bajo de la tarifa en binario

ENERRESH:       .byte 1         ;Byte alto del dinero no abonado en binario
ENERRESL:       .byte 1         ;Byte bajo del dinero no abonado en binario

ABON1:          .byte 1         ;Byte alto de la tarifa multiplicada por mil en binario
ABON2:          .byte 1         ;Byte medio de la tarifa multiplicada por mil en binario
ABON3:          .byte 1         ;Byte bajo de la tarifa multiplicada por mil en binario
TRIES:          .byte 1         ;Numero de intentos fallidos al introducir el codigo
TRTIME:         .byte 1         ;Tiempo transcurrido desde el ultimo intento fallido
DEACTKEY:       .byte 1         ;Indicador que desactiva el teclado
CREDTDEC:       .byte 10        ;Credito en ASCII para desplegar


.CSEG                   ;Segmento de codigo

.def    fbin    =r16            ;Valor binario de 8 bits
.def    tBCDL   =r16            ;Resultado BCD (digito mas significativo)
.def    tBCDH   =r17            ;Resultado BCD (digito menos significativo)

.def    TXDATA  =r17            ;Registros para comunicacion con el CS5460A
.def    TXDATA0 =r18
.def    TXDATA1 =r19
.def    TXDATA2 =r20
.def    RXDATA0 =r21
.def    RXDATA1 =r22
.def    RXDATA2 =r23
.def    temp    =r24            ;Registro de proposito general
.def    temp2   =r25            ;Registro de proposito general
.def    COUNT   =r26            ;Registro de proposito general
.def    stareg  =r27            ;Registro para almacenar temporalmente el registro de estado del procesador
```

```
.equ     SYNC0   = $FE                    ;Constantes para comunicacion con el CS5460A
.equ     SYNC1   = $FF                    ;

;*************** Registros I2C ***************************************
.def     TWIdelay  = r16                  ;Variable para retardo
.def     TWIdata   = r17                  ;Registro IIC para transferencia de datos
.def     TWIadr    = r18                  ;Registro para direccion IIC y sentido de la transferencia
.def     TWIstat   = r19                  ;Registro de estado del bus IIC
.def     dato      =r21                   ;Dato a escribir en RTC
.def     numreg    =r22                   ;Direccion a escribir en RTC

;*************** Datos de calibracion a escribir en el CS5460A ************
.equ     CORRMAXH        =0X1D            ;Corriente maxima 7500 (1D4C) 750 (2EE)
.equ     CORRMAXL        =0X4C            ;

.equ     VOLTAMAXH       =0X75            ;Voltaje maximo 30000(7530h) 3000 (BB8h) 300 (12Ch)
.equ     VOLTAMAXL       =0X30            ;

.equ     PAMAXH          =0X57            ;Potencia  maxima 22500 (57E4) 2250 (8CA)
.equ     PAMAXL          =0XE4

.equ     voltcalH        =0X3E            ;Constantes de calibración del CS5460A
.equ     voltcalM        =0XD9
.equ     voltcalL        =0XF9

.equ     corrcalH        =0X3F
.equ     corrcalM        =0XA5
.equ     corrcalL        =0X00

.equ     ofcorrH         =0XAE
.equ     ofcorrM         =0X77
.equ     ofcorrL         =0XB4

.equ     ofvoltH         =0X05
.equ     ofvoltM         =0X87
.equ     ofvoltL         =0X69

.equ     prrH            =0X00
.equ     prrM            =0X00
.equ     prrL            =0X48            ;Pulse rate register= 2.25 3600 pulsos por KW-H

;*************** LLaves de la desencriptacion *************
.equ     RSA_N_1=0x0A
.equ     RSA_N_2=0x49
.equ     RSA_N_3=0x86
.equ     RSA_N_4=0x3F                     ;N= 189765432191559752533
.equ     RSA_N_5=0x28
.equ     RSA_N_6=0x6A
```

```
.equ    RSA_N_7=0x65
.equ    RSA_N_8=0x5B
.equ    RSA_N_9=0x55



.equ    RSA_D_1=0x00
.equ    RSA_D_2=0x47
.equ    RSA_D_3=0x34
.equ    RSA_D_4=0xA6                    ;D= 5130908430267877761
.equ    RSA_D_5=0x08
.equ    RSA_D_6=0x59
.equ    RSA_D_7=0x2C
.equ    RSA_D_8=0x89
.equ    RSA_D_9=0x81

;**** Constantes IIC ****

.equ    SCLP    = 4                     ; SCL
.equ    SDAP    = 5                     ; SDA
.equ    b_dir   = 0                     ; Bit para direccion de la transferencia en TWIadr
.equ    TWIrd   = 1                     ; Bit para lectura desde el dispositivo IIC
.equ    TWIwr   = 0                     ; Bit para escritura hacia el dispositivo IIC

.macro  timer
        out     OCR1AH,fbin             ;Inicializacion del contador A del Timer 1
        out     OCR1AL,temp             ;
.endmacro

.macro  loadmens
        ldi     ZH,high(2*@0)           ;Carga el byte mas significativo en ZH
        ldi     ZL,low(2*@0)            ;Carga el byte menos significativo en ZL
.endmacro

.macro SBIs_HR                          ;Habilita bits en el regitro indicado como primer parametro
        in temp,@0
        ori temp,@1
        out @0,temp
.endmacro

.macro CBIs_HR                          ;Deshabilita bits en el regitro indicado como primer parametro
        in temp,@0
        andi temp,@1
        out @0,temp
.endmacro
```

```
;Tabla de vectores de interrupcion

        rjmp    RESET
        rjmp    EX_INT0         ;Interrupcion externa 0
        rjmp    EX_INT1         ;Interrupcion externa 1
        reti                    ;
        rjmp    TC1M            ;Interrupcion del timer 1

;*********************************************************************
;*      EX_INT0
;*
;*      INTERRUPCION EXTERNA 0
;*      Pin PD0 conectado a EOUT
;*********************************************************************
EX_INT0:
                push    stareg
                in      stareg,SREG
                push    temp
                push    temp2
                push    count
                push    YH
                push    YL
                push    ZH
                push    ZL

                lds     ZH,TARIFH       ;Carga la tarifa en Y
                lds     ZL,TARIFL

                lds     YL,ENER1        ;Carga los pulsos contados en los registros de energia
                lds     YH,ENER2        ;

                in      count,PINB      ;Lee EDIR
                in      count,PINB
                andi    count,0x08      ;Limpia los demas bits
                cpi     count,0         ;Si es cero decrementa la cuenta en 1
                breq    decreg          ;Salta para decrementar la cuenta

                adiw    YL,1            ;Incrementa la cuenta de energia en 1
                cpi     YH,0x0E         ;Compara para saber si la cuenta llego a 3600
                brne    finint          ;Primero compara el byte mas significativo
                cpi     YL,0x10         ;Luego el menos significativo
                brne    finint          ;Si la cuenta es diferente a 3600 salta al fin de la interrupcion

                clr     YH              ;Pone a cero los registros para reiniciar la cuenta
                clr     YL
                sts     ENER1,YL        ;Guarda los pulsos generados en RAM
                sts     ENER2,YH
```

```
                lds     temp2,ENERGYH       ;Carga los registros de la energia abonada
                lds     temp,ENERGYL        ;

                sbiw    temp,1              ;Decrementa en un KW-H la energia total abonada
                breq    endcredit           ;Si el resultado de la resta es cero salta a endcredit

                sts     ENERGYH,temp2       ;Guarda el nuevo valor en memoria
                sts     ENERGYL,temp

                lds     count,ABON1         ;Byte mas significativo del credito abonado
                lds     temp2,ABON2
                lds     temp,ABON3

                sub     temp,ZL             ;Resta del credito abonado una unidad de tarifa
                sbc     temp2,ZH            ;Resta con acarreo
                clr     ZL                  ;Limpia este registro para empatar tres restas
                sbc     count,ZL            ;Resta con acarreo

                sts     ABON1,count         ;Guarda el credito modificado en memoria
                sts     ABON2,temp2
                sts     ABON3,temp
                rjmp    finint0             ;Brinca al final de la interrupcion

finint:         sts     ENER1,YL
                sts     ENER2,YH
                rjmp    finint0

endcredit:      CBIs_HR GIMSK,0b10000000    ;Deshabilita la interrupcion externa 0
                cbi     PORTC,6             ;Desconecta el relevador
                clr     temp
                sts     SERVICIO,temp       ;SERVICIO es la bandera para indicar que ya no hay credito disponible
                sts     ENERGYH,temp        ;Guarda cero en la energia abonada
                sts     ENERGYL,temp
                sts     ABON1,temp          ;Guarda cero en el credito abonado
                sts     ABON2,temp
                sts     ABON3,temp
                rjmp    finint0             ;Salta al final de la rutina

decreg:         sbiw    YL,1                ;Decrementa en 1 la cuenta de energia
                brcs    finint0             ;Si la resta es positiva salta al fin de la interrupcion

                ldi     YH,0x0E             ;Carga 3599 en YH:YL
                ldi     YL,0x0F             ;
                sts     ENER1,YL            ;Guarda los pulsos generados en RAM
                sts     ENER2,YH

                lds     temp2,ENERGYH       ;Carga los registros de los KW-H abonados
                lds     temp,ENERGYL        ;
```

```
                adiw    temp,1              ;Incrementa en uno la energia abonada
                sts     ENERGYH,temp2       ;Guarda el nuevo valor en memoria
                sts     ENERGYL,temp

                lds     count,ABON1         ;Byte mas significativo del credito abonado
                lds     temp2,ABON2
                lds     temp,ABON3

                add     temp,ZL             ;Resta del credito abonado una unidad de tarifa
                adc     temp2,ZH            ;Resta con acarreo
                clr     ZL                  ;Limpia este registro para empatar tres restas
                adc     count,ZL            ;Resta con acarreo

                sts     ABON1,count         ;Guarda el credito modificado en memoria
                sts     ABON2,temp2
                sts     ABON3,temp

finint0:        pop     ZL
                pop     ZH
                pop     YL
                pop     YH
                pop     count
                pop     temp2
                pop     temp
                out     SREG,stareg         ;Restaura SREG
                pop     stareg
                reti
;*************************************************************************
;*      EX_INT1
;*
;*      INTERRUPCION EXTERNA 1
;*      Pin PD1 conectado a los botones
;*************************************************************************
EX_INT1:

                push    stareg
                in      stareg,SREG
                push    temp
                push    count
                push    ZH
                push    ZL
                push    YH
                push    YL



                cbi     PORTC,PC4           ;Activa la lectura del teclado
                nop                         ;Retardo para poder leer el teclado
```

```
            nop                         ;
            in      temp,PINC           ;Lectura al puerto C para leer la tecla oprimida
            in      temp,PINC           ;
            sbi     PORTC,PC4           ;Desactiva el teclado
            andi    temp,0x0F           ;Limpia la parte alta del puerto C la parte que interesa esta en el nibble bajo


            cpi     temp,0x0A           ;Compara si el digito tecleado es mayor de 9
            brpl    no_digit            ;No, sigue analizando que boton fue tecleado
            lds     count,cuantos       ;Inicializar en ceros cuantos, reliza a cabo otras funciones incluida la desencriptacion si se oprimio "entrar credito"
            cpi     count,21            ;Compara si son 21 digitos los tecleados
            breq    sale1               ;Si mas de 21 digitos no seguira aceptando mas digitos por parte del usuario y simplemente regresara al
            inc     count               ;Incrementa el numero de digitos tecleados
            sts     cuantos,count       ;Como no se ha llegado al limite guarda nuevamente en RAM la cuanta de cuantos digitos han sido tecleados
            clr     YH                  ;Limpia YH
            ldi     YL,CODNASCII        ;Y apunta a CODNASCII para guardar el numero tecleado sin el offset del ascii
            clr     ZH                  ;Carga en el registro Z la direccion en RAM donde se guardaran los digitos tecleados
            ldi     ZL,codigo           ;
            dec     count               ;Decrementa el contador ya que sufrio un preincremento
            add     ZL,count            ;Suma al apuntador Z la posicion relativa donde se guardara el numero
            add     YL,count            ;Suma al apuntador Y la posicion relativa donde se guardara el numero
            st      Y+,temp             ;Guarda en la direccion especificada por YL, con el offset añadido por count y apunta a la siguiente dir en ram
            subi    temp,-$30           ;Añade el offset para desplegar en ASCII
            st      Z+,temp             ;Guarda en la direccion especificada por ZL, con el offset añadido por count y apunta a la siguiente dir en ram
            clr     temp                ;Carga un cero en temp
            st      Z+,temp             ;Pone un cero al final de la cadena, esto solo es util cuando los digitos tecleados son menos o igual a diez
            ldi     temp,1              ;Carga un 1 en temp
            sts     BANDERA,temp        ;Guarda en bandera 1 para indicar el modo de entrada de digitos
            ldi     temp,30             ;
            sts     TIMEOUT,temp        ;30 segundos de despliegue despues de cada caracter introducido
            ldi     temp,1
            sts     REFRESH,temp        ;Carga un 1 en REFRESH para indicar que se actualice la pantalla lo mas pronto posible
            rjmp    sale1               ;


no_digit:   cpi     temp,0x0A           ;Checa si se oprimio la tecla de funcion A
            breq    V_I_P               ;Si, salta a la rutina de despliegue de I,V y Pot aparente

            cpi     temp,0x0B           ;Checa si se oprimio la tecla de funcion B
            breq     W_FP               ;

            cpi     temp,0x0C           ;Modo para limpiar uno a uno los numeros introducidos
            breq    cleandig

            cpi     temp,0x0D           ;Modo para comenzar la desencriptacion
            brne    sale1
            rjmp    ckdecript
```

```
sale:       ldi     temp,30             ;Carga en timeout 30 segundos de despliegue
            sts     TIMEOUT,temp        ;Reinicia el tiempo de despliegue en caso de que se seleccione otra visualizacion mientras otra no termine aun

sale1:      pop     YL
            pop      YH
            pop     ZL
            pop     ZH
            pop     count
            pop     temp
            out     SREG,stareg         ;Restaura SREG
            pop     stareg

            reti

V_I_P:      clr     count               ;Pone un cero en count
            sts     WFP,count           ;Limpia WFP para desplegar desde potencia real en caso de que sea seleccionado despues de VIP
            lds     count,VIP
            inc     count               ;Incrementa para que la cuenta siempre empiece en 1
            sts     VIP,count
            cpi     count,1
            breq    selvolts
            cpi     count,2
            breq    selcorr
            cpi     count,3
            breq    selpa
            clr     count
            rjmp    V_I_P

W_FP:       clr     count               ;Pone un cero en count
            sts     VIP,count           ;Limpia VIP para desplegar desde voltaje en caso de que sea seleccionado despues de WFP
            lds     count,WFP
            inc     count               ;Incrementa para que la cuenta siempre empiece en 1
            sts     WFP,count
            cpi     count,1
            breq    selwatts
            cpi     count,2
            breq    selfp
            clr     count
            rjmp    W_FP

cleandig:   ldi     temp,7              ;Limpieza de digitos tecleados
            sts     BANDERA,temp
            ldi     temp,1
            sts     REFRESH,temp        ;Carga un 1 en REFRESH para indicar que se actualice la pantalla
            ldi     temp,30             ;
            sts     TIMEOUT,temp        ;30 segundos de despliegue despues de cada caracter que limpia
            rjmp    sale1
```

```
ckdecript:
                ldi     temp,1                  ;Carga un uno en temp
                sts     SDECRYPT,temp           ;Indica que se oprimio el boton para desencriptar
                rjmp     sale1


selvolts:       ldi     temp,2
                sts     BANDERA,temp
                rjmp    sale

selcorr:        ldi     temp,3
                sts     BANDERA,temp
                rjmp    sale

selpa:          ldi     temp,4
                sts     BANDERA,temp
                rjmp    sale


selwatts:       ldi     temp,5
                sts     BANDERA,temp
                rjmp    sale

selfp:          ldi     temp,6
                sts     BANDERA,temp
                rjmp    sale

;*********************************************************************
;*      TC1M
;*      Rutina de servicio al timer
;*
;*
;*********************************************************************
TC1M:
                push    stareg
                in      stareg,SREG
                push    temp
                push    count

                ldi     temp,1
                sts     REFRESH,temp            ;Carga un 1 en REFRESH para indicar que se actualice la pantalla
                lds     count,TIMEOUT           ;Carga la variable TIMEOUT en count
                cpi     count,0                 ;Si es cero, salta a times
                breq    times                   ;
                dec     count                   ;Decrementa cada segundo
                sts     TIMEOUT,count           ;Guarda TIMEOUT despues de decrementarlo
                rjmp    times1                  ;Continua con la rutina
```

```
times:      clr     temp            ;Limpia temp
            sts     BANDERA,temp    ;Lo guarda en bandera para que regrese al modo de despliegue normal BANDERA==0
            sts     VIP,temp        ;Limpia el modo de despliegue entre voltaje, corriente y potencia   VIP==0
            sts     W_FP,temp       ;Limpia el modo de despliegue entre potencia real y factor de potencia W_FP==0
times1:     lds     temp,SEGUNDOS   ;Carga los segundos transcurridos en temp
            inc     temp            ;Los incrementa
            cpi     temp,60         ;Los compara con sesenta
            breq    SETMINUTOS      ;Si la cuenta llego a sesenta incrementa los minutos
            sts     SEGUNDOS,temp   ;No, guarda en memoria la cuenta de los segundos

end_time:   pop     count           ;Restaura los registros modificado por esta interrupcion
            pop     temp
            out     SREG,stareg     ;Restaura SREG
            pop     stareg
            reti                    ;Regresa y habilita las interrupciones

SETMINUTOS: clr     temp            ;Limpia temp que contiene la cuenta de los segundos (cuenta de 0..59)
            sts     SEGUNDOS,temp   ;Guarda en memoria los segundos (0)
            lds     temp,MINUTOS    ;Carga los minutos transcurridos en temp
            inc     temp            ;Los incrementa
            lds     count,ENERSAVE
            cpi     count,10        ;
            brge    saveen
saveset:    cpi     temp,60         ;Los compara con sesenta
            breq    SETHORAS        ;Si la cuenta llego a sesenta incrementa las horas
            sts     MINUTOS,temp    ;No, guarda en memoria la cuenta de los minutos
            rjmp    end_time        ;Como no se incrementan las horas brinca al final de la rutina
saveen:     lds     temp,SERVICIO   ;Carga SERVICIO en temp
            cpi     temp,0          ;Si servicio es cero no almacena los valores de energia en NVRAM
            breq    saveset
            ldi     count,1
            sts     ENERSAVE,count
            rjmp    saveset

SETHORAS:   lds     temp,TRTIME     ;Cada hora incrementa este contador para mantener desactivado el teclado
            inc     temp            ;
            sts     TRTIME,temp     ;
            clr     temp            ;Limpia temp que contiene la cuenta de los minutos (cuenta de 0..59)
            sts     MINUTOS,temp    ;Guarda en memoria los minutos (0)
            lds     temp,HORAS      ;Carga las horas transcurridas en temp
            inc     temp            ;Las incrementa
            cpi     temp,12         ;Si llego a las 12 refresca valores desde el RTC
            brge    setrtc
            cpi     temp,13         ;Las compara con trece
            brge    CLRHORAS        ;Si la cuenta llego a Trece pone las horas en 1
noRTC:      sts      HORAS,temp     ;Guarda el valor modificado de horas
            rjmp    end_time        ;Brinca al final de la rutina
```

```
setrtc:     lds     count,SEGUNDOS      ;Verifica si son las 12:00 exactamente para refrescar valores desde el RTC
            cpi     count,0             ;Compara que los segundos sean 0
            brne    noRTC               ;No, salta
            ldi     temp,1
            sts     REALTIME,temp       ;Guarda un 1 en REALTIME para indicar la actualizacion de todas las variables
            rjmp    end_time            ;Salta al fin de la rutina
CLRHORAS:   ldi     temp,1              ;Carga un 1 en temp
            sts     HORAS,temp          ;Guarda horas (1)
            rjmp    end_time            ;Brinca al final de la rutina


;*****************************************************************************************************************************
;*    RESET------------AQUI INICIA EL PROGRAMA
;*****************************************************************************************************************************

RESET:      ldi     temp,LOW(RAMEND)    ;Inicializa el Stack
            out     SPL,temp            ;
            ldi     temp,HIGH(RAMEND)   ;
            out     SPH,temp            ;

;************* Escribe en RAM las llaves para la desencriptacion   ************

            ldi     temp,RSA_N_1
            sts     VALOR_N,temp
            ldi     temp,RSA_N_2
            sts     VALOR_N+1,temp
            ldi     temp,RSA_N_3
            sts     VALOR_N+2,temp
            ldi     temp,RSA_N_4
            sts     VALOR_N+3,temp
            ldi     temp,RSA_N_5
            sts     VALOR_N+4,temp
            ldi     temp,RSA_N_6
            sts     VALOR_N+5,temp
            ldi     temp,RSA_N_7
            sts     VALOR_N+6,temp
            ldi     temp,RSA_N_8
            sts     VALOR_N+7,temp
            ldi     temp,RSA_N_9
            sts     VALOR_N+8,temp
            ldi     temp,RSA_D_1
            sts     VALOR_D,temp
            ldi     temp,RSA_D_2
            sts     VALOR_D+1,temp
            ldi     temp,RSA_D_3
            sts     VALOR_D+2,temp
```

```
                ldi     temp,RSA_D_4
                sts     VALOR_D+3,temp
                ldi     temp,RSA_D_5
                sts     VALOR_D+4,temp
                ldi     temp,RSA_D_6
                sts     VALOR_D+5,temp
                ldi     temp,RSA_D_7
                sts     VALOR_D+6,temp
                ldi     temp,RSA_D_8
                sts     VALOR_D+7,temp
                ldi     temp,RSA_D_9
                sts     VALOR_D+8,temp

                ldi     temp,0x02               ;Carga la tarifa en RAM (0.596 pesos/KW-H)
                sts     TARIFH,temp
                ldi     temp,0x54
                sts     TARIFL,temp

;************* Escribe los datos de calibracion en el CS5460A  ************
                rcall   spi_setup               ;Inicializa el puerto SPI
                rcall   CS5460_RESET            ;Reset por software al CS5460A e inicializa el puerto SPI del CS5460A
                rcall   CS5460_RESET            ;Reset por software al CS5460A e inicializa el puerto SPI del CS5460A

                ldi     TXDATA,0b01001000       ;Comando de escritura al registro Voltage Gain Calibration
                ldi     TXDATA0,voltcalH
                ldi     TXDATA1,voltcalM
                ldi     TXDATA2,voltcalL
                rcall   SPIWRITE

                ldi     TXDATA,0b01000100       ;Comando de escritura al registro Current Gain Calibration
                ldi     TXDATA0,corrcalH
                ldi     TXDATA1,corrcalM
                ldi     TXDATA2,corrcalL
                rcall   SPIWRITE

                ldi     TXDATA,0b01100000       ;Comando de escritura al registro Current ac OFFSET
                ldi     TXDATA0,ofcorrH
                ldi     TXDATA1,ofcorrM
                ldi     TXDATA2,ofcorrL
                rcall   SPIWRITE

                ldi     TXDATA,0b01100010       ;Comando de escritura al registro Voltage ac OFFSET
                ldi     TXDATA0,ofvoltH
                ldi     TXDATA1,ofvoltM
                ldi     TXDATA2,ofvoltL
                rcall   SPIWRITE

                ldi     TXDATA,0b01001100       ;Comando de escritura al registro pulse rate register
```

```
                ldi     TXDATA0,prrH
                ldi     TXDATA1,prrM
                ldi     TXDATA2,prrL
                rcall   SPIWRITE

;*****************************************************************************

                rcall   lcd_init                ;Inicializa el Display
                rcall   clear_display           ;Limpia el Display
                rcall   KEYBO                   ;Inicializacion del teclado
                sbi     DDRC,PC6                ;El pin PC6 es la salida que controla el relevador
                cbi     PORTC,6                 ;Desconecta el relevador hasta verificar que hay credito disponible

                SBIs_HR MCUCR,0b00001110        ;Interrupcion para el teclado en el flanco de subida de int1 y en el flanco de bajada para int0 (energia)
                SBIs_HR GIMSK,0b10000000        ;Interrupcion externa 1 habilitada y la cero deshabilitada

                clr     temp
                sts     BANDERA,temp            ;Modo normal de despliegue
                sts     CUANTOS,temp            ;Limpia contador de los digitos tecleados
                sts     VIP,temp                ;Limpia contador del modo de despliegue      de voltaje corriente y pot. aparente
                sts     WFP,temp                ;Limpia contador del modo de despliegue de pot real y factor de potencia
                sts     TIMEOUT,temp            ;Limpia contador para tiempo de despliegue
                sts     ENERSAVE,temp           ;Inicializa en cero la variable para guardar la energia en NVRAM
                sts     SERVICIO,temp           ;Inicializa SERVICIO en cero
                sts     SDECRYPT,temp           ;Inicializa indicador de desencriptacion en cero

                rcall   LEER_NVRAM              ;Lee energia y contadores de meses desde la NVRAM del RTC

;***************Determina SERVICIO*************************************************************

                lds     temp,ENERGYL            ;Carga el byte bajo de energia en temp
                lds     temp2,ENERGYH           ;Carga el byte alto de energia en temp2
                or      temp,temp2              ;Operacion OR entre los 2 bytes. Si la operacion es cero Z=1
                breq    no_ser                  ;Si no es cero pone a uno la bandera de SERVICIO y conecta el relevador
                ldi     temp,1                  ;Carga un uno en temp
                sts     SERVICIO,temp           ;Pone a uno la bandera de SERVICIO y el relevador se activara mas delante
                SBIs_HR GIMSK,0b01000000        ;Habilita la interrupcion externa 0 (Energia)

;*****************************************************************************************************

no_ser:         ldi     fbin,0x70               ;
                ldi     temp,0x80               ;Carga el registro OCR1A con el valor
                timer   fbin,temp               ;28800
                ldi     temp,0x40               ;Pone a 1 el bit correspondiente a
                out     TIMSK,temp              ;la interrupcion OC1E1 del timer

                rcall   LEER_RTC                ;Lee fecha y hora desde el reloj de tiempo real
```

```
                ldi     TXDATA,0b11101000       ;Comando para que el CS5460 comience a hacer conversiones de manera continua
                rcall   SPICOMANDO              ;
                ldi     temp,0b00001100         ;CK/256 CTC1=1 limpia el timer cuando la cuenta es igual a 28800 (1segundo)
                out     TCCR1B,temp             ;Carga el registro      TCCR1B y comienza a correr el contador interno
                sei                             ;Habilita todas las interrupciones

main:           lds     temp,SERVICIO           ;Carga servicio en temp
                cpi     temp,1                  ;Compara con uno, uno implica que hay credito y por lo tanto el servicio debe de estar activo

                brne    des_con                 ;Si temp=0, entonces salta a des_con para deahabilitar el relevador
                sbi     PORTC,6                 ;Conecta el relevador ya que hay credito disponible
main2:          lds     temp,REFRESH
                cpi     temp,1
                breq    actualiza
                lds     temp,ENERSAVE
                cpi     temp,1
                breq    save_nv
                lds     temp,REALTIME
                cpi     temp,1
                breq    actclock
                lds     temp,SDECRYPT
                cpi     temp,1
                breq    ckdecrypt
                lds     temp,DEACTKEY
                cpi     temp,1
                breq    des_key
                rjmp    main

des_con         cbi     PORTC,6                 ;Asegura la desconexion del relevador hasta verificar que hay credito disponible
                CBIs_HR GIMSK,0b10000000        ;Asegura la deshabilitacion de la interrupcion externa 0
                rjmp    main2                   ;Continua en main2

save_nv:        rcall   WE_NVRAM                ;Guarda los valores de energia en la NVRAM del RTC cada 10 minutos
                clr     temp                    ;Limpia temp
                sts     ENERSAVE,temp           ;Pone a cero el indicador para que guarde los valores solo cada 10 minutos
                rjmp    main                    ;Continua en main

actclock:       rcall   LEER_RTC                ;Refresca los valores desde el RTC para mantener presicion y actualizar la fecha
                clr     temp                    ;Limpia temp
                sts     REALTIME,temp           ;Pone a cero el indicador para refrescar los datos del RTC cada que pasen 12 horas
                rcall   CKYEAR
                rjmp    main

ckdecrypt:      rcall   VALIDECRYPT
                rjmp    main

des_key:        CBIs_HR GIMSK,0b01000000        ;Asegura la deshabilitacion de la interrupcion externa 1 (teclado)
                rjmp    main
```

```
actualiza:      lds     temp,DEACTKEY           ;Carga en temp DEACTKEY
                cpi     temp,0                  ;Lo compara con cero
                breq    ok_normod               ;Si es cero salta
                lds     temp,TRTIME             ;NO, carga las horas transcurridas desde la desactivacion del teclado
                cpi     temp,24                 ;Compara con 24
                brne    ok_normod               ;Si aun no son 24 horas salta
                SBIs_HR GIMSK,0b10000000        ;Se cumplio el plazo por lo tanto reactiva el teclado
                clr     temp                    ;Se cumplieron las 24 horas
                sts     DEACTKEY,temp           ;Guarda en cero en DEACTKEY para indicar que el teclado ya fue reactivado
                sts     TRIES,temp              ;Pone a cero el contador de intentos

ok_normod:      rcall   clear_display           ;Limpia la pantalla antes de visualizar cada modo
                lds     temp,BANDERA
                cpi     temp,0
                breq     normod
                cpi     temp,1
                breq    muenter
                cpi     temp,2
                breq     muvolt
                cpi     temp,3
                breq    mucorr
                cpi     temp,4
                breq    mupota
                cpi     temp,5
                breq    mupreal
                cpi     temp,6
                breq    mufp
                cpi     temp,7
                breq    clrdigit
                cpi     temp,8
                breq    errdigit
                cpi     temp,9
                breq    errdesc

normod:         clr     count                   ;Limpia count
                sts     cuantos,count           ;Como pasaron 30 seg sin que se tecleara un digito o se iniciara la desencriptacion
                                                ;Limpia el contador de digitos tecleados
                rcall   DIS_HORA                ;Modo normal de despliegue
                rjmp    set_refresh
muenter:        rcall   FIXCODE
                rjmp    set_refresh

muvolt:         rcall   DIS_VOLTAJE
                rjmp    set_refresh

mucorr:         rcall   DIS_CORR
                rjmp    set_refresh
```

```
mupota:     rcall   DIS_POTA
            rjmp    set_refresh

mupreal     rcall   DIS_POTR
            rjmp    set_refresh

mufp:       rcall   DIS_FP
            rjmp    set_refresh

clrdigit:   clr     ZH              ;Limpia ZH
            ldi     ZL,codigo       ;ZL apunta a codigo
            lds     count,cuantos   ;Carga el numero de digitos tecleados en count
            cpi     count,0         ;Compara si el numero de digitos es cero
            breq    set_refresh     ;Si es cero no lo modifica y salta
            dec     count           ;Decrementa en uno la cuenta de digitos
            add     ZL,count        ;Apunta al digito a modificar en ascii
            sts     cuantos,count   ;Actualiza el numero de digitos tecleados
            ldi     temp,0          ;Carga un cero en temp
            st      Z,temp          ;Guarda un cero en la posicion para que solo despliegue los numeros hasta donde encuentre el cero
            ldi     temp,1
            sts     BANDERA,temp
            rjmp    muenter         ;Salta a muenter ya que tiene que actualizar inmediatamente los digitos tecleados

errdigit:
            loadmens coderr1        ;Carga la direccion donde se encuentra el mensaje de error por la falta de digitos para iniciar desencriptacion
            rcall   LINEA1          ;Imprime el mensaje de error

            ldi     temp,0x0C       ;Carga la direccion de la segunda linea
            rcall   manda_lcd       ;En la direccion 40H
            ldi     temp,0x00       ;
            rcall   manda_lcd       ;
            loadmens coderr2        ;Carga la direccion donde se encuentra el mensaje de error por la falta de digitos para iniciar desencriptacion
            rcall   LINEA1          ;Imprime el mensaje de error
            rjmp    set_refresh

errdesc:    loadmens coderr1        ;Carga la direccion donde se encuentra el mensaje de error por la falta de digitos para iniciar desencriptacion
            rcall   LINEA1          ;Imprime el mensaje de error
            ldi     temp,0x0C       ;Carga la direccion de la segunda linea
            rcall   manda_lcd       ;En la direccion 40H
            ldi     temp,0x00       ;
            rcall   manda_lcd       ;
            loadmens deserr1        ;Carga la direccion donde se encuentra el mensaje de error por la falta de digitos para iniciar desencriptacion
            rcall   LINEA1          ;Imprime el mensaje de error
            rjmp    set_refresh


set_refresh: clr    temp
```

```
                    sts     REFRESH,temp            ;Guarda un cero en REFRESH
                    rjmp    main
;*********************************************************************
;Se emplea el puerto A para controlar el lcd
;A7 A6 A5 A4 A3 A2 A1 A0
;x  RS RW E  D7 D6 D5 D4
;El control se hace empleando la interface de 4 bits
;*********************************************************************
lcd_init:           rcall   busywait
                    ldi     temp,0x02               ;RS bajo,E bajo, RW bajo y 0010 (modo 4 bits)
                    rcall   manda_lcd
                    ldi     temp,0x02               ;Function set 4 bits parte alta
                    rcall   manda_lcd
                    ldi     temp,0x08               ;Function set 2 lineas y formato de 5x7
                    rcall   manda_lcd
                    ldi     temp,0x00               ;Display on & cursor (underline, blink on)
                    rcall   manda_lcd
                    ldi     temp,0x0C;
                    rcall   manda_lcd
                    ret
;*********************************************************************
manda_lcd:          out     PORTA,temp
                    rcall   pulse_e
                    rcall   busywait
                    ret
;*********************************************************************
pulse_e:            sbi     PORTA,4                 ;Pone E en alto
                    nop                             ;Lo mantiene 2 ciclos= 500ns
                    nop
                    nop
                    nop                             ;
                    cbi     PORTA,4                 ;Pone E en bajo
                    ret                             ;Regresa
;*********************************************************************
busywait:           ldi     temp,0xF0               ;A0 a A3 Salidas
                    out     DDRA,temp               ;D4 a D7 como entrada
                    sbi     PORTA,5                 ;RW=1, modo de lectura
                    nop
                    cbi     PORTA,6                 ;RS=0
                    nop
                    nop                             ;tAS=140 ns minimo
                    nop
busyread:           sbi     PORTA,4                 ;E en alto
                    nop                             ;tEH=450 ns minimo
                    nop
                    in      temp,PINA               ;Lee el puerto A
                    in      temp,PINA               ;2 VECES
                    bst     temp,3                  ;Guarda el bit 3 (D7) en la bandera T
```

```
                    cbi     PORTA,4             ;Pone a E en bajo
                    nop                         ;500 ns de espera de E en bajo
                    nop                         ;Solo si MCLK=4Mhz
                    nop
                    nop
                    rcall   pulse_e             ;Se transfiere el Nibble Bajo que no interesa
                    brts    busyread            ;Si la bandera T (D7) esta en alto regresa
                    ser     temp                ;Temp todos unos
                    out     DDRA,temp           ;Al puerto A
                    ret                         ;
;***************************************************************************
clear_display:      ldi     temp,0x00           ;Limpia el display y regresa a la direccion 0
                    rcall   manda_lcd
                    ldi     temp,0x01
                    rcall   manda_lcd
                    ret
;***************************************************************************
;*  MULTIP2424
;*
;*          Rutina para multiplicar 2 datos de 24 bits
;***************************************************************************
.def    mc480   =r23                            ;Byte bajo del multiplicando
.def    mc481   =r22                            ;Byte medio del multiplicando
.def    mc482   =r21                            ;Byte alto del multiplicando


.def    mp480   =r20                            ;Byte bajo del multiplicador
.def    mp481   =r19                            ;Byte medio del multiplicador
.def    mp482   =r18                            ;Byte alto del multiplicador

.def    mp483   =r0                             ;Byte 4 del resultado (LSB)
.def    mp484   =r1                             ;Byte 5 del resultado (MSB)
.def    mp485   =r2                             ;Byte 6 del resultado (MSB)

MULTIP2424:         clr     mp485               ;Limpia los tres bytes mas altos del resultado
                    clr     mp484
                    clr     mp483
                    ldi     count,24            ;Inicializar contador para procesar 48 bits del resultado
                    clc
                    lsr     mp482
                    ror     mp481
                    ror     mp480
m48u_1:             brcc    noad48              ;Si el bit 0 del multiplicador es uno
                    add     mp483,mc480         ;Suma el multiplicando bajo al byte 1 del resultado
                    adc     mp484,mc481         ;Suma el multiplicando medio al byte 2 del resultado
                    adc     mp485,mc482         ;Suma el multiplicando alto al byte 3 del resultado

noad48:             ror     mp485               ;Corrimiento a la derecha del byte 6 del resultado
```

```
                ror     mp484                   ;Rota a la derecha el byte 5 del resultado
                ror     mp483                   ;Rota a la derecha el byte 4 del resultado
                ror     mp482                   ;Rota a la derecha el byte 3 del multiplicador
                ror     mp481                   ;Rota a la derecha el byte 2 del multiplicador
                ror     mp480                   ;Rota a la derecha el byte 1 del multiplicador
                dec     count                   ;Decrementa contador
                brne    m48u_1                  ;Continua si no ha procesado todos los bits
                ret

;***************************************************************************
;*  MULTIP1616
;*
;*          Rutina para multiplicar 2 datos de 16 bits
;***************************************************************************
.def    mc160   =r22                            ;Byte bajo del multiplicando
.def    mc161   =r21                            ;Byte alto del multiplicando
.def    mp160   =r19                            ;Byte bajo del multipliador
.def    mp161   =r18                            ;Byte alto del multipliador

.def    mp162   =r0                             ;Byte 2 del resultado
.def    mp163   =r1                             ;Byte 3 del resultado (MSB)

MULTIP1616:     clr     mp163                   ;Limpia los 2 bytes mas altos del resultado
                clr     mp162
                clc                             ;limpia carry
                ldi     COUNT,16                ;Inicializa contador
                lsr     mp161
                ror     mp160
m16u_1:         brcc    noad16                  ;Si el bit 0 del multiplicador es uno
                add     mp162,mc160             ;Suma el multiplicando bajo al byte 2 del resultado
                adc     mp163,mc161             ;Suma el multiplicando bajo al byte 3 del resultado
noad16:         ror     mp163                   ;Corrimiento a la derecha del byte 3 del resultado
                ror     mp162                   ;Rota a la derecha el byte 2 del resultado
                ror     mp161                   ;Rota a la derecha el byte de resultado 1 y el byte alto del multiplicador
                ror     mp160                   ;Rota a la derecha el byte de resultado 0 y el byte bajo del multiplicador
                dec     COUNT                   ;Decrementa contador
                brne    m16u_1                  ;Continua si no ha procesado todos los bits
                ret
;*****************************************************************************
;*          "div16u"
;*          División de 16 bits / 16
;*****************************************************************************

.def    rem16L=r4                               ;Residuo bajo
.def    rem16H=r5                               ;Residuo alto
.def    dd16uL  =r17                            ;Pot real y resultado
.def    dd16uH  =r18                            ;Pot real y resultado
.def    dd16uHH =r19                            ;Pot real
```

```
.def    dv16uL  =r0                             ;Pot aparente
.def    dv16uH  =r1                             ;Pot aparente

div16u:         clr     rem16L                  ;Limpia el byte bajo del residuo
                sub     rem16H,rem16H           ;Limpia el byte alto del residuo y el acarreo
                ldi     count,25                ;Inicializa contador
d16_1:          rol     dd16uL                  ;Corrimiento a la izquierda del dividendo
                rol     dd16uH
                rol     dd16uHH
                dec     count                   ;Decrementa contador
                brne    d16_2                   ;Salta si no ha procesado todos los bits
                ret
d16_2:          rol     rem16L                  ;Corre el dividendo dentro del residuo
                rol     rem16H
                sub     rem16L,dv16uL           ;Residuo = Residuo – Divisor
                sbc     rem16H,dv16uH           ;
                brcc    d16_3                   ;Si el resultado es negativo
                add     rem16L,dv16uL           ;Restaura el residuo
                adc     rem16H,dv16uH
                clc                             ;Introduce un cero en el resultado
                rjmp    d16_1
d16_3:          sec                             ;Introduce un uno en el resultado
                rjmp    d16_1
;*****************************************************************************
;*  bin2BCD16
;*
;*      Rutina para convertir un numero de 16 bits a bcd de 5 digitos
;*****************************************************************************
.def    tBCD0   =r13                            ;Valor BCD de los digitos 1 y 0
.def    tBCD1   =r14                            ;Valor BCD de los digitos 3 y 2
.def    tBCD2   =r15                            ;Valor BCD del digito 4
.def    fbinL   =r0                             ;Valor binario byte bajo
.def    fbinH   =r1                             ;Valor binario byte alto

.equ    AtBCD0  =13             ;Direccion de tBCD0
.equ    AtBCD2  =15             ;Direccion de tBCD1

bin2BCD16:      ldi     count,16                ;Inicializa contador
                clr     tBCD2                   ;Limpia los 3 bytes del resultado
                clr     tBCD1
                clr     tBCD0
                clr     ZH                      ;
bBCDx_1:        lsl     fbinL                   ;Corrimiento a la izquierda del valor de entrada
                rol     fbinH                   ;En todos los bytes
                rol     tBCD0                   ;
                rol     tBCD1
                rol     tBCD2
                dec     count                   ;Decrementa contador
```

```
                brne    bBCDx_2                 ;Si el contador no es cero
                ret                             ;Regresa

bBCDx_2:        ldi     r30,AtBCD2+1            ;Z apunta al MSB+1 del resultado
bBCDx_3:        ldi     temp,-Z                 ;Carga Z con predecremento
                subi    temp,-$03  ;add 0x03
                sbrc    temp,3                  ;Si el bit 3 no es cero
                st      Z,temp                  ;Vuelve almacenar Z
                ld      temp,Z                  ;Carga Z
                subi    temp,-$30  ;add 0x30
                sbrc    temp,7                  ;Si el bit 7 no es cero
                st      Z,temp                  ;Lo vuelve almacenar
                cpi     ZL,AtBCD0               ;Proceso los 3?
                brne    bBCDx_3                 ;Regresa si no
                rjmp    bBCDx_1
;***************************************************************************
;*      "bin2BCD8"
;*      Conversion de binario a BCD
;***************************************************************************
bin2bcd8:       clr     tBCDH                   ;Limpia el digito mas significativo del resultado
bBCD8_1:        subi    fbin,10                 ;Entrada = Entrada - 10
                brcs    bBCD8_2                 ;Salta si hay carry
                inc     tBCDH                   ;Incrementa el digito mas significativo
                rjmp    bBCD8_1                 ;
bBCD8_2:        subi    fbin,-10                ;Compensa la resta extra
                subi    tBCDH,-0x30             ;Suma ASCII para desplegar
                subi    tBCDL,-0x30             ;Suma ASCII para desplegar
                ret
;***************************************************************************
;*      DIS_VOLTAJE
;*
;***************************************************************************
DIS_VOLTAJE:    loadmens mensaje1               ;Carga la direccion del mensaje de voltaje
                rcall   LINEA1                  ;Escribe el mensaje en la primer linea del LCD
                rcall   SPIRES                  ;
                ldi     TXDATA,0b00011000       ;Comando de Lectura al registro Vrms
                ldi     mp161,VOLTAMAXH         ;
                ldi     mp160,VOLTAMAXL         ;
                rcall   LEE_IV                  ;Lee el registro y lo regresa en BCD
                rcall   FIX_V                   ;Arregla el numero para poder ser desplegado "punto decimal"
potcorr:        ldi     temp,0x0C               ;Carga la direccion de la segunda linea
                rcall   manda_lcd               ;En la direccion 40H
                ldi     temp,0x04               ;
                rcall   manda_lcd               ;
                clr     ZH                      ;Limpia ZH
                ldi     ZL,valor                ;Carga ZL con la direccion de valor
                rcall   DESP_VALOR              ;Imprime el voltaje RMS
                ret
```

```
;*****************************************************************************
;*          DIS_CORR
;*
;*****************************************************************************
DIS_CORR:       loadmens mensaje2               ;Carga la direccion del mensaje de corriente
                rcall    LINEA1                 ;Escribe el mensaje en la primer linea del LCD
                rcall    SPIRES                 ;Checar si realmente se necesita
                ldi      TXDATA,0b00010110      ;Comando de Lectura al registro Irms
                ldi      mp161,CORRMAXH         ;
                ldi      mp160,CORRMAXL         ;
                rcall    LEE_IV                 ;Lee el registro y lo regresa en BCD
                rcall    FIX_I                  ;Arregla el numero para poder ser desplegado "punto decimal"
                rjmp     potcorr
;*****************************************************************************
;*          DIS_POTA
;*
;*****************************************************************************
DIS_POTA:       loadmens mensaje3               ;Carga la direccion del mensaje de potencia aparente
                rcall    LINEA1                 ;Escribe el mensaje en la primer linea del LCD
                rcall    SPIRES                 ;Checar si realmente se necesita
                ldi      TXDATA,0b00010110      ;Comando de Lectura al registro Irms
                rcall    SPIREAD                ;Lee el registro cuya direccion esta en TXDATA (Irms)
                mov      TXDATA0,RXDATA0        ;Mueve temporalmente los datos antes de hacer la siguiente lectura
                mov      TXDATA1,RXDATA1
                mov      TXDATA2,RXDATA2
                push     TXDATA0
                push     TXDATA1
                push     TXDATA2
                rcall    SPIRES                 ;Checar si realmente se necesita
                pop      TXDATA2
                pop      TXDATA1
                pop      TXDATA0
                ldi      TXDATA,0b00011000      ;Comando de Lectura al registro Vrms
                rcall    SPIREAD                ;Lee el registro cuya direccion esta en TXDATA (Vrms)
                rcall    MULTIP2424             ;Multiplica V*I (los registros los CS5460A)
                mov      mc160,mp484            ;Mueve los 2 bytes mas significativos del resultado y prepara para volver a multiplicar
                mov      mc161,mp485            ;

                ldi      mp161,PAMAXH           ;
                ldi      mp160,PAMAXL           ;
                rcall    MULTIP1616             ;Multiplica por PAMAX (potencia aparente maxima)
                rcall    bin2BCD16              ;
                clr      ZH                     ;Limpia ZH
                ldi      ZL,valor               ;Carga al registro Z para que apunte a la direccion en RAM donde se guardara el voltaje en ASCII
                rcall    BCDTOASCII             ;Convierte a ASCII para pod
                rcall    FIX_P                  ;
                rjmp     potcorr                ;
```

```
;*****************************************************************************
;*          DIS_POTR
;*
;*****************************************************************************
DIS_POTR:       loadmens mensaje4               ;Carga la direccion del mensaje de voltaje
                rcall    LINEA1                 ;Escribe el mensaje en la primer linea del LCD
                rcall    SPIRES                 ;Checar si realmente se necesita
                ldi      TXDATA,0b00010100      ;Comando de Lectura al registro de energia
                rcall     SPIREAD               ;Lee el registro cuya direccion esta en TXDATA
                clc                             ;Limpia el carry
                rol      RXDATA2                ;Mueve el bit mas significativo al carry
                rol      RXDATA1
                rol      RXDATA0
                brcs     twosc                  ;Si el bit MSB es 1 la energia es negativa y salta
rconti:         ldi      mp161,PAMAXH           ;
                ldi      mp160,PAMAXL           ;
                rcall    MULTIP1616             ;Multiplica por PAMAX (potencia maxima)
                rcall    bin2BCD16
                clr      ZH                     ;Limpia ZH
                ldi      ZL,valor               ;Carga al registro Z para que apunte a la direccion en RAM donde se guardara el voltaje en ASCII
                rcall    BCDTOASCII             ;Convierte a ASCII para pod
                lds      temp,valor+2           ;Recorre los digitos para quitar los 2 ceros que siempre esta en el primer byte
                sts      valor,temp
                lds      temp,valor+3
                sts      valor+1,temp
                lds      temp,valor+4
                sts      valor+2,temp
                ldi      temp,'.'               ;Pone un punto decimal en la tercera posicion
                sts      valor+3,temp
                lds      temp,valor+5
                sts      valor+4,temp
                ldi      temp,'W'               ;Como temp2==0, pone la W de Watts
                sts      valor+5,temp           ;Lo guarda en la cadena
                ldi      temp,0                 ;Indicador de fin de cadena
                sts      valor+6,temp           ;
                clr      YH                     ;
                ldi      YL,valor               ;Y apunta al primer digito en ascii de valor
                ldi      count,2                ;Solo 2 ceros consecutivos se intercambiaran por espacios

otroPr:         ld       temp,Y                 ;Carga el primer digito
                cpi      temp,0x30              ;Lo compara para saber si es un cero (ascii)
                breq      zeroPr                ;Si es un cero salta para intercambiarlo por un espacio en blanco
                rjmp     potcorr
zeroPr:         ldi      temp,' '               ;Carga en temp un espacio en blanco (ascii)
                st       Y+,temp                ;Lo intercambia por el valor original
                dec      count
                brne     otroPr
                rjmp     potcorr
```

```
twosc:          clr     RXDATA0             ;Como la energia es negativa prepara para desplegar cero
                clr     RXDATA1
                rjmp    rconti

;*************************************************************************
;*      DIS_FP
;*
;*************************************************************************
DIS_FP:         loadmens mensaje5           ;Carga la direccion del mensaje de voltaje
                rcall   LINEA1              ;Escribe el mensaje en la primer linea del LCD
                rcall   SPIRES              ;Reset al puerto SPI
                ldi     TXDATA,0b00010110   ;Comando de Lectura al registro Irms
                rcall   SPIREAD             ;Lee el registro cuya direccion esta en TXDATA (Irms)
                mov     TXDATA0,RXDATA0     ;Mueve temporalmente los datos antes de hacer la siguiente lectura
                mov     TXDATA1,RXDATA1
                mov     TXDATA2,RXDATA2
                push    TXDATA0
                push    TXDATA1
                push    TXDATA2
                rcall   SPIRES              ;
                pop     TXDATA2
                pop     TXDATA1
                pop     TXDATA0
                ldi     TXDATA,0b00011000   ;Comando de Lectura al registro Vrms
                rcall   SPIREAD             ;Lee el registro cuya direccion esta en TXDATA (Vrms)
                rcall   MULTIP2424          ;Multiplica V*I (los registros los CS5460A)
                mov     mc160,mp484         ;Mueve los 2 bytes mas significativos del resultado y prepara para volver a multiplicar
                mov     mc161,mp485         ;
                ldi     mp161,PAMAXH        ;Carga los valores para multiplicar
                ldi     mp160,PAMAXL        ;
                rcall   MULTIP1616          ;Multiplica por PAMAX (potencia aparente maxima)
                mov     temp,mp163
                cpi     temp,0
                brne    nzerop
                mov     temp,mp162
                cpi     temp,0
                brne    nzerop
fpover1:        ldi     temp,'1'
                sts     VALOR+2,temp
                ldi     temp,'0'
                sts     VALOR+4,temp
                sts     VALOR+5,temp
                rjmp    papfix
nzerop:         sts     PAPTEMP1,mp163      ;Guarda la potencia aparente
                sts     PAPTEMP2,mp162
                rcall   CALC_FP             ;Calcula el factor de potencia
                mov     fbinH,dd16uH
                mov     fbinL,dd16uL
```

```
                cpi     dd16uL,101
                brge    fpover1
                rcall   bin2BCD16
                clr     ZH                      ;Limpia ZH
                ldi     ZL,valor                ;Carga al registro Z para que apunte a la direccion en RAM donde se guardara el voltaje/corriente en ASCII
                rcall    BCDTOASCII
                lds     temp,VALOR+3
                sts     VALOR+2,temp
papfix:         ldi     temp,'.'
                sts     VALOR+3,temp
                ldi     temp,' '
                sts     VALOR,temp
                sts     VALOR+1,temp
                clr     temp
                sts     VALOR+6,temp
                rjmp    potcorr
;*************************************************************************
;*      CALC_FP
;*
;*************************************************************************
CALC_FP:        rcall   SPIRES                  ;Reset al puerto SPI
                ldi     TXDATA,0b00010100       ;Comando de Lectura al registro de energia
                rcall   SPIREAD                 ;Lee el registro cuya direccion esta en TXDATA
                clc                             ;Limpia el carry
                rol     RXDATA2                 ;Mueve el bit mas significativo al carry
                rol     RXDATA1
                rol     RXDATA0
                brcs    twosc2                  ;Si el bit MSB es 1 la energia es negativa y salta
prezero:        ldi     mp161,PAMAXH            ;Carga los valores para multiplicar
                ldi     mp160,PAMAXL            ;
                rcall   MULTIP1616              ;Multiplica por PAMAX (potencia maxima)
                mov     mc161,mp163             ;Prepara para volver a multiplicar
                mov     mc160,mp162
                clr     mp161                   ;Carga un 100 en el multiplicando
                ldi     mp160,0x64
                rcall   MULTIP1616              ;Multiplica la potencia real por 100
                mov     dd16uHH,mp162           ;Carga la potencia real previamente multiplicada por 10
                mov     dd16uH,mp161            ;y prepara para dividir
                mov     dd16uL,mp160
                lds     dv16uH,PAPTEMP1
                lds     dv16uL,PAPTEMP2
                rcall   div16u                  ;Divide potencia real/potencia aparente
                ret
twosc2:         clr     RXDATA0                 ;Como la energia es negativa prepara para desplegar cero
                clr     RXDATA1
                rjmp    prezero
```

```
;*****************************************************
;*        LEE_IV
;*
;*****************************************************
LEE_IV:         rcall   SPIREAD         ;Lee el registro cuya direccion esta en TXDATA
                rcall   MULTIP1616
                rcall   bin2BCD16       ;Convierte a BCD
                clr     ZH              ;Limpia ZH
                ldi     ZL,valor        ;Carga al registro Z para que apunte a la direccion en RAM donde se guardara el voltaje/corriente en ASCII
                rcall   BCDTOASCII      ;Convierte a ASCII para poder desplegarlo
                ret
;*****************************************************
;*        LINEA1
;*        Imprime mensajes desde la memoria de programa
;*****************************************************
LINEA1:
next_byte:      lpm
                tst     r0
                breq    end_line
                push    r0
                mov     temp,r0
                swap    temp
                andi    temp,$0F
                ori     temp,$40
                rcall   manda_lcd
                pop     r0
                mov     temp,r0
                andi    temp,$0F
                ori     temp,$40
                rcall   manda_lcd
                adiw    ZL,1
                rjmp    next_byte
end_line:       ret
;*****************************************************
;*        DESP_VALOR
;*        Imprime cadenas desde la memoria de datos (RAM)
;*****************************************************
DESP_VALOR:
nxtbyte:        ld      r0,Z+           ;Carga el primer byte en r0
                tst     r0              ;Verifica si tst es cero
                breq    listo           ;Si cero salta a listo
                push    r0              ;Guarda r0
                mov     temp,r0         ;Mueve r0 a temp
                swap    temp            ;Intercambia los nibbles de temp
                andi    temp,$0F        ;Pone a cero el nibble alto
                ori     temp,$40        ;Mascara a temp
                rcall   manda_lcd       ;Manda temp al display (primer nibble)
```

```
                pop     r0              ;Restaura r0
                mov     temp,r0         ;Mueve r0 a temp
                andi    temp,$0f        ;Pone a cero el nibble alto
                ori     temp,$40        ;Mascara a temp
                rcall   manda_lcd       ;Manda temp al display (segundo nibble)
                rjmp    nxtbyte         ;Imprime el siguiente valor
listo:          ret
;*********************************************************
;*      SEND_CHA
;*  Manda al display un solo caracter almacenado en temp
;*********************************************************
SEND_CHA:       push    temp
                swap    temp            ;Intercambia los nibbles de temp
                andi    temp,$0F        ;Pone a cero el nibble alto
                ori     temp,$40        ;Mascara a temp
                rcall   manda_lcd       ;Manda temp al display (primer nibble)
                pop     temp            ;Restaura temp
                andi    temp,$0F        ;Pone a cero el nibble alto
                ori     temp,$40        ;Mascara a temp
                rcall   manda_lcd       ;Manda temp al display (segundo nibble)
                ret
;*********************************************************
;*      BCDTOASCII
;*
;*********************************************************
BCDTOASCII:     ldi     COUNT,3         ;Carga el contador
                clr     YH              ;Limpia YH
                ldi     YL,AtBCD2+1     ;Carga la direccion de tBCD2
NXTBCD:         ld      temp,-Y         ;Carga el primer Byte
                push    temp            ;Guarda temp
                swap    temp            ;Intercambia los nibbles
                andi    temp,0x0F       ;Borra la parte alta del Byte
                subi    temp,-$30       ;Le suma el offset del ASCII
                st      Z+,temp         ;Guarda temp en la direccion que apunta Z e incrementa el apuntador
                pop     temp            ;Restaura temp
                andi    temp,0x0F       ;Borra la parte alta del Byte
                subi    temp,-$30       ;Le suma el offset del ASCII
                st      Z+,temp         ;Guarda temp en la direccion que apunta Z e incrementa el apuntador
                dec     COUNT           ;Decrementa el contador
                brne    NXTBCD
                inc     ZL              ;Al incrementar Z apunta a valor+7
                ldi     temp,0          ;Carga un cero para indicar el final de la cadena
                st      Z,temp          ;Lo guarda al final de la cadena, el byte Valor+6 es donde se pone la unidad de medicion (I,V,W)
                ret
```

```
;*****************************************************
;*          DIS_HORA
;*
;*****************************************************
DIS_HORA:       rcall       clear_display
                loadmens dias                   ;Z apunta a dias
                lds         temp,RTCDY          ;Carga en temp el dia de la semana
                dec         temp                ;la cuenta de los dias de la semana es de 0 a 6
                lsl         temp                ;Multiplica por 2
                lsl         temp                ;Multiplica por 2
                add         ZL,temp             ;Se posiciona ZL sobre el dia a desplegar
                clr         temp                ;Pone temp a 0
                adc         ZH,temp             ;Suma un cero a la parte alta mas el carry que se pudiera generar
                rcall       LINEA1              ;Imprime el dia de la semana

                lds         fbin,FECHA          ;Carga en fbin el dia del mes en que esta en binario
                rcall       bin2bcd8            ;Convierte el dato en ASCII
                mov         temp,tBCDH          ;Manda el primer caracter
                rcall       SEND_CHA            ;
                mov         temp,tBCDL          ;Manda el segundo caracter
                rcall       SEND_CHA            ;
                ldi         temp,'-'            ;Imprime el separador ':'
                rcall       SEND_CHA            ;
                loadmens    meses               ;Z apunta a meses
                lds         temp,MES            ;Carga en temp el mes del año
                dec         temp                ;La cuenta de los meses del año va de 0 a 11
                lsl         temp                ;Multiplica por 2
                lsl         temp                ;Multiplica por 2
                add         ZL,temp             ;Z apunta al mes a desplegar
                clr         temp                ;Pone temp a 0
                adc         ZH,temp             ;Suma un cero a la parte alta mas el carry que se pudiera generar
                rcall       LINEA1
                ldi         temp,'-'            ;Imprime el separador ':'
                rcall       SEND_CHA            ;
                lds         fbin,HORAS          ;Carga en las horas que estan en binario
                rcall       bin2bcd8            ;Convierte el dato en ASCII
                mov         temp,tBCDH          ;Manda el primer caracter
                rcall       SEND_CHA            ;
                mov         temp,tBCDL          ;Manda el segundo caracter
                rcall       SEND_CHA            ;
                ldi         temp,':'            ;Imprime el separador ':'
                rcall       SEND_CHA            ;
                lds         fbin,MINUTOS        ;Carga en fbin los minutos que estan en binario
                rcall       bin2bcd8            ;Convierte el dato en ASCII
                mov         temp,tBCDH          ;Manda el primer caracter
                rcall       SEND_CHA            ;
                mov         temp,tBCDL          ;Manda el segundo caracter
                rcall       SEND_CHA            ;
```

```
                lds     temp,AMPM
                cpi     temp,0
                breq    si_AM
                ldi     temp,'P'
                rcall   SEND_CHA
                rjmp    credito
si_AM:          ldi     temp,'A'
                rcall   SEND_CHA
credito:        ldi     temp,0x0C           ;Carga la direccion de la segunda linea
                rcall   manda_lcd           ;En la direccion 44H
                ldi     temp,0x00           ;
                rcall   manda_lcd           ;
                lds     temp,DEACTKEY
                cpi     temp,1
                breq    tec_dec
                loadmens credit
                rcall   LINEA1
                rcall   FIXCREDIT
                ldi     ZH,HIGH(CREDTDEC)
                ldi     ZL,LOW(CREDTDEC)    ;Z apunta a la direccion donde se esncuentra el credito en ASCII
                rcall   DESP_VALOR          ;Despliega en pantalla el mensaje apuntado por Z
                ret
tec_dec:        loadmens TECDESACT
                rcall   LINEA1
                ret
;*********************************************************
;*          FIX_V
;*
;*********************************************************
FIX_V:          lds     temp,valor+1        ;Recorre los digitos para quitar el cero que siempre esta en el primer byte
                sts     valor,temp
                lds     temp,valor+2
                sts     valor+1,temp
                lds     temp,valor+3
                sts     valor+2,temp
                ldi     temp,'.'            ;Pone un punto decimal en la tercera posicion
                sts     valor+3,temp
                ldi     temp,'V'            ;Como temp2==0, pone la V de Volts
                sts     valor+6,temp        ;Lo guarda en la cadena
                clr     YH                  ;
                ldi     YL,valor            ;Y apunta al primer digito en ascii de valor
                ldi     count,2             ;Solo 2 ceros consecutivos se intercambiaran por espacios
otro:           ld      temp,Y              ;Carga el primer digito
                cpi     temp,0x30           ;Lo compara para saber si es un cero (ascii)
                breq    zeroV               ;Si es un cero salta para intercambiarlo por un espacio en blanco
                ret                         ;Termina la subrutina
zeroV:          ldi     temp,' '            ;Carga en temp un espacio en blanco (ascii)
                st      Y+,temp             ;Lo intercambia por el valor original
```

```
                dec     count
                brne    otro
                ret

;*****************************************************
;*      FIX_I
;*
;*****************************************************
FIX_I:          ldi     temp,' '
                sts     valor,temp
                lds     temp,valor+2        ;Recorre los digitos para quitar el cero que siempre esta en el primer byte
                sts     valor+1,temp
                ldi     temp,'.'
                sts     valor+2,temp
                ldi     temp,'A'            ;Unidad de Amperes
                sts     valor+6,temp        ;Lo guarda en esta localidad de valor..
                ret                         ;Termina al haber encontrado un valor diferente de cero
;*****************************************************
;*      FIX_P
;*
;*****************************************************
FIX_P:          lds     temp,valor+2        ;Recorre los digitos para quitar el cero que siempre esta en el primer byte
                sts     valor,temp
                lds     temp,valor+3
                sts     valor+1,temp
                lds     temp,valor+4
                sts     valor+2,temp
                ldi     temp,'.'            ;Pone un punto decimal en la tercera posicion
                sts     valor+3,temp
                lds     temp,valor+5
                sts     valor+4,temp
                ldi     temp,'V'            ;Como temp2==0, pone la V de Volts
                sts     valor+5,temp        ;Lo guarda en la cadena
                ldi     temp,'A'            ;Como temp2==0, pone la V de Volts
                sts     valor+6,temp        ;Lo guarda en la cadena
                clr     YH                  ;
                ldi     YL,valor            ;Y apunta al primer digito en ascii de valor
                ldi     count,2             ;Solo 2 ceros consecutivos se intercambiaran por espacios
otroP:          ld      temp,Y              ;Carga el primer digito
                cpi     temp,0x30           ;Lo compara para saber si es un cero (ascii)
                breq    zeroP               ;Si es un cero salta para intercambiarlo por un espacio en blanco
                ret                         ;Termina la subrutina
zeroP:          ldi     temp,' '            ;Carga en temp un espacio en blanco (ascii)
                st      Y+,temp             ;Lo intercambia por el valor original
                dec     count
                brne    otroP
                ret
```

```
;*********************************************************************
;*          FIXCODE
;*          Prepara el codigo tecleado por el usuario para poder desplegarlo en pantalla
;*
;*********************************************************************
FIXCODE:
                    loadmens menscode               ;Carga la direccion del mensaje de "clave:"
                    rcall       LINEA1              ;Escribe el mensaje en la primer linea del display
                    clr         ZH                  ;Limpia ZH
                    ldi         ZL,codigo           ;Carga ZL con la direccion de codigo
                    lds         temp,cuantos        ;Carga en temp cuantos para saber cuantos digitos han sido tecleados
                    cpi         temp,11             ;Compara con once
                    brge        masdiez             ;Si se han tecleado once o mas digitos salta a "mas diez"
                    rcall       DESP_VALOR          ;Imprime la primer linea de digitos tecleados
                    ret                             ;Termina la rutina
masdiez:            clr         temp2               ;Carga un cero en temp2
                    lds         temp,codigo+10      ;Carga en temp el onceavo digito tecleado
                    sts         codigo+10,temp2     ;Sustituye el onceavo valor por un cero para que sea el final de la cadena y se imprima en la primer linea
                    push        temp                ;Guarda temp ya que la siguiente linea lo modifica
                    rcall       DESP_VALOR          ;Imprime la primer linea de digitos tecleados
                    pop         temp                ;Restaura temp
                    sts         codigo+10,temp      ;Restaura el valor que tenia la posicion once
                    ldi         temp,0x0C           ;Carga la direccion de la segunda linea
                    rcall       manda_lcd           ;En la posicion 0X45H
                    ldi         temp,0x00           ;
                    rcall       manda_lcd           ;
                    clr         ZH                  ;Limpia ZH
                    ldi         ZL,codigo+10        ;Carga ZL con la direccion de codigo
                    rcall       DESP_VALOR          ;Imprime la segunda linea de digitos tecleados (*nota)
                    ret                             ;Termina la rutina

;*nota: La segunda linea ya tiene el indicador de fin de cadena el cual fue introducido previamente

;*********************************************************************
;*  SPI_SETUP
;*          Rutina de inicializacion del puerto SPI
;*          que se localiza en el puerto B del AT90S8515
;*********************************************************************
SPI_SETUP:          sbi         DDRB,PB7            ;SCLK salida
                    cbi         DDRB,PB6            ;MISO entrada
                    sbi         DDRB,PB5            ;MOSI salida
                    sbi         DDRB,PB4            ;/CS del CS5460
                    sbi         PORTB,PB4           ;/CS en alto
                    ldi         temp,0b01010000     ;MSB FIRST, CPOL=0, CPHA=0, SCK=CLK/4
                    out         SPCR,temp           ;Inicializa el registro de control del SPI
                    ret
```

```
;*********************************************************************
;*  CS5460_RESET
;*
;*          Rutina de inicializacion del CS5460A y de su puerto SPI
;*          realiza el reset via el puerto SPI (software reset)
;*
;*********************************************************************
CS5460_RESET:

                    ldi     TXDATA,0b01000000       ;Guarda en TXDATA el comando para escribir en el registro de configuracion del CS5460A
                    ldi     TXDATA0,0b00000000      ;Los dos siguientes bytes del registro no son afectados bits 8 - 23
                    ldi     TXDATA1,0b00000000      ;Se pone el bit RS en 1 comenzando un ciclo de reset, el bit K0 se pone en 1
                    ldi     TXDATA2,0b10000001      ;Indicando que el divisor de la señal de reloj es 1
                    rcall   SPIWRITE                ;Escribe estos 4 bytes en el CS5460A
SPIRES:             ldi     TXDATA,SYNC1            ;Se reinicializa el puerto serial del CS5460A mandando tres comandos
                    ldi     TXDATA0,SYNC1           ;SYNC1 seguidos de un SYNC0
                    ldi     TXDATA1,SYNC1           ;
                    ldi     TXDATA2,SYNC0           ;
                    rcall   SPIWRITE                ;
                    ret
;*********************************************************************
;*  SPICOMANDO
;*          Manda un comando al CS5460A
;*          Empleando el registro TXDATA
;*********************************************************************
SPICOMANDO:         cbi     PORTB,PB4               ;Habilita /CS
                    out     SPDR,TXDATA             ;Guarda el valor de TXDATA en SPDR
cmd:                sbis    SPSR,SPIF               ;Espera a que la bandera SPIF se ponga en 1
                    rjmp    cmd                     ;Lo cual indica que ya saco el dato por el puerto
                    sbi     PORTB,PB4               ;Deshabilita /CS
                    ret
;*********************************************************************
;*  SPIWRITE
;*          Escribe en los registros internos del CS5460A
;*          Empleando los registros TXDATA,TXDATA0,TXDATA1,TXDATA2 (32 bits)
;*     TXDATA contiene la direccion del registro a ser modificado
;*********************************************************************
SPIWRITE:           cbi     PORTB,PB4               ;/CS activo
                    out     SPDR,TXDATA
w1:                 sbis    SPSR,SPIF
                    rjmp    w1
                    out     SPDR,TXDATA0
w2:                 sbis    SPSR,SPIF
                    rjmp    w2
                    out     SPDR,TXDATA1
w3:                 sbis    SPSR,SPIF
                    rjmp    w3
                    out     SPDR,TXDATA2
```

```
w4:             sbis    SPSR,SPIF
                rjmp    w4
                sbi     PORTB,PB4               ;/CS inactivo
                ret
;*************************************************************************
;*  SPIREAD
;*          Lee cualquier registro interno del CS5460A
;*          Empleando el registro TXDATA para indicar la direccion del registro que se
;*     desea leer, los registros RXDATA0,RXDATA1 y RXDATA2 se usan para almacenar
;*     el contenido de los registros intermos del CS5460A (3 Bytes)
;*************************************************************************
SPIREAD:        cbi     PORTB,PB4               ;/CS activo
                out     SPDR,TXDATA             ;Carga el comando a ejecutar en SPSR
r1:             sbis    SPSR,SPIF               ;Lo recorre 8 veces dentro del CS5460
                rjmp    r1                      ;
                ldi     temp,SYNC0              ;CARGA temp con SYNC0
                out     SPDR,temp               ;Carga Sync0 en SPSR
r2:             sbis    SPSR,SPIF               ;Lo recorre 8 veces dentro del CS5460
                rjmp    r2                      ;
                in      RXDATA0,SPDR            ;Lee el valor que fue reccorido dentro del puerto del avr
                in      RXDATA0,SPDR
                ldi     temp,SYNC0
                out     SPDR,temp
r3:             sbis    SPSR,SPIF
                rjmp    r3
                in      RXDATA1,SPDR
                in      RXDATA1,SPDR
                ldi     temp,SYNC0
                out     SPDR,temp
r4:             sbis    SPSR,SPIF
                rjmp    r4
                in      RXDATA2,SPDR
                in      RXDATA2,SPDR
                sbi     PORTB,PB4               ;/CS inactivo
                ret
;*************************************************************************
;*          EERead (lectura de la EEPROM del AT90S8515)
;* Esta subrutina espera hasta que la EEPROM este lista para ser programada
;* entonces lee el registro EErd con la direccion definida por EEard
;*************************************************************************
;***** Registros empleados por la subrutina

.def    EEdrd   =r0             ;Resultado
.def    EEard   =r16            ;Direccion a ser leida

EERead:         sbic    EECR,EEWE               ;Si EEWE no es cero
                rjmp    EERead                  ;Espera
                out     EEARL,EEard             ;Direccion de salida
```

```
                    sbi     EECR,EERE           ;Pone a uno el estrobo de lectura de la EEPROM
                    sbi     EECR,EERE           ;Nuevamente
                    in      EEdrd,EEDR          ;Obtiene el dato
                    ret

;*****************************************************************************
;*          KEYBO
;*          Inicializacion del puerto para el teclado
;*****************************************************************************
KEYBO:              cbi     DDRC,PC0            ;LSB entrada
                    cbi     DDRC,PC1            ;
                    cbi     DDRC,PC2            ;
                    cbi     DDRC,PC3            ;MSB entrada
                    sbi     DDRC,PC4            ;Output enable activa en bajo
                    ret
;*****************************************************************************
;*          TWIIC
;*          Rutinas I2C
;*****************************************************************************
delay1:             ldi     TWIdelay,10
delay1_loop:        dec     TWIdelay
                    brne    delay1_loop
                    ret
delay2:             ldi     TWIdelay,5
delay2_loop:        dec     TWIdelay
                    brne    delay2_loop
                    ret
;*****************************************************************************
TWI_rep_start:      sbi     DDRD,SCLP           ;SCL bajo
                    cbi     DDRD,SDAP           ;SDA alto
                    rcall   delay1              ;Retraso de medio periodo
                    cbi     DDRD,SCLP           ;SCL alto
                    rcall   delay2              ;Retraso de un quarto de periodo
;*****************************************************************************
TWI_start:          mov     TWIdata,TWIadr      ;Copia la direccion en el registro de transmision
                    sbi     DDRD,SDAP           ;SDA bajo
                    rcall   delay2              ;Retraso de un quarto de periodo
;*****************************************************************************
TWI_write:          sec                         ;Pone carry = 1
                    rol     TWIdata             ;Introduce carry
                    rjmp    TWI_write_first
TWI_write_bit:      lsl     TWIdata             ;Si el registro de transmision esta vacio
TWI_write_first:    breq    TWI_get_ack         ;Obtiene acknowledge
                    sbi     DDRD,SCLP           ;SCL bajo
                    brcc    TWI_write_low       ;Si el bit es alto
                    nop                         ;(Iguala numero de ciclos)
                    cbi     DDRD,SDAP           ;SDA alto
                    rjmp    TWI_write_high
```

```
TWI_write_low:      sbi     DDRD,SDAP               ;SDA bajo
                    rjmp    TWI_write_high          ;(Iguala numero de ciclos)
TWI_write_high:     rcall   delay1                  ;Retraso de medio periodo
                    cbi     DDRD,SCLP               ;SCL alto
                    rcall   delay1                  ;Retraso de medio periodo
                    rjmp    TWI_write_bit
;***************************************************************************
TWI_get_ack:
                    sbi     DDRD,SCLP               ;SCL bajo
                    cbi     DDRD,SDAP               ;SDA alto
                    rcall   delay1                  ;Retraso de medio periodo
                    cbi     DDRD,SCLP               ;SCL bajo
TWI_get_ack_wait:   sbis    PIND,SCLP               ;Espera SCL alto
                    rjmp    TWI_get_ack_wait
                    clc                             ;Limpia carry
                    sbic    PIND,SDAP               ;Si SDA alto
                    sec                             ;Carry = 1
                    rcall   delay1                  ;Retraso de medio periodo
                    ret
;***************************************************************************
TWI_do_transfer:    sbrs    TWIadr,b_dir            ;Si dir = write
                    rjmp    TWI_write               ;Salta a escribir dato
;***************************************************************************
TWI_read:           rol     TWIstat                 ;Guarda acknowledge
                                                    ;(Empleado por TWI_put_ack)
                    ldi     TWIdata,0x01            ;data = 0x01
TWI_read_bit:                                       ;
                    sbi     DDRD,SCLP               ;SCL bajo
                    rcall   delay1                  ;Retraso de medio periodo
                    cbi     DDRD,SCLP               ;SCL alto
                    rcall   delay1                  ;Retraso de medio periodo
                    clc                             ;Limpia carry
                    sbic    PIND,SDAP               ;SDA alto
                    sec                             ;Carry = 1
                    rol     TWIdata                 ;Almacena bit de datos
                    brcc    TWI_read_bit            ;Mientras no se llene el registro de recepcion
;***************************************************************************
TWI_put_ack:        sbi     DDRD,SCLP               ;SCL bajo
                    ror     TWIstat                 ;Obtiene bit de estado
                    brcc    TWI_put_ack_low         ;
                    cbi     DDRD,SDAP               ;SDA alto
                    rjmp    TWI_put_ack_high
TWI_put_ack_low:    sbi     DDRD,SDAP               ;SDA bajo
TWI_put_ack_high:   rcall   delay1                  ;Retraso de medio periodo
                    cbi     DDRD,SCLP               ;SCL alto
TWI_put_ack_wait:   sbis    PIND,SCLP               ;Espera SCL alto
                    rjmp    TWI_put_ack_wait
                    rcall   delay1                  ;Retraso de medio periodo
```

```
                ret
;*****************************************************************************
TWI_stop:       sbi     DDRD,SCLP               ;SCL bajo
                sbi     DDRD,SDAP               ;SDA bajo
                rcall   delay1                  ;Retraso de medio periodo
                cbi     DDRD,SCLP               ;SCL alto
                rcall   delay2                  ;Retraso de un quarto de periodo
                cbi     DDRD,SDAP               ;SDA alto
                rcall   delay1                  ;Retraso de medio periodo
                ret
;*****************************************************************************
TWI_init:       ldi     TWIstat,0xCF
                out     PORTD,TWIstat           ;Pines IIC como colector abierto
                clr     TWIstat
                out     DDRD,TWIstat            ;
                ret
;****Escribe un byte al reloj de tiempo real****

wr_data:        ldi     TWIadr,$D0+TWIwr        ;Pone direccion del dispositivo y comando de escritura
                rcall   TWI_start               ;Manda condicion de comienzo y direccion
                mov     TWIdata,numreg
                rcall   TWI_do_transfer         ;Ejecuta la transferencia
                mov     TWIdata,dato            ;Dato de escritura = 01010101b
                rcall   TWI_do_transfer         ;Ejecuta la transferencia
                rcall   TWI_stop                ;Manda condicion de paro
                ret
;****Lee un byte desde el reloj de tiempo real ****

rd_data:        ldi     TWIadr,$D0+TWIwr        ;Pone direccion del dispositivo y comando de escritura
                rcall   TWI_start               ;Manda condicion de comienzo y direccion
                mov     TWIdata,numreg          ;Escribe la direccion
                rcall   TWI_do_transfer         ;Ejecuta transferencia
                ldi     TWIadr,$D0+TWIrd        ;Pone direccion del dispositivo y comando de lectura
                rcall   TWI_rep_start           ;Envia nuevamente condicion de comienzo y direccion
                sec                             ;Indica no acknowledge
                rcall   TWI_do_transfer         ;Ejecuta transferencia (Lectura)
                rcall   TWI_stop                ;Envia condicion de paro
                ret
;*****************************************************************************
;*      LEER_RTC
;*      Lectura del reloj de tiempo real
;*****************************************************************************
LEER_RTC:       rcall   TWI_init                ;Inicializa interface IIC
                clr     ZH
                ldi     ZL,RTCS
                ldi     count,7
                ldi     numreg,0
```

```
                rcall   rd_rtc

                lds     temp,RTCS
                rcall   convhex
                sts     SEGUNDOS,temp

                lds     temp,RTCM
                rcall   convhex
                sts     MINUTOS,temp
                lds     temp,RTCH
                mov     count,temp
                andi    temp,0x1F
                sts     RTCH,temp
                rcall   convhex
                sts     HORAS,temp

                andi    count,0x20
                cpi     count,0
                brne    set_pm
                sts     AMPM,count

rtc_pm:         lds     temp,RTCDATE
                rcall   convhex
                sts     FECHA,temp

                lds     temp,RTCMES
                rcall   convhex
                sts     MES,temp

                lds     temp,RTCANI
                rcall   convhex
                sts     ANNO,temp

finrtc:         ret

set_pm:         ldi     count,1
                sts     AMPM,count
                rjmp    rtc_pm


;*********************************************************
rd_rtc:         rcall   rd_data
                st      Z+,TWIdata
                inc     numreg
                dec     count
                brne    rd_rtc
                ret
;*********************************************************
```

```
wr_rtc:         ld      dato,Z+
                rcall   wr_data
                inc     numreg
                dec     count
                brne    wr_rtc
                ret
;*********************************************************
convhex:
                push    temp
                swap    temp
                andi    temp,0x0F
                lsl     temp
                mov     temp2,temp
                lsl     temp
                lsl     temp
                add     temp2,temp
                pop     temp
                andi    temp,0x0F
                add     temp,temp2
                ret

;*************************************************************************
;*      LEER_NVRAM
;*      Lectura de datos almacenados en la NVRAM del reloj de tiempo real
;*      Esta rutina se utiliza solo cuando se inicializa el sistema
;*************************************************************************
LEER_NVRAM:     rcall   TWI_init                ;Inicializa interface IIC
                clr     ZH                      ;Limpia ZH
                ldi     ZL,ENERGYH              ;ZL apunta a la primera direccion de energia
                ldi     count,16                ;Se van a leer 16 registros
                ldi     numreg,0x08             ;A partir de la direccion 0x08H
                rcall   rd_rtc
                ldi     ZH,HIGH(ENERRESH)       ;
                ldi     ZL,LOW(ENERRESL)        ;Z apunta a la direccion donde se encuentra el resto de los registros
                ldi     count,8
                ldi     numreg,0x18
                rcall   rd_rtc
                ret
;*************************************************************************
;*
;*      WE_NVRAM
;*
;*      Almacena los datos de energia en la NVRAM del reloj tiempo real
;*  Esta rutina es llamada de manera periodica para almacenar los datos
;*************************************************************************
WE_NVRAM:
                rcall   TWI_init                ;Inicializa interface IIC
                clr     ZH                      ;Limpia ZH
```

```
                ldi     ZL,ENERGYH              ;ZL apunta a la primera direccion de energia
                ldi     count,16                ;Se va a escribir en 4 registros
                ldi     numreg,0x08             ;A partir de la direccion 0x08H
                rcall   wr_rtc
                ldi     ZH,HIGH(ENERRESH)
                ldi     ZL,LOW(ENERRESL)        ;Z apunta a la direccion donde se encuentra el resto de los registros
                ldi     count,8
                ldi     numreg,0x18
                rcall   wr_rtc
                ret
;*****************************************************************************
;*      CKYEAR
;*      Verifica el cambio de año para poner a cero los contadores de meses en la NVRAM y en la RAM del AVR
;*      Los contadores de meses llevan la cuenta del numero de transacciones efectuadas por mes
;*      Si se detecta un cambio de año de debe de incrementar ANNREF y guardarlo en la EEPROM y RAM del AVR
;*****************************************************************************
CKYEAR:         lds     temp,ANNO               ;Carga el año en binario, siempre es mayor o igual a ANNREF
                lds     temp2,ANNREF            ;Esta variable es igual a año-n donde n es el ultimo año en que estuvo operando el equipo
                sub     temp,temp2              ;ANNO-ANNREF, si la diferencia es mayor que 2, entonces limpia los registros de meses
                cpi     temp,1                  ;Compara para saber si la diferencia es de uno
                brge    endck                   ;Si es uno sale de la rutina
                clr     temp
                clr     ZH
                ldi     ZL,NVMESES              ;El registro Z apunta al primer valor de meses
                ldi     count,12                ;Se van a limpiar 12 registros
sigmes:         st      Z+,temp                 ;Almacena ceros en las direcciones de RAM apuntadas por Z e incrementa el apuntador
                dec     count                   ;Decrementa el contaor
                brne    sigmes                  ;Salta mientras no limpie los 12 registros
                clr     ZH
                ldi     ZL,NVMESES
                ldi     count,12
                ldi     numreg,0x12             ;Apunta a partir de la direccion 12 de la NVRAM
                rcall   wr_rtc                  ;Escribe los valores en la NVRAM
endck:          lds     temp,ANNO               ;Esta parte la ejecuta para actualizar ANNREF en caos de que el equipo no haya funcionado en varios años
                dec     temp
                sts     ANNREF,temp
                ret
;*****************************************************************************
;*      COPIA_D
;*      Hace una copia de la llave D almacenada en VALOR_D a partir de las localidades de memoria en D_TEMP
;*      ya que D_TEMP es modificado cada que se realiza una desencriptacion
;*****************************************************************************
COPIA_D:        clr     YH
                clr     ZH
                ldi     YL,VALOR_D              ;Y apunta a la posicion desde donde se copiaran los datos
                ldi     ZL,D_TEMP               ;Z apunta a la posicion donde se copiaran los datos
                ldi     count,9                 ;Contador con el numero de datos a copiar
```

```
nxtd:           ld      temp,Y+             ;Carga en temp el valor a copiar y apunta al siguiente
                st      Z+,temp             ;Guarda temp en la direccion a la cual apunta Z y se autoincrementa
                dec     count               ;Decrementa en uno el contador
                brne    nxtd                ;Si no es cero continua copiando datos
                ret
;***************************************************************************
;*
;*      INIT_M
;*
;*      Inicializa la variables almacenadas en VALOR_M.
;*  En las primeras 8 direcciones carga ceros y en la ultima un uno
;***************************************************************************
INIT_M:         clr     ZH
                ldi     ZL,VALOR_M          ;Z apunta a la primera direccion de VALOR_M
                ldi     count,8             ;Contador para escribir los primeros ocho ceros
                clr     temp                ;Pone temp a cero
nxtm:           st      Z+,temp             ;Almacena un cero a la direccion apuntada por Z y se autoincrementa
                dec     count               ;Decrementa en uno el contador
                brne    nxtm                ;Si no es cero continua poniendo ceros en VALOR_M
                ldi     temp,1              ;Carga un uno en temp
                st      Z,temp              ;Almacena un uno en la ultima localidad de VALOR_M
                ret
;***************************************************************************
;*      DESENCRIPTA
;*      Desencripta el numero introducido por el usuario
;***************************************************************************
DESENCRIPTA:

;Registros y definiciones empleadas para las multiplicaciones de 9*9 bytes

.def    res1    =r0     ;result byte 0          (LSByte)
.def    res2    =r1     ;result byte 1
.def    res3    =r2     ;result byte 2
.def    res4    =r3     ;result byte 3
.def    res5    =r4     ;result byte 4
.def    res6    =r5     ;result byte 5
.def    res7    =r6     ;result byte 6
.def    res8    =r7     ;result byte 7
.def    res9    =r8     ;result byte 8
.def    res10   =r9     ;result byte 9
.def    res11   =r10    ;result byte 10
.def    res12   =r11    ;result byte 11
.def    res13   =r12    ;result byte 12
.def    res14   =r13    ;result byte 13
.def    res15   =r14    ;result byte 14
.def    res16   =r15    ;result byte 15
.def    res17   =r16    ;result byte 16
.def    res18   =r17    ;result byte 17          ;Este es el MSByte
```

```
.def    mc0     =r18                    ;multiplicand low byte
.def    mc1     =r19
.def    mc2     =r20
.def    mc3     =r21
.def    mc4     =r22
.def    mc5     =r23
.def    mc6     =r24
.def    mc7     =r25
.def    mc8     =r26                    ;multiplicand high byte

.def    temp_d  =r18
.def    cont_d=r27                      ;Para la desencriptacion

;Registros y definiciones empleadas para las divisiones de 18/9 bytes

.def    dd1     =r0                     ;Byte menos significativo del dividendo
.def    dd2     =r1
.def    dd3     =r2
.def    dd4     =r3
.def    dd5     =r4
.def    dd6     =r5
.def    dd7     =r6
.def    dd8     =r7
.def    dd9     =r8
.def    dd10    =r9
.def    dd11    =r10
.def    dd12    =r11
.def    dd13    =r12
.def    dd14    =r13
.def    dd15    =r14
.def    dd16    =r15
.def    dd17    =r16
.def    dd18    =r17                    ;Byte mas significativo del dividendo

.def    tempdiv =r19                    ;Registros para algunas operaciones temporales
.def    contdiv =r20                    ;Registros para algunas operaciones temporales


;D>0

dn_cero:        clr     ZH              ;Limpia ZH
                ldi     ZL,D_TEMP       ;ZL apunta al byte mas significativo de D_TEMP
otr_d:          ld      temp_d,Z+       ;Carga cada byte de D desde el mas significativo
                cpi     temp_d,0        ;Lo compara con 0
                brne    d_next          ;Si no es cero continua la desencriptacion
                dec     cont_d          ;Como el byte anterior de D fue cero continua revisando los demas bytes
                brne    otr_d           ;
```

```
                ret                             ;Si D==0 termina la desencriptacion ya que todos los bytes de D fueron cero

;*D mod 2

d_next:         clr     ZH                      ;Limpia ZH
                ldi     ZL,D_TEMP               ;Carga cada byte de D desde el mas significativo
                ld      temp_d,Z+               ;Carga el byte mas significativo
                lsr     temp_d                  ;Lo divide entre 2 (introduce un cero hacia la derecha) y el LSBit lo introduce al carry
                ldi     cont_d,8                ;Carga el contador para procesar los siguientes 8 bytes
s_rorm:         ld      temp_d,Z+               ;Carga los siguientes bytes a procesar
                ror     temp_d                  ;Introduce el carry por el MSbit e introduce el LSbit al carry
                dec     cont_d                  ;Decrementa la cuenta de bytes a procesar
                brne    s_rorm                  ;Salta si aun no ha procesado los 8 bytes
                brcc    no_mod                  ;Si el carry es 0 quiere decir que (Dmod2==0) y salta a no_mod

                clr     YH                      ;Esta parte del codigo carga el primer multiplicando M en los registros definidos para poder multiplicar por C
                ldi     YL,27                   ;YL apunta a r26+1
                clr     ZH
                ldi     Zl,VALOR_M              ;ZL apunta al byte mas significativo de VALOR_M
                ldi     cont_d,9                ;Se emplea el contador para mover los 9 bytes de VALOR_M a los registros MC0...MC8(r18-r26)
lo_vam:         ld      temp_d,Z+               ;Carga valores desde el mas significativo de VALOR_M
                dec     cont_d                  ;
                brne    lo_vam                  ;
                ldi     r17,0
                rcall   x_cmodn                 ;Ejecuta M=(M*C)mod N
                clr     YH                      ;Mueve el reminder a VALOR_M
                ldi     YL,reminder             ;Apunta a reminder
                clr     ZH
                ldi     ZL,VALOR_M
                ldi     cont_d,9
mov_rm:         ld      temp_d,Y+
                st      Z+,temp_d
                dec     cont_d
                brne    mov_rm

;*D=D-1
                clr     ZH                      ;Limpia ZH
                ldi     ZL,D_TEMP+9             ;ZL apunta al byte delante de D_TEMP (menos significativo)
                ld      temp_d,-Z               ;Carga el byte menos significativo de D_TEMP en temp
                subi    temp_d,1                ;Le resta al byte menos significativo 1
                st      Z,temp_d
                ldi     cont_d,8                ;Carga en cont_d el numero restante de bytes a procesar
otr_re:         ld      temp_d,-Z               ;Carga en temp el siguiente byte a procesar
                sbci    temp_d,0                ;Resta con carry
                st      Z,temp_d                ;Guarda el nuevo valor de D_TEMP despues de cada resta
                dec     cont_d                  ;Decrementa la cuenta de bytes a procesar
                brne    otr_re                  ;Resta el siguiente byte
```

```
no_mod:     ldi     r17,1           ;Indica la copia de los valores del segundo multiplicando al primero
            rcall   x_cmodn         ;Ejecuta C=(C*C)mod N
            clr     YH              ;Mueve el reminder a VALOR_C
            ldi     YL,reminder     ;Apunta a reminder
            clr     ZH
            ldi     ZL,VALOR_C
            ldi     cont_d,9
mov_rc:     ld      temp_d,Y+
            st      Z+,temp_d
            dec     cont_d
            brne    mov_rc          ;Mover el reminder a VALOR_C
;*D/2
            clr     ZH              ;Limpia ZH
            ldi     ZL,D_TEMP       ;Carga cada byte de D desde el mas significativo
            ld      temp_d,Z        ;Carga el byte mas significativo
            lsr     temp_d          ;Lo divide entre 2 (introduce un cero hacia la derecha) y el LSBit lo introduce al carry
            st      Z+,temp_d       ;Guarda el primer byte dividido entre 2 y apunta al siguiente byte
            lds     cont_d,8        ;Carga el contador para procesar los siguientes 8 bytes
s_rord:     ld      temp_d,Z        ;Carga los siguientes bytes a procesar
            ror     temp_d          ;Introduce el carry por el MSbit e introduce el LSbit al carry
            st      Z+,temp_d       ;Guarda los siguientes bytes divididos
            dec     cont_d          ;Decrementa la cuenta de bytes a procesar
            brne    s_rord          ;Salta si aun no ha procesado los 8 bytes
            rjmp    dn_cero         ;Salta al comienzo de la rutina para continuar el proceso mientras que D>0

x_cmodn:  push  temp_d
            clr     YH              ;Esta parte del codigo carga el segundo multiplicando (que siempre es C -VALOR_C-)
            ldi     YL,9            ;YL apunta a r8+1
            clr     ZH
            ldi     Zl,VALOR_C      ;ZL apunta al byte mas significativo de VALOR_M
            ldi     cont_d,9        ;Se emplea el contador para mover los 9 bytes de VALOR_M a los registros MC0...MC8(r18-r26)
lo_vac:     ld      temp_d,Z+       ;Carga valores desde el mas significativo de VALOR_M y apunta al siguiente
            st      -Y,temp_d       ;Guarda el VALOR_M en los registros r26..r25...r18 desde el mas significativo
            dec     cont_d          ;
            brne    lo_vac          ;
            pop     temp_d

            cpi     r17,1           ;Compara para saber si los 2 factores a multiplicar son iguales
            brne    bmul            ;Si no son iguales quiere decir que se esta multiplicando MxC y salta esta seccion
            mov     mc8,res9        ;Si son iguales hace una copia en estos registros
            mov     mc7,res8
            mov     mc6,res7
            mov     mc5,res6
            mov     mc4,res5
            mov     mc3,res4
            mov     mc2,res3
            mov     mc1,res2
            mov     mc0,res1
```

```
bmul:   clr   res18      ;Comienza a hacer la multiplicacion de 9x9 bytes
        clr   res17      ;Limpia los 9 bytes mas significativos del resultado de la multiplicacion
        clr   res16
        clr   res15
        clr   res14
        clr   res13
        clr   res12
        clr   res11
        clr   res10

        ldi   cont_d,72  ;cont_d se va a emplear como el contador general de esta rutina
        lsr   res9       ;Comienza a procesar el segundo multiplicando C
        ror   res8
        ror   res7
        ror   res6
        ror   res5
        ror   res4
        ror   res3
        ror   res2
        ror   res1

m18_1:  brcc  noad18     ;if bit 0 of multiplier set

        add   res10,mc0  ;add multiplicand Low to byte 2 of res
        adc   res11,mc1  ;add multiplicand high to byte 3 of res
        adc   res12,mc2  ;add multiplicand high to byte 3 of res
        adc   res13,mc3  ;add multiplicand high to byte 3 of res
        adc   res14,mc4  ;add multiplicand high to byte 3 of res
        adc   res15,mc5  ;add multiplicand high to byte 3 of res
        adc   res16,mc6  ;add multiplicand high to byte 3 of res
        adc   res17,mc7  ;add multiplicand high to byte 3 of res
        adc   res18,mc8  ;add multiplicand high to byte 3 of res


noad18: ror   res18
        ror   res17
        ror   res16
        ror   res15
        ror   res14
        ror   res13
        ror   res12
        ror   res11
        ror   res10
        ror   res9       ;shift right result byte 3
        ror   res8       ;rotate right result byte 2
```

```
                ror     res7                    ;rotate result byte 1 and multiplier High
                ror     res6                    ;rotate result byte 0 and multiplier Low
                ror     res5
                ror     res4
                ror     res3
                ror     res2
                ror     res1
                dec     cont_d                  ;Decrementa el contador general
                brne    m18_1                   ;Si no ha procesado los 72 bits salta para seguir procesando

;*************************************************************************
;*Comienza la division, el resultado que interesa se encuentra en el reminder
;*************************************************************************

                clr     ZH                      ;Apuntador a reminder
                ldi     ZL,REMINDER             ;Z apunta a la primer localidad de reminder
                ldi     cont_d,9    ;
                clr     temp_d                  ;Carga las localidades 1 al 9 del reminder (de numeros) con 0
ot_rem:         st      Z+,temp_d
                dec     cont_d
                brne    ot_rem

                clc                             ;Limpia carry
                ldi     cont_d,145              ;Inicializa el contador
d16u_1:         rol     dd1                     ;Shift left dividend
                rol     dd2
                rol     dd3
                rol     dd4
                rol     dd5
                rol     dd6
                rol     dd7
                rol     dd8
                rol     dd9
                rol     dd10
                rol     dd11
                rol     dd12
                rol     dd13
                rol     dd14
                rol     dd15
                rol     dd16
                rol     dd17
                rol     dd18
                dec     cont_d                  ;Decrementa el contador
                brne    d16u_2                  ;Si termino
                ret                             ;Regresa

d16u_2:         clr     ZH
```

```
                ldi     ZL,reminder+9
                ldi     contdiv,9
remy:           ld      temp_d,-Z
                rol     temp_d
                st      Z,temp_d
                dec     contdiv
                brne remy
                clr     YH
                ldi     YL,reminder+9           ;
                clr     ZH
                ldi     Zl,valor_n+9            ;Apunta al byte menos significativo de ntemp checar el decremento
                ld      temp_d,-Z
                ld      tempdiv,-Y
                st      Y,tempdiv               ;Guarda de donde saco rem1 y apunta a la sig
                ldi     contdiv,8               ;Solo procesa 8 restas
otro_d:         ld      temp_d,-Z
                ld      tempdiv,-Y              ;En tempdiv se guardan los valores de los registros rem2-rem9
                sbc     tempdiv,temp_d          ;Resta con acarreo
                st      Y,tempdiv
                dec     contdiv
                brne    otro_d
                brcc    d16u_3                  ;Si el resultado es negativo
                clr     YH
                ldi     YL,reminder+9           ;
                clr     ZH
                ldi     Zl,VALOR_N+9            ;Apunta al byte menos significativo de ntemp checar el decremento
                ld      temp_d,-Z
                ld      tempdiv,-Y
                add     tempdiv,temp_d          ;Restablece el residuo
                st      Y,tempdiv
                ldi     contdiv,8               ;Solo procesa 8 restas
res_d:          ld      temp_d,-Z
                ld      tempdiv,-Y              ;En tempdiv se guardan los valores de los registros rem2-rem9
                adc     tempdiv,temp_d          ;Suma con acarreo
                st      Y,tempdiv
                dec     contdiv
                brne    res_d

                clc                             ;Y limpia el acarreo para introducirlo al resultado (es decir cuando el denominador es mas pequeño que el numerador)
                rjmp    d16u_1                  ;Si no.....
d16u_3:  sec                                    ;Pone a 1 la bandera de acarreo y lo introduce en el resultado (division exitosa, puede seguir habiendo o no acarreo)
                rjmp    d16u_1
                                                ;Se puede emplear este mismo algoritmo para un numero mayor de bits
```

```
;*****************************************************************************
;*
;*          CODTOBIN
;*
;*          Convierte el codigo tecleado por el usuario (21 digitos en decimal) a binario (72 bits maximo)
;*
;*****************************************************************************
CODTOBIN:

.def        rescv1      =r0             ;rescv1 a rescv9 es donde se guardara el numero de 20 digitos en binario
.def        rescv2      =r1
.def        rescv3      =r2
.def        rescv4      =r3
.def        rescv5      =r4
.def        rescv6      =r5
.def        rescv7      =r6
.def        rescv8      =r7
.def        rescv9      =r8
.def        rescv10     =r9

.def        mpdb1       =r10            ;mpdb1 a mpdb5 se emplean para guardar el resultado parcial de la multiplicacion por 10, 10E1,10E2...etc
.def        mpdb2       =r11
.def        mpdb3       =r12
.def        mpdb4       =r13
.def        mpdb5       =r14
.def        mpdb6       =r15
.def        mpdb7       =r16
.def        mpdb8       =r17
.def        mpdb9       =r18
.def        mpdb10      =r19

.def        mcdb        =r20            ;Se emplea como contador y como multiplicando cada que se multiplique por 10
.def        cicles      =r21            ;
.def        cont1       =r22
.def        cont2       =r23

            clr         rescv2          ;Limpiar los registros antes comenzar operaciones
            clr         rescv3
            clr         rescv4
            clr         rescv5
            clr         rescv6
            clr         rescv7
            clr         rescv8
            clr         rescv9


            clr         ZH                          ;Apuntador a numeros para procesarlos sucesivamente
            ldi         ZL,CODNASCII                ;Apunta al CODNASCII donde se guardaron los numeros en decimal
```

```
        ldi     temp,1                  ;Registro auxiliar para multiplicar por 10, 10E1, 10E2 etc.
        ldi     cicles,20               ;Carga cicles con el numero de digitos a procesar (21)
        ldi     cont1,0
        ld      rescv1,Z+               ;Carga el primer digito en rescv1
Sdig:   ld      mpdb1,Z+                ;Carga siguiente digito en mp1
        clr     mpdb2                   ;Limpia los registros antes de procesar otro digito
        clr     mpdb3
        clr     mpdb4
        clr     mpdb5
        clr     mpdb6
        clr     mpdb7
        clr     mpdb8
        clr     mpdb9
        clr     mpdb10
;*************************************************************************
MX10:   ldi     cont2,72                ;Numero de corrimientos para multiplicar 72X8 bits
        ldi     mcdb,10                 ;El multiplicando siempre es 10

        lsr     mpdb10
        ror     mpdb9
        ror     mpdb8
        ror     mpdb7
        ror     mpdb6
        ror     mpdb5
        ror      mpdb4
        ror     mpdb3
        ror     mpdb2
        ror     mpdb1

M72X8:  brcc    noad72
        add     mpdb10,mcdb

noad72: ror     mpdb10
        ror     mpdb9
        ror     mpdb8
        ror     mpdb7
        ror     mpdb6
        ror     mpdb5
        ror     mpdb4
        ror     mpdb3
        ror     mpdb2
        ror     mpdb1
        dec     cont2
        brne    M72X8
;*************************************************************************
        inc     cont1                   ;Incrementa a uno el contador
        cp      cont1,temp              ;Compara para saber si ya multiplico por 10 las veces necesarias
```

```
        brne    MX10

        clr     cont1                   ;Pone a cero cont1
        add     rescv1,mpdb1            ;Suma los resultados
        adc     rescv2,mpdb2
        adc     rescv3,mpdb3
        adc     rescv4,mpdb4
        adc     rescv5,mpdb5
        adc     rescv6,mpdb6
        adc     rescv7,mpdb7
        adc     rescv8,mpdb8
        adc     rescv9,mpdb9
        adc     rescv10,mpdb10
        inc     temp                    ;Aumenta el numero de veces a multiplicar por 10
        dec     cicles                  ;Verifica si ya se procesaron los 20 digitos
        brne    Sdig                    ;No, toma es siguiente digito

        sts     VALOR_C,rescv9          ;Mueve el resultado de la conversion a VALOR_C desde el mas significativo en la primer posicion
        sts     VALOR_C+1,rescv8
        sts     VALOR_C+2,rescv7
        sts     VALOR_C+3,rescv6
        sts     VALOR_C+4,rescv5
        sts     VALOR_C+5,rescv4
        sts     VALOR_C+6,rescv3
        sts     VALOR_C+7,rescv2
        sts     VALOR_C+8,rescv1

        ret
;*****************************************************************************
;*
;*      BINTOCOD
;*
;*      Convierte el codigo desencriptado nuevamente a decimal
;*      Para que convierta los numeros correctamente
;*****************************************************************************
BINTOCOD:

.def    dbd1    =r0     ;Cargar aqui el byte menos significativo de la desencriptacion
.def    dbd2    =r1
.def    dbd3    =r2
.def    dbd4    =r3
.def    dbd5    =r4
.def    dbd6    =r5
.def    dbd7    =r6
.def    dbd8    =r7
.def    dbd9    =r8     ;Cargar aqui el byte mas significativo de la desencriptacion

.def    rem1    =r11
```

```
.def    rem2    =r12
.def    rem3    =r13
.def    rem4    =r14
.def    rem5    =r15

.def    div1    =r20
.def    dcnt16u =r21
.def    cont1   =r22

        ldi     div1,0x0A               ;Carga el divisor (10)

        lds     dbd9,VALOR_M            ;Carga VALOR_M para poder convertirlo a decimal desde al mas significativo
        lds     dbd8,VALOR_M+1
        lds     dbd7,VALOR_M+2
        lds     dbd6,VALOR_M+3
        lds     dbd5,VALOR_M+4
        lds     dbd4,VALOR_M+5
        lds     dbd3,VALOR_M+6
        lds     dbd2,VALOR_M+7
        lds     dbd1,VALOR_M+8

        ldi     cont1,20                ;Numero de digitos a procesar pueden ser hasta 21 "checar"
        clr     ZH                      ;Limpia ZH
        ldi     ZL,VALOR_MDEC           ;Z apunta a VALOR_MDEC donde se guardara el numero desencriptado en decimal

sidiv:  rcall   divbd
        st      Z+,rem1
        dec     cont1
        brne    sidiv
        ret


divbd:  clr     rem1                    ;Limpia byte bajo del residuo
        clc                             ;Limpia carry

        ldi     dcnt16u,73              ;init loop counter
d40_1:  rol     dbd1                    ;shift left dividend
        rol     dbd2
        rol     dbd3
        rol     dbd4
        rol     dbd5
        rol     dbd6
        rol     dbd7
        rol     dbd8
        rol     dbd9

        dec     dcnt16u                 ;decrement counter
        brne    d40_2                   ;if done
```

```
        ret                                     ;return
d40_2:  rol     rem1                            ;shift dividend into remainder
        sub     rem1,div1                       ;remainder = remainder - divisor
        add     rem1,div1                       ;restore remainder
        clc                                     ;clear carry to be shifted into result
        rjmp    d40_1                           ;else
d40_3:  sec                                     ;set carry to be shifted into result
        rjmp    d40_1

;***************************************************************************
;*      SWAPCOD
;*      Invierte el orden en que estan guardados hasta 21 digitos en RAM (introducidos por el usuario)
;*      Se emplea en conjunto con las rutinas de conversion
;***************************************************************************
SWAPCOD:        clr     YH
                clr     ZH
                ldi     YL,CODNASCII            ;Y apunta a CODNASCII
                ldi     ZL,CODNASCII+21         ;Z apunta a la ultima posicion de CODNASCII ya que se utilizara predecremento en el apuntador
                ldi     count,10                ;Numero de pares de digitos que se moveran
sigswap:        ld      temp,Y                  ;Carga en orden ascendente los datos en temp
                ld      temp2,-Z                ;Carda en orden descendente los datos en temp2
                st      Y+,temp2                ;Realiza el primer intercambio
                st      Z,temp                  ;Realiza el segundo intercambio
                dec     count                   ;Decrementa la cuenta
                brne    sigswap
                ret

;***************************************************************************
;*      SWAPCODM
;*      Invierte el orden en que estan guardados 20 digitos en RAM (resultado de la desencriptacion)
;*      Se emplea antes de verificar el numero desencriptado
;***************************************************************************
SWAPCODM:       ldi     ZH,high(VALOR_MDEC+20)
                ldi     ZL,low(VALOR_MDEC+20)
                ldi     YH,high(VALOR_MDEC)
                ldi     YL,LOW(VALOR_MDEC)
                ldi     count,10                ;Numero de pares de digitos que se moveran
sigswap2:       ld      temp,Y                  ;Carga en orden ascendente los datos en temp
                ld      temp2,-Z                ;Carda en orden descendente los datos en temp2
                st      Y+,temp2                ;Realiza el primer intercambio
                st      Z,temp                  ;Realiza el segundo intercambio
                dec     count                   ;Decrementa la cuenta
                brne    sigswap2
                ret
```

```
;***************************************************************************
;*      VALIDECRYPT
;*      Verifica que se hayan tecleado 21 digitos para poder comenzar la desencriptacion y
;*      si no se han tecleado 21 digitos manda mensaje de error
;*      Si el codigo es invalido manda mensaje con el numero de intentos restantes (tres intentos por dia maximo)
;*      Si el codigo es valido abona credito y activa el servicio
;***************************************************************************
VALIDECRYPT:
            lds     temp,cuantos            ;Carga en temp el numero de digitos tecleados
            cpi     temp,21                 ;Los compara con 21
            breq    okstart                 ;Si son 21 digitos comienza el proceso de desencriptacion
            ldi     temp,4                  ;Carga en count un 4
            sts     TIMEOUT,temp            ;El 3 es cargado en TIMEOUT para que despliegue el mensaje de error por 2 segundos minimo
            ldi     temp,8                  ;Carga un ocho en temp
            sts     BANDERA,temp            ;Guarda el ocho en bandera para poder refrescar el mensaje de error cada segundo
            clr     temp                    ;Pone un cero en temp
            sts     SDECRYPT,temp           ;Pone un cero en SDECRYPT para indicar que no se esta en modo de desencriptacion
            ldi     temp,1                  ;Carga un uno en temp
            sts     REFRESH,temp            ;Pone a uno REFRESH para que imprima lo mas pronto el mensaje de error
            ret                             ;Termina la rutina

okstart:    lds     temp,TRIES
            inc     temp
            sts     TRIES,temp
            cpi     temp,4
            brge    codefail                ;
            rcall   INIT_M                  ;Inicializa la variable M a 1 (M=1) antes de comenzar la desencriptacion
            rcall   COPIA_D                 ;copia los datos de VALOR_D a D_temp antes de cada desencriptacion
            rcall   SWAPCOD                 ;Prepara el codigo introducido para convertirlo a binario
            rcall   CODTOBIN                ;Convierte el codigo a binario

            rcall   DESENCRIPTA             ;Desencripta el numero tecleado por el usuario
            rcall   BINTOCOD                ;Convierte a decimal el numero desencriptado
            rcall   SWAPCODM                ;Invierte el orden del numero desencriptado almacenado en RAM
            rcall   VALIDNUM                ;Checa si el numero introducido es valido
            cpi     temp,1                  ;Compara si es cero
            brne    codefaila               ;Si es cero salta para indicar que el codigo introducido por el usuario no es valido
            rcall   CVRCREDIT               ;Convierte a binario el credito abonado al sistema
            ldi     temp,1
            sts     SERVICIO,temp           ;Bandera para indicar la reactivacion del servicio,
            clr     temp
            sts     TRIES,temp              ;Despues de una desencriptacion exitosa limpia el contador de intentos validos
            sts     cuantos,temp            ;Limpia el contador de los numeros tecleados
            sts     SDECRYPT,temp
            rcall   WE_NVRAM                ;Guarda inmediatamente todas las variables de energia modificadas
            ret
codefail:   clr     temp                    ;Carga un cero en temp
            sts     TRTIME,temp             ;Limpia el contador para que cuente las horas que permanecera desactivado el servicio
```

```
                ldi     temp,1                  ;Carga un uno en temp
                sts     DEACTKEY,temp           ;Pone a uno el indicador para desactivar el teclado
                rcall   WE_NVRAM                ;Guarda inmediatamente todas las variables de energia modificadas
codefaila:      ldi     temp,4                  ;Carga en count un 4
                sts     TIMEOUT,temp            ;El 4 es cargado en TIMEOUT para que despliegue el mensaje de error por 3 segundos minimo
                ldi     temp,9                  ;Carga un nueve en temp
                sts     BANDERA,temp            ;Guarda el nueve en bandera para poder refrescar el mensaje de error cada segundo
                clr     temp                    ;Pone un cero en temp
                sts     SDECRYPT,temp           ;Pone un cero en SDECRYPT para indicar que no se esta en modo de desencriptacion
                sts     cuantos,temp            ;Limpia el contador de los numeros tecleados
                ldi     temp,1                  ;Carga un uno en temp
                sts     REFRESH,temp            ;Pone a uno REFRESH para que imprima lo mas pronto el mensaje de error

                ret

;*****************************************************************************
;*      VALIDNUM
;*      Verifica que el numero introducido sea valido
;*       En caso de que sea valido pone a uno la bandera de ABONAR
;*****************************************************************************
VALIDNUM:       rcall   LDSERIAL                ;Carga el numero de serie del equipo en RAM en la direccion SERIALN
                lds     temp,VALOR_MDEC+2       ;Carga el digito de decimas de año desencriptado
                lds     temp2,VALOR_MDEC+3      ;Carga el digito de unidades de año desencriptado
                swap    temp                    ;Invierte el orden de los bytes de decimas de año
                or      temp,temp2              ;Junta las decimas de año y las unidades en un solo registro en BCD
                lds     temp2,RTCANI            ;Carga en temp2 el año actual
                brne    wrngnum                 ;Si no coincide brinca a la etiqueta wrngnum
                rcall   CKSERIAL                ;Revisa que el numero de serie del sistema sea igual al tecleado
                cpi     temp,1                  ;Verifica si temp es uno
                brne    wrngnum                 ;Si no es uno salta a wrngnum ya que el numero de serie no es valido
                rcall   CKVERID                 ;Revisa el digito verificador
                cpi     temp,1                  ;Verifica si temp es uno
                brne    wrngnum                 ;Si no es uno salta a wrngnum
                ldi     temp,1
                ret
wrngnum:        clr     temp
                ret

;*****************************************************************************
;*      LDSERIAL
;*      Carga el numero de serie del equipo en RAM en la localidad SERIALN
;*      Carga el numero desde el digito mas significativo (nueve digitos)
;*      Se pude modificar para aceptar cualquier numero de serie
;*****************************************************************************
LDSERIAL:       ldi     temp,0x09
                sts     SERIALN,temp
                ldi     temp,0x08
                sts     SERIALN+1,temp
```

```
                ldi     temp,0x07
                sts     SERIALN+2,temp
                ldi     temp,0x06
                sts     SERIALN+3,temp
                ldi     temp,0x05
                sts     SERIALN+4,temp
                ldi     temp,0x04
                sts     SERIALN+5,temp
                ldi     temp,0x03
                sts     SERIALN+6,temp
                ldi     temp,0x02
                sts     SERIALN+7,temp
                ldi     temp,0x01
                sts     SERIALN+8,temp
                ret
;*************************************************************************
;*      CKSERIAL
;*      Lee el numero de serie del equipo en RAM en la localidad SERIALN
;*      y lo compara con el numero de serie desencriptado
;*      si la comparacion es exitosa carga temp con uno y de lo contrario con cero
;*************************************************************************
CKSERIAL:       ldi     ZH,high(SERIALN)
                ldi     ZL,low(SERIALN)         ;Z apunta al numero de serie almacenado en el sistema
                ldi     YH,high(VALOR_MDEC+6)
                ldi     YL,LOW(VALOR_MDEC+6)    ;Y apunta al numero de serie desencriptado

                ldi     count,9                 ;Carga count con el numero de digitos a comparar
nxserie:        ld      temp,Z+                 ;Carga un digito del numero de serie en temp y apunta al siguiente
                ld      temp2,Y+                ;Carga un digito desencriptado en temp2 y apunta al siguiente
                cp      temp,temp2              ;Compara ambos registros
                brne    ser_nok                 ;Si no son iguales salta a ser_nok
                dec     count                   ;Decrementa la cuenta
                brne    nxserie                 ;Si la cuenta no es cero procesa los siguientes numeros
                ldi     temp,1                  ;Como el numero de serie es valido carga un uno en temp
                ret
ser_nok:        clr     temp                    ;Como el numero de serie no es valido carga un cero en temp
                ret
;*************************************************************************
;*      CKVERID
;*      Verifica que el digito verificador coincide con el del sistema
;*      si la comparacion es exitosa carga temp con uno y de lo contrario con cero
;*************************************************************************
CKVERID:        lds     temp,VALOR_MDEC         ;Carga el digito de decimas de mes desencriptado
                lds     temp2,VALOR_MDEC+1      ;Carga el digito de unidades de mes desencriptado
                swap    temp                    ;Invierte el orden de los bytes de decimas de mes
                or      temp,temp2              ;Junta las decimas de mes y las unidades en un solo registro en BCD
                rcall   convhex                 ;Convierte el mes a hexadecimal y lo guarda en temp
                sts     DECMES,temp             ;Guarda el mes desencriptado en la variable DECMES
```

```
                lds     temp,VALOR_MDEC+4       ;Carga el digito de decimas del numero verificador
                lds     temp2,VALOR_MDEC+5      ;Carga el digito de unidades del numero verificador
                swap    temp                    ;Invierte el orden de los bytes de decimas del numero verificador
                or      temp,temp2              ;Junta las decimas y las unidades en un solo registro en BCD
                rcall   convhex                 ;Convierte el digito verificador a hexadecimal y lo guarda en temp
                sts     VERIFD,temp             ;Guarda el DIGITO verificador en VERIFD

                clr     ZH
                ldi     ZL,NVMESES
                lds     temp,DECMES
                add     ZL,temp                 ;temp apunta al contador del mes correspondiente
                lds     temp2,VERIFD
                sub     temp2,temp              ;Resta
                cpi     temp2,1
                breq    k_verifd
                clr     temp                    ;Pone temp a cero para indicar que el digito verificador es invalido
                ret

k_verifd:       subi    temp,-1                 ;Suma un uno al contador correspondiente del mes
                st      Z,temp                  ;Lo almacena nuevamente en RAM
                ldi     temp,1                  ;Carga un uno en temp para indicar que el digito verificador es uno
                ret
;*****************************************************************************
;*      CVRCREDIT
;*      Convierte a binario el numero desencriptado que corresponde al credito a abonar
;*      Despues el numero en binario lo multiplica por 1000 ya que la tarifa se maneja como un numero entero
;*      Calcula el numero de KW-H a abonar como unidades de energia y como unidades monetarias
;*      Abona unidades de energia pendientes
;*      Debido a que se abonan cantidades enteras casi siempre existe un residuo menor a la tarifa el cual se
;*      almacena como pendiente.
;*****************************************************************************
CVRCREDIT:

.def    res41   =r0     ;res41 a res43 es donde se guardara el numero de 4 digitos en binario
.def    res42   =r1
.def    res43   =r2
.def    mp41    =r3     ;mp41 a mp43 se emplean para guardar el resultado parcial de la multiplicacion por 10, 10E1,10E2...etc
.def    mp42    =r4
.def    mp43    =r5

.def    mc4     =r16    ;Se emplea como contador y como multiplicando cada que se multiplique por 10

.def    cicles  =r21    ;Contadores auxiliares
.def    cont1   =r22
.def    cont2   =r23

                ldi     ZH,HIGH(VALOR_MDEC+20)          ;Apuntador al credito a abonar previamente desencriptado
                ldi     ZL,LOW(VALOR_MDEC+20)
```

```
                clr     res42                   ;Limpiar los registros antes comenzar operaciones
                clr     res43

                ldi     temp,1                  ;Registro auxiliar para multiplicar por 10, 10E1, 10E2 etc.
                ldi     cicles,3                ;Carga cicles con el numero de digitos a procesar (4)
                ldi     cont1,0
                ld      res41,-Z                ;Carga el primer digito en res1
Sdig4:          ld      mp41,-Z                 ;Carga siguiente digito en mp1
                clr     mp42                    ;Limpia los registros antes de procesar otro digito
                clr     mp43
;*****************************************************************************
MX4:            ldi     cont2,16                ;Numero de corrimientos para multiplicar 16X8 bits
                ldi     mc4,10                  ;El multiplicando siempre es 10
                lsr     mp43
                ror     mp42
                ror     mp41

M16X8:          brcc    noad4
                add     mp43,mc4

noad4:          ror     mp43
                ror     mp42
                ror     mp41
                dec     cont2
                brne    M16X8
;*****************************************************************************
                inc     cont1                   ;Incrementa a uno el contador
                cp      cont1,temp              ;Compara para saber si ya multiplico por 10 las veces necesarias
                brne    MX4

                clr     cont1                   ;Pone a cero cont1
                add     res41,mp41              ;Suma los resultados
                adc     res42,mp42
                adc     res43,mp43
                inc     temp                    ;Aumenta el numero de veces a multiplicar por 10
                dec     cicles                  ;Verifica si ya se procesaron los 20 digitos
                mov     mp161,res42
                mov     mp160,res41
                ldi     mc161,HIGH(1000)        ;Carga un 1000 en el multiplicando
                ldi     mc160,LOW(1000)

                rcall   MULTIP1616              ;Multiplica la energia abonada por mil

                CBIs_HR GIMSK,0b10000000        ;Deshabilita la interrupcion externa 0 momentaneamente para actualizar los registros de credito abonado

                lds     count,ABON1             ;Carga desde RAM la energia restante (si es que hay)
                lds     temp2,ABON2
                lds     temp,ABON3
```

```
add     mp160,temp          ;Suma la energia restante con la que se va a abonar
adc     mp161,temp2
adc     mp162,count

lds     temp2,ENERRESH      ;Carga en temp2 el byte alto de credito que no fue abonado la vez anterior
lds     temp,ENERRESL       ;Carga en temp el byte bajo de credito que no fue abonado la vez anterior
clr     count               ;Limpia count que se usa solo para empatar el registro faltante en la suma en caso de que se genere un acarreo
add     mp160,temp          ;Realiza la suma desde el byte menos significativo
adc     mp161,temp2
adc     mp162,count         ;Hasta el mas significativo

sts     ABON1,mp162         ;Almacena el credito a abonar en RAM
sts     ABON2,mp161         ;Almacena el credito a abonar en RAM
sts     ABON3,mp160         ;Almacena el credito a abonar en RAM

lds     dv24uH,TARIFH       ;Carga la parte alta de la tarifa en el divisor
lds     dv24uL,TARIFL       ;Carga la parte baja de la tarifa en el divisor
rcall   DIV24U

sts     ENERGYH,dd24uH      ;Guarda en los registros correspondientes la energia a abonar en KW-H
sts     ENERGYL,dd24uL      ;Estos valores son enteros
lds     count,ABON1         ;Carga el credito a abonar para restar el residuo de la division anterior
lds     temp2,ABON2         ;De tal manera que el credito abonado siempre es multiplo de la tarifa
lds     temp,ABON3          ;
clr     r23                 ;Limpia r23 que se usa solo para empatar el registro faltante en la resta en caso de que se genere un acarreo
sub     temp,rem24L         ;Resta al credito a abonar el residuo de la division
sbc     temp2,rem24H        ;
sbc     count,r23           ;

sts     ABON1,count
sts     ABON2,temp2
sts     ABON3,temp


SBIs_HR GIMSK,0b10000000    ;Habilita nuevamente la interrupcion externa 0

sts     ENERRESH,rem24H     ;Guarda la energia no abonada para tomarla en cuenta la proxima vez que se abone
sts     ENERRESL,rem24L     ;Credito al sistema
ret
```

```
;***************************************************************************
;*          DIV24U
;*          Divide el credito introducido entre la tarifa para determinar el numero de KW-Horas a abonar
;*  El residuo contine la parte fraccionaria que no sera abonada hasta la siguiente vez que se introduzca credito
;***************************************************************************
DIV24U:

.def        rem24L    =r4             ;Byte bajo del residuo
.def        rem24H    =r5             ;Byte alto del residuo (Para ajustar la proxima vez)
.def        dd24uL    =r19            ;Byte bajo del dividendo
.def        dd24uH    =r18            ;Byte medio del dividendo
.def        dd24uHH =r0               ;Byte alto del dividendo
.def        dv24uL    =r2             ;Byte bajo del divisor (tarifa)
.def        dv24uH    =r3             ;Byte alto del divisor (tarifa)

                    clr     rem24L                  ;clear remainder Low byte
                    sub     rem24H,rem24H           ;clear remainder High byte and carry
                    ldi     count,25                ;init loop counter
d24u_1:             rol     dd24uL                  ;shift left dividend
                    rol     dd24uH
                    rol     dd24uHH
                    dec     count                   ;decrement counter
                    brne    d24u_2                  ;if done
                    ret                             ;return
d24u_2:             rol     rem24L                  ;shift dividend into remainder
                    rol     rem24H
                    sub     rem24L,dv24uL           ;remainder = remainder - divisor
                    brcc    d24u_3                  ;if result negative
                    add     rem24L,dv24uL           ;restore remainder
                    adc     rem24H,dv24uH
                    clc                             ;clear carry to be shifted into result
                    rjmp    d24u_1                  ;else
d24u_3:             sec                             ;set carry to be shifted into result
                    rjmp    d24u_1
;***************************************************************************
;*          FIXCREDIT
;*          Rutinas necesarias para poder desplegar en pantalla el credito disponible
;***************************************************************************
FIXCREDIT:

.def        resid=r11
.def        dv1       =r0       ;res
.def        dv2       =r1       ;res
.def        dv3       =r2
.def        divs1     =r21
.def        conta     =r16
```

```
                lds     dv1,ABON3           ;Carga el credito que esta en hexadecimal
                lds     dv2,ABON2
                lds     dv3,ABON1
                ldi     divs1,0x0A          ;Divisor= 10
                ldi     conta,8             ;Numero de digitos a procesar
                ldi     ZH,HIGH(CREDTDEC+8)
                ldi     ZL,LOW(CREDTDEC+8)
s_div:          rcall   divcre
                mov     temp,resid
                subi    temp,-0x30
                st      -Z,temp
                dec     conta
                brne    s_div
                lds     temp,CREDTDEC+7
                sts     CREDTDEC+8,temp
                lds     temp,CREDTDEC+6
                sts     CREDTDEC+7,temp
                lds     temp,CREDTDEC+5
                sts     CREDTDEC+6,temp
                ldi     temp,'.'
                sts     CREDTDEC+5,temp
                clr     temp
                sts     CREDTDEC+9,temp     ;Cero al final de la cadena

                ldi     ZH,HIGH(CREDTDEC)
                ldi     ZL,LOW(CREDTDEC)    ;Z apunta a la direccion donde se esncuentra el credito en ascii
                ldi     count,4             ;Se van a procesar los primeros cuatro digitos para que en caso de
                                            ;ser ceros se cambien por espacios en blanco
csigs:          ld      temp,Z              ;Carga un digito en ASCII desde CREDTEC
                cpi     temp,0x30           ;Lo compara con cero (ASCII)
                brne    nospc               ;Si no es cero termina
                ldi     temp,' '            ;Carga un especio en blanco en temp
                st      Z+,temp             ;Lo guarda en la direccion especificada por Z
                dec     count               ;Checa el siguiente digito
                brne    csigs               ;Sigue hasta que procese los 4 digitos o uno sea diferente de cero

nospc:          ret                         ;Fin de la rutina


divcre:         clr     resid               ;clear remainder Low byte
                clc                         ;Limpia carry

                ldi     count,25            ;init loop counter
c27_1:          rol     dv1                 ;shift left dividend
                rol     dv2
                rol     dv3

                dec     count               ;decrement counter
```

```
                brne    c27_2           ;if done
                ret                     ;return
c27_2:          rol     resid           ;shift dividend into remainder
                sub     resid,divs1     ;remainder = remainder - divisor
                brcc    c27_3           ;if result negative
                add     resid,divs1     ;restore remainder
                clc                     ;clear carry to be shifted into result
                rjmp    c27_1           ;else
c27_3:          sec                     ;set carry to be shifted into result
                rjmp    c27_1


mensaje1:
.db  "    VOLTAJE"
.db 0
mensaje2:
.db  "    CORRIENTE"
.db 0
mensaje3:
.db  "  POTENCIA APA."
.db 0
mensaje4:
.db  "  POTENCIA REAL"
.db 0
mensaje5:
.db  "  FACTOR DE POT."
.db 0
menscode:
.db  "CLAVE:"
.db 0
credit:
.db  "CRED.$"
.db 0
coderr1:
.db  "     CODIGO"
.db 0
coderr2:
.db  "   INCORRECTO"
.db 0
deserr1:
.db             "   INCORRECTO"
.db 0

TECDESACT:
.db             "TECLADO INACTIVO"
.db 0

dias:
```

```
.db "LU-"
.db "MA-"
.db "MI-"
.db "JU-"
.db "VI-"
.db "SA-"
.db "DO-"
meses:
.db  "ENE"
.db  "FEB"
.db  "MAR"
.db  "ABR"
.db  "MAY"
.db  "JUN"
.db  "JUL"
.db  "AGO"
.db  "SEP"
.db  "OCT"
.db  "NOV"
.db  "DIC"
```

ANEXO C

PROGRAMA PARA LA ENCRIPTACIÓN DE CLAVES

A continuación se muestra el listado del programa en C que realiza la encriptación de las claves de 20 dígitos.

```
#include <NTL/ZZ.h>


void main(void)
{
	ZZ p,q,n,l,d,e,div;		//Enteros del tipo Z
	ZZ remi,A[30],B[30],a,b;
	long err=450;
	long y=34;
	int i=1,cont;

	GenPrime(p, y,err );		//Calcula P
	cout <<"\nP= "<< p <<"\n";	//Imprime P
	y=34;

	GenPrime(q, y,err );		//Calcula Q

	cout <<"Q= "<< q <<"\n";	//Imprime Q
	n=p*q;
	l=(p-1)*(q-1);			//Calcula N

	cout <<"N= "<< n <<"\nL= " << l <<"\n";	//Imprime la llave N

	e=65537;			//Valor seleccionado para e
	remi=1;

	while(remi!=0){			//Calculo de la llave D
		div=l/e;
		A[i]=div;
		remi=l%e;
		B[i]=remi;
		cout <<"\n"<<l<<"="<<e<<"*"<< div <<"+"<< remi<<"\n";
		l=e;
		e=remi;
		i++;
	}

	cont=i-2;

	for(i=1;i<cont+2;i++) {
		cout <<"A"<<i<<"="<<A[i]<<"\tB"<<i<<"="<<B[i]<<"\n";
	}

	a=1;
	b=A[cont];
	for(i=cont;i>=2;i--){
```

```
        if (i%2==1)
                b=a*A[i-1]+b;
        else
                a=a+b*A[i-1];
        }

        cout <<"\nEl valor de d= "<<a<<"\n";          //Imprime la llave D


        ZZ result,m,desencrypt,c,o; //encriptacion
        result=1;
        m=to_ZZ("07020198765432109999");        //Dato a ser encriptado

        cout<<m<<"\n";          //Imprime el dato a ser encriptado

        e=65537;

        while(e>0){             //Encripta el dato
                if((e%2)==1){
                        result=(result*m)%n;
                        e=e-1;
                        }
                m=(m*m)%n;
                e=e/2;
        }

        cout   <<result<<"\n"; //Imprime dato encriptado

        desencrypt = 1;         //Desencripta resultado de la encriptación para verificar
        c=result;               //Dato encriptado
        d=a;

        while(d>0){             //Comienza la desencriptación
                if((d%2)==1){
                        desencrypt=(desencrypt*c)%n;
                        d=d-1;
                }
                c=(c*c)%n;
                d=d/2;
        }

cout <<desencrypt<<"\n";     //Imprime dato desencriptado
```

BIBLIOGRAFÍA

[CF: 2002] Comisión Federal de Electricidad, 2002, hoja web con información sobre el sector eléctrico mexicano. http://www.cfe.gob.mx.

[LF: 2002] Luz y Fuerza del centro, 2002, hoja web con información sobre el sector eléctrico, servicios y tarifas en la zona centro del país. Http://www.lfc.gob.mx.

[MT: 2002] Metering Technology Corporation, 2002, hoja web con información sobre medidores de prepago que operan vía telefónica y mediante radiofrecuencia. http://www.metertech.com.

[EM: 2002] Energy Measurements (Pty) Ltd, 2002, hoja web con información sobre medidores de prepago. http://www.cashpower.com.

[HP: 2002] Homeplug, 2002, hoja web con información del estándar PLC (Power Line Carrier). http://www.homeplug.org.

[RA: 2002] Ramar Technologies, 2002, hoja web con información de medidores que operan mediante radiofrecuencia. http://www.ramartech.com.

[AT: 2002] Atmel, hoja de especificaciones del AT90S8515, Atmel, 2002. http://www.atmel.com.

[CL: 2002] Cirrus Logic, hoja de especificaciones del CS5460A, Cirrus Logic, 2002. http://www.crystal.com.

[FS: 2001] Fairchild Semiconductors, hoja de especificaciones del MM74C922, Fairchild Semiconductors, 2001.

[TA: 2002] Tianma, hoja de especificaciones de la pantalla de cristal líquido TM16AAC, Tianma, 2002.

[BS: 1996] Schneier, B., Applied Cryptography, EUA, John Wiley & Sons, Inc, 1996.

[RP: 2002] Other Algorithms, 2002, hoja web con información acerca del “Russian Peasant algorithm”. http://online.edfac.unimelb.edu.au/485129/wnproj/multiply/lattice.htm.

[NT: 2002] NTL a library for doing number theory, 2002, hoja web con información y librerías para emplear en teoría de números. http://www.shoup.net/ntl/index.html.

[DE: 1983] Denning, Dorothy E., Cryptography and Data Security, EUA, Addison Wesley, 1983.

[DM: 2002] Direct Metering, 2002, hoja web con información de medidores eléctricos de prepago. http://www.direct-metering.com.

[IE: 2002] International Electrotechnical Commission, 2002, hoja web con información de los estándares internacionales que rigen el diseño y operación de los medidores eléctricos. http://www.iec.ch.

[CA: 2003] Cadence PCB, hoja web con información acerca del paquete de diseño Orcad http://www.cadendepcb.com.

[EA: 2002] The Euclidean algorithm, hoja web con información sobre el algoritmo euclidiano. http://www.cs.vu.nl/~jketema/gofer/doc/exteuc.pdf.