REGULADOR DE PANEL SOLAR - BATERÍA - CARGA CON MICROCONTROLADOR PIC

Christian Paúl Henríquez Prevoo

Piura, 21 de Marzo de 2003

FACULTAD DE INGENIERÍA

Área Departamental de Ingeniería Mecánico-Eléctrica

Marzo 2003

UNIVERSIDAD DE PIURA

FACULTAD DE INGENIERIA

rograma Academico de Ingenieria Mecanico - Electrica

"Regulador de panel solar - bateria - carga con microcontrolador PIC"

Tesis para optar el Titulo' de:

Ingeniero Mecanico - Electrico

Christian Paul Henriquez Prevoo

Piura Marzo, 2003

A mi familia

PRÓLOGO

Actualmente las tendencias mundiales en cuanto a la investigación para el desarrollo de sistemas convencionales de electrificación fotovoltaico están aumentando por ser la energía solar; la menos contaminante y por ende menos dañina para el medio. Para el desarrollo de la presente tesis se ha implementado un nuevo sistema de regulación de carga y adquisición datos que contribuirán con el desarrollo de los sistemas de electrificación fotovoltaico.

Los objetivos de la presente tesis son:

- Implementación de un Regulador de Carga de un sistema convencional utilizando el microcontrolador PIC 16C711.

- Implementación de una Tarjeta de Adquisición de Datos de uso generalizado aplicada a sistemas convencionales fotovoltaicos.

El módulo físico consiste en un Regulador conectado a una Tarjeta de Adquisición de Datos, a un Panel Solar y a una Batería que fue desarrollado como parte de los frutos de la presente tesis. Este módulo implementado permite verificar si se cumple el tipo de regulación propuesta, transmisión de datos así como la recepción de los mismos mediante una PC. También sirve para experimentar nuevas formas de regulación para estudios posteriores.

El módulo consta de dos tarjetas, una es la Tarjeta Reguladora y otra es la Tarjeta de Adquisición de Datos. La tarjeta Reguladora posee dos entradas, un puerto bidireccional y dos salidas. Una entrada es para la conexión del Panel, otra es para la batería, el puerto bidireccional es para la comunicación con la tarjeta de adquisición de datos, una salida es para la carga (Usuario) y otra es para la alimentación de la tarjeta de adquisición de datos.

La presente tesis apertura una nueva línea de desarrollo en cuanto a la búsqueda de nuevas formas de regulación de carga y de mejor aprovechamiento de la electricidad solar en el departamento de Electrónica y Automática de la Universidad de Piura.

El sistema desarrollado en la presente tesis busca la optimización de un sistema de regulación y adquisición de datos cuya finalidad es el óptimo rendimiento de un sistema convencional de electrificación fotovoltaico autónomo que cada vez más se están implementando en la actualidad.

La Regulación que se ha implementado en la presente tesis consta de un sistema inteligente que toma decisiones a partir de entradas que reportan el estado del sistema. Por otro lado en paralelo a la ejecución de la regulación, se esta enviando y recibiendo datos mediante un *bus* bidireccional que comunica a la Tarjeta Reguladora con la Tarjeta de Adquisición de Datos.

Deseo expresar mi más profundo agradecimiento a mi asesor, Dr. Ing. Justo Oquelis Cabredo, al Ing. Edward Cornelis Mulder y a los miembros del Departamento de Electrónica y Automática de la Universidad de Piura, por el apoyo incondicional y motivación constante que me brindaron haciendo posible de esa manera la realización de la presente tesis.

RESUMEN

Actualmente existen reguladores de carga, que no son flexibles en cuanto a la modificación de sus parámetros de regulación. Con estos reguladores no es posible verificar el estado de carga ni el de descarga de la batería, siendo estos estados influyentes en la vida útil de la misma.

Los objetivos de la presente tesis son implementar un regulador que sea flexible en cuanto a la modificación de sus parámetros de regulación e implementar una tarjeta de adquisición de datos que se pueda usar en cualquier proceso y que acoplada al regulador podamos conocer el estado de carga y de descarga de la batería a través del tiempo.

Para el desarrollo de lo antes expuesto se han aplicado recientes tecnologías en *hardware* y *software* como lo es el microcontrolador PIC 16C711 y la programación orientada a objetos.

La presente tesis cumple con los objetivos planteados. Además proyecta nuevas ideas que harán que el sistema de regulación sea también un sistema de medición y monitoreo de las variables que influyen en el óptimo rendimiento de un sistema de electrificación fotovoltaico.

INDICE

INTRODUCCIÓN

El presente trabajo se ha desarrollado con el fin de:

a) Implementar un regulador de carga para sistemas de electrificación fotovoltaica usando el microcontrolador PIC 16C711.
b) Implementar una tarjeta de adquisición de datos de uso generalizado aplicada al sistema de electrificación fotovoltaica donde actúe el regulador.
c) Implementar plataformas de software basadas en lenguaje ensamblador y Visual C++, orientado a resolver los problemas presentes en un sistema fotovoltaico autónomo.

El presente trabajo se ha dividido en cinco capítulos, a los que se han anexado las conclusiones y dos apéndices.

En el capítulo I, se presenta una descripción básica de los elementos de un sistema convencional de electrificación fotovoltaica. Se presenta de manera bastante resumida los aspectos más importantes del sistema como su configuración interna y se plantean los problemas que el diseño implementado debe resolver.

En el capítulo II, se presenta una descripción detallada del microncontrolador PIC 16C711, su potencialidad y las funcionalidades que nos brinda.

En el capítulo III, se explica el hardware en detalle del diseño implementado en tanto a flexibilidad, tipo de regulación, seguridad, bus de comunicación, adquisición de datos, comunicación con la PC.

En el capítulo IV, se trata del estudio del software del regulador , la tarjeta de adquisición de datos y de la PC, se muestra el desarrollo del programa en diagramas de bloques y el orden de ejecución de tareas de las tarjetas implementadas.

En el capítulo V, se presenta el desarrollo del software y hardware, los resultados de la regulación ante un sistema de carga simulado por una fuente de tensión, la aplicación desarrollada en Visual C++ y la evaluación del los datos obtenidos por la tarjeta de adquisición de datos. Se especifican sus funciones más importantes y los detalles considerados para su desarrollo.

Seguidamente se presentan las diversas conclusiones y recomendaciones a las que se llegó después del desarrollo del presente trabajo.

En los apéndices A y B se presentan los programas principales de las aplicaciones en lenguaje ensamblador y archivos fuente en Visual C++.

CAPÍTULO-1

SISTEMA CONVENCIONAL DE CARGA CON PANEL SOLAR

El presente capítulo tiene como objetivo:

Una visión general de la energía solar, como es que llega a nosotros y como podríamos aprovecharla.

Una descripción del sistema de obtención de la energía eléctrica a partir de la energía solar. Se tratarán a fondo los elementos que intervienen en un sistema de carga panel solar – batería.

Se plantearán los problemas que el regulador diseñado debe solucionar.

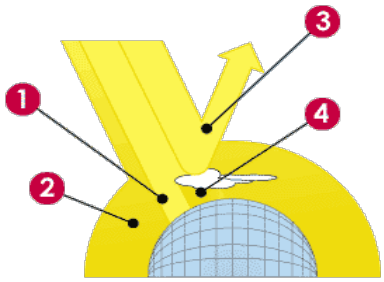
1.1 Manifestación de la energía solar

1.1.1 Radiación

El sol emite constantemente enormes cantidades de energía; una fracción de ésta alcanza la tierra. La cantidad de energía solar que recibimos en un solo día resulta más que suficiente para cubrir la demanda mundial de todo un año. Sin embargo, no toda la energía proveniente del sol puede ser utilizada de manera efectiva. Parte de la luz solar es absorbida en la atmósfera terrestre o, reflejada nuevamente al espacio.
La intensidad de la luz solar que alcanza nuestro planeta varía según el momento del día y del año, el lugar y las condiciones climáticas [8]. La energía total registrada sobre una base diaria o anual se denomina 'radiación' e indica la intensidad de dicha luz. La radiación se expresa en Wh/m^2 por día o, también, en kWh/m^2 por día.

1.1.2 Orientación

La luz solar viaja en línea recta desde el sol hasta la tierra. Al penetrar la atmósfera terrestre, una parte se dispersa y otra cae sobre la superficie en línea recta. Finalmente, una última parte es absorbida por la atmósfera. La luz solar dispersa se denomina radiación difusa o luz difusa. La luz del sol que cae sobre la superficie sin dispersarse ni ser absorbida, es, por supuesto, radiación directa. Como todos habrán constatado, gracias a los baños de sol y al trabajo al aire libre, la radiación solar directa es más intensa.



1. Directa
2. Absorción
3. Reflexión
4. Indirecta

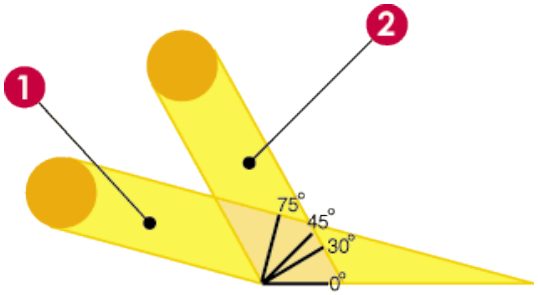
Fig. 1.1: *Radiación y Orientación*

Un panel solar genera electricidad incluso en ausencia de luz solar directa. Por ende, un sistema solar generará energía aún con cielo nublado. Sin embargo, las condiciones óptimas de operación implican: la presencia de luz solar plena y un panel orientado lo mejor posible hacia el sol, con el fin de aprovechar al máximo la luz solar directa. En el Hemisferio Norte, el panel deberá orientarse hacia el sur y en el Hemisferio Sur, hacia el norte.
Por lo tanto, en la práctica, los paneles solares deberán ser colocados en ángulo con el plano horizontal (inclinados). Cerca del ecuador, el panel solar deberá colocarse ligeramente inclinado (casi horizontal) para permitir que la lluvia limpie el polvo.
Una pequeña desviación en la orientación no influye significativamente en la generación de electricidad, ya que durante el día el sol se traslada en el cielo de este a oeste.

1.1.3 Ángulo de inclinación

El sol se desplaza en el cielo de este a oeste. Los paneles solares alcanzan su máxima efectividad cuando están orientados hacia el sol, en un ángulo perpendicular con éste a

mediodía. Por lo general, los paneles solares son colocados sobre un techo o una estructura y tienen una posición fija; no pueden seguir la trayectoria del sol en el cielo. Por lo tanto, no estarán orientados hacia el astro con un ángulo óptimo (90 grados) durante toda la jornada. El ángulo entre el plano horizontal y el panel solar se denomina ángulo de inclinación. Debido al movimiento terrestre alrededor del sol, existen también variaciones estacionales. En invierno, el sol no alcanzará el mismo ángulo que en verano. Idealmente en verano, los paneles solares deberían ser colocados en posición ligeramente más horizontal para aprovechar al máximo la luz solar. Sin embargo, los mismos paneles no estarán entonces, en posición óptima para el sol del invierno. Con el propósito de alcanzar un mejor rendimiento anual promedio, los paneles solares deberán ser instalados en un ángulo fijo, determinado en algún punto entre los ángulos óptimos para el verano y para el invierno. Cada latitud presenta un ángulo de inclinación óptimo [8]. Los paneles deben colocarse en posición horizontal únicamente en zonas cercanas al Ecuador.



1. sol de invierno

2. sol de verano

Fig. 1.2: *Angulo de Inclinación*

1.1.4 – Energía solar incidente.

El Sol produce una enorme cantidad de energía: aproximadamente 1,1 x 10^{20} kW hora cada segundo (1 kW hora es la energía necesaria para iluminar una bombilla de 100 *watts* durante 10 horas). La atmósfera exterior intercepta aproximadamente la mitad de una billonésima parte de la energía generada por el sol, o aproximadamente 1.5 trillones (1.500.000.000.000.000.000) de kW hora al año. Sin embargo, debido a la reflexión, dispersión y absorción producida por los gases de la atmósfera, sólo un 47% de esta energía, o aproximadamente 0.7 trillones (700.000.000.000.000.000) de kW hora alcanzan la superficie de la tierra.

La cantidad de energía que se consume en el mundo anualmente es aproximadamente 85 billones (85.000.000.000.000) de kW hora [9]. Esto es lo que se puede medir, es decir la energía que se compra, vende o comercializa. No hay forma de saber exactamente qué cantidad de energía no comercial consume cada persona (por ejemplo cuanta madera se quema, o que cantidad de agua se utiliza en pequeños saltos de agua para producir energía eléctrica). Según algunos expertos esta energía no comercial puede constituir como mucho una quinta parte del total de energía consumida. Aunque fuera este el caso, la energía total consumida por el mundo significaría sólo 1/7000 de la energía solar que incide sobre la superficie de la tierra cada año.

Con el fin de simplificar los cálculos realizados en base a la información sobre radiación, la energía solar se expresa en equivalentes a horas de luz solar plena. La luz solar plena registra una potencia de unos 1,000 W/m²; por lo tanto, una hora de luz solar plena equivale a 1 kWh/m² de energía.

Ésta es, aproximadamente, la cantidad de energía solar registrada durante un día soleado de verano, con cielo despejado, en una superficie de un metro cuadrado, colocada en perpendicular al sol.

La radiación varía según el momento del día. Sin embargo, también puede variar considerablemente de un lugar a otro, especialmente en regiones montañosas. La radiación fluctúa entre un promedio de 1,000 kWh/m² al año, en países y 2,000 a 2,500 kWh/m² al año, en las zonas desérticas. Estas variaciones se deben a las condiciones climáticas y a la diferencia con respecto a la posición relativa del sol en el cielo (elevación solar), la cual depende de la latitud de cada lugar.

1.2 Electricidad Solar.

1.2.1 Producción de la energía eléctrica a partir del Sol

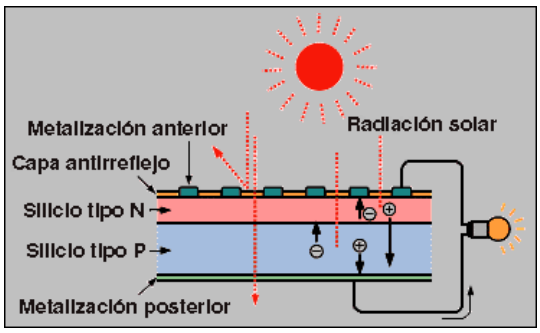


Fig. 1.3: *Efecto fotovoltaico en una célula solar*

La producción está basada en el fenómeno físico denominado "efecto fotovoltaico", que básicamente consiste en convertir la luz solar en energía eléctrica por medio de unos dispositivos semiconductores denominados células fotovoltaicas.
Las células están elaboradas a base de silicio puro (uno de los elementos más abundantes, componente principal de la arena) con adición de impurezas de ciertos elementos químicos (boro y fósforo), y son capaces de generar cada una corriente de 2 a 4 Amperios, a un voltaje de 0,46 a 0,48 Voltios, utilizando como fuente la radiación luminosa.
Las células se montan en serie sobre paneles o módulos solares para conseguir un voltaje adecuado. Parte de la radiación incidente se pierde por reflexión (rebota) y otra parte por transmisión (atraviesa la célula). El resto es capaz de hacer saltar electrones de una capa a la otra, creando una corriente proporcional a la radiación incidente. Una capa antirreflejo, instalada en la superficie de la célula, aumenta la eficacia de la misma.
La energía proveniente del sol puede servir para distintos propósitos. Uno de ellos es la generación de electricidad, conocida como 'electricidad solar'. Con la ayuda de paneles solares, la luz del sol es convertida directamente en energía eléctrica. Este proceso es denominado efecto fotovoltaico o FV.
El uso de electricidad solar presenta muchas ventajas, puesto que se trata de una fuente de energía limpia, silenciosa y confiable [5]. En un principio, fue utilizada en satélites; en 1958, se lanzó el *Vanguard I*, primer satélite provisto de celdas solares para la generación de energía. Hoy en día, el uso de la electricidad solar se ha generalizado. En zonas remotas donde no hay

conexión a la red de distribución pública, esta forma de energía solar es empleada para satisfacer la demanda de electricidad de los hogares y para alimentar bombas de agua y refrigeradores para vacunas; con frecuencia, estos sistemas cuentan con baterías para almacenar electricidad.

1.2.1 Impacto ambiental.

La energía solar fotovoltaica, al igual que otras energías renovables, constituye, frente a los combustibles fósiles, una fuente inagotable que contribuye al autoabastecimiento energético nacional y es menos perjudicial para el medio ambiente [5], evitando los efectos de su uso directo (contaminación atmosférica, residuos, etc) y los derivados de su generación (excavaciones, minas, canteras, etc.).

Los efectos de la energía solar fotovoltaica sobre los principales factores ambientales son los siguientes:

Clima: la generación de energía eléctrica directamente a partir de la luz solar no requiere ningún tipo de combustión, por lo que no se produce polución térmica ni emisiones de CO2 que favorezcan el efecto invernadero.

Geología: Las células fotovoltaicas se fabrican con silicio, elemento obtenido de la arena, muy abundante en la Naturaleza y del que no se requieren cantidades significativas. Por lo tanto, en la fabricación de los paneles fotovoltaicos no se producen alteraciones en las características litológicas, topográficas o estructurales del terreno.

Suelo: al no producirse ni contaminantes, ni vertidos, ni movimientos de tierra, la incidencia sobre las características físico-químicas del suelo o su erosionabilidad es nula.

Aguas superficiales y subterráneas: No se produce alteración de los acuíferos o de las aguas superficiales ni por consumo, ni por contaminación por residuos o vertidos.

Flora y fauna: la repercusión sobre la vegetación es nula y, al eliminarse los tendidos eléctricos, se evitan los posibles efectos perjudiciales para las aves.

Paisaje: los paneles solares tienen distintas posibilidades de integración, lo que hace que sean un elemento fácil de integrar y armonizar en diferentes tipos de estructuras, minimizando su impacto visual. Además, al tratarse de sistemas autónomos, no se altera el paisaje con postes y líneas eléctricas.

Ruidos: el sistema fotovoltaico es absolutamente silencioso, lo que representa una clara ventaja frente a los generadores de motor en viviendas aisladas.

Medio social: El suelo necesario para instalar un sistema fotovoltaico de dimensión media, no representa una cantidad significativa como para producir un grave impacto. Además, en gran parte de los casos, se pueden integrar en los tejados de las viviendas.

Por otra parte, la energía solar fotovoltaica representa la mejor solución para aquellos lugares a los que se quiere dotar de energía eléctrica preservando las condiciones del entorno; como es el caso por ejemplo de los Espacios Naturales Protegidos.

1.2.2 Rentabilidad.

La rentabilidad depende del lugar del mundo donde nos encontremos. Una gran parte de la humanidad, en los países en desarrollo, no tiene acceso a la electricidad por carecer de una infraestructura eléctrica básica. En estos países la energía solar fotovoltaica resulta ser la fuente más rentable para obtener electricidad, y en algunos lugares, la única.
En los países desarrollados, en los que existe una amplia infraestructura eléctrica, la cuestión es diferente. En este caso, en términos puramente económicos, los sistemas fotovoltaicos sólo resultan rentables en lugares alejados de la red convencional. No obstante, la cuestión cambiaría bastante si, además de la rentabilidad económica, tuviéramos en cuenta también el costo ambiental de cada fuente de energía.

1.2.3 Aplicaciones

Prácticamente cualquier aplicación que necesite electricidad para funcionar se puede alimentar con un sistema fotovoltaico adecuadamente dimensionado. La única limitación es el coste del equipo y, en algunas ocasiones, el tamaño del campo de paneles. No obstante, en lugares remotos alejados de la red de distribución eléctrica, lo más rentable suele ser instalar energía solar fotovoltaica antes que realizar el enganche a la red.
Entre las principales aplicaciones se incluyen: electrificación de viviendas, sistemas de bombeo y riego, iluminación de carreteras, repetidores de radio y televisión, depuradoras de aguas residuales, generación de energía para casas, oficinas, etc. o el suministro de energía a la red de distribución utilizando sistemas generadores de electricidad solar [15].

1.3 Sistemas fotovoltaicos

1.3.1 Tipos de Sistemas

Para utilizar paneles solares como fuente de energía segura y confiable, es necesario contar con los siguientes componentes adicionales: cables, una estructura de soporte y, dependiendo del tipo de sistema (conectado a la red, autónomo o de emergencia), un inversor o un controlador de carga + baterías. El sistema completo se denomina sistema de generación de electricidad solar.

Existen tres tipos de sistemas:

a) Sistemas solares autónomos o fotovoltaicos domiciliarios

b) Sistemas solares conectados a la red

c) Sistemas solares de emergencia

a) Sistemas solares autónomos o sistemas fotovoltaicos domiciliarios

Los sistemas solares autónomos o fotovoltaicos domiciliarios (SFD) son instalados en los casos en que no se tiene acceso a la red de distribución pública. Ellos requieren de una batería, con el fin de asegurar el suministro de electricidad durante la noche o periodos de escasez de luz solar. Con frecuencia, los sistemas solares domiciliarios son utilizados para satisfacer las necesidades de electricidad de un hogar. Los sistemas pequeños (disponibles a nivel comercial como kitSFD) cubren las necesidades más básicas (iluminación y, en algunos casos, televisión o radio); los sistemas más grandes pueden alimentar, además, una bomba de agua, un teléfono inalámbrico, un refrigerador, herramientas eléctricas (un taladro, una máquina de coser, etc.) y un equipo. El sistema está compuesto por un panel solar, un controlador, una batería de almacenamiento, cables, la carga eléctrica y una estructura de soporte.

El sistema consta de los siguientes elementos (ver esquema):

- Un generador solar, compuesto por un conjunto de paneles fotovoltaicos, que captan la radiación luminosa procedente del sol y la transforman en corriente continua a baja tensión (12 ó 24 V).

- Un acumulador, que almacena la energía producida por el generador y permite disponer de corriente eléctrica fuera de las horas de luz o días nublados.

- Un regulador de carga, cuya misión es evitar sobrecargas o descargas excesivas al acumulador, que le produciría daños irreversibles; y asegurar que el sistema trabaje siempre en el punto de máxima eficiencia.

- Un inversor (opcional), que transforma la corriente continua de 12 ó 24 V almacenada en el acumulador, en corriente alterna de 230 V.

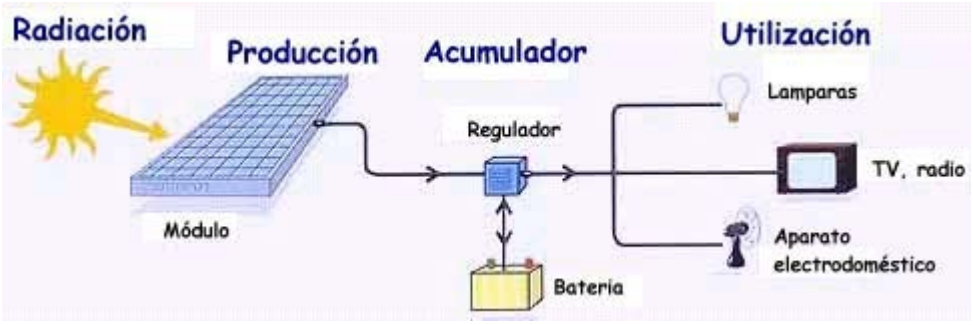


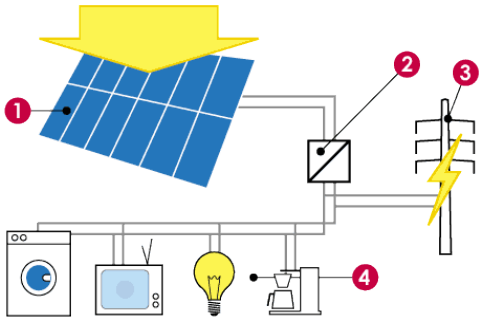
Fig. 1.4: *Una instalación solar fotovoltaica sin inversor, utilización a 12Vcc*

Fig. 1.5: *Una instalación solar fotovoltaica con inversor, utilización a 230Vca*

Una vez almacenada la energía eléctrica en el acumulador hay dos opciones: sacar una línea directamente de éste para la instalación y utilizar lámparas y elementos de consumo de 12 ó 24 Vcc (primer esquema) o bien transformar la corriente continua en alterna de 230 V a través de un inversor (segundo esquema).

b) Sistemas solares conectados a la red

En aquellos casos en los que, aun habiendo conexión a una red de distribución pública, el usuario desea contar con electricidad generada por una fuente limpia (solar), se puede conectar los paneles solares a la red. Si se instala suficientes paneles, los artefactos eléctricos en el hogar /edificio operarán, entonces, con electricidad solar. Un sistema solar conectado a la red consta, básicamente, de uno o más paneles solares, un inversor, cables, la carga eléctrica y la estructura de soporte para montar los paneles solares.

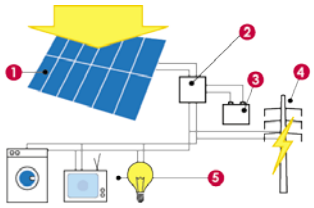


1. paneles solares **3.** red
2. inversores **4.** artefactos eléctricos

Fig. 1.6: *Esquema de un sistema solar conectado a la red*

c) Sistemas de emergencias

Los sistemas de generación de electricidad solar de emergencia, se instalan cuando hay conexión a la red de distribución pública, pero el suministro de electricidad no es confiable. El sistema solar de emergencia puede ser utilizado para suministrar electricidad durante los cortes de fluido eléctrico de la red (apagones). Un sistema solar de emergencia pequeño puede generar corriente eléctrica para cubrir las necesidades más importantes, tales como iluminación y alimentación de equipos de computación y telecomunicaciones (teléfono, radio, fax, etc.). Un sistema más grande puede ser dimensionado para abastecer un refrigerador durante un corte de fluido. Cuanta más energía consuman los artefactos y mayor sea la duración normal de los cortes de fluido, más grande debe ser el sistema fotovoltaico.



1. paneles solares
2. inversor
3. batería
4. red
5. artefactos eléctricos

Fig. 1.7: *Esquema de un sistema de emergencia*

1.3.2 Mantenimiento de sistemas fotovoltaicos.

Las instalaciones fotovoltaicas requieren un mantenimiento mínimo y sencillo, que se reduce a las siguientes operaciones:

- Paneles: requieren un mantenimiento nulo o muy escaso, debido a su propia configuración: no tienen partes móviles y las células y sus conexiones internas están encapsuladas en varias capas de material protector. Es conveniente hacer una inspección general 1 ó 2 veces al año: asegurarse de que las conexiones entre paneles y al regulador están bien ajustadas y libres de corrosión. En la mayoría de los casos, la acción de la lluvia elimina la necesidad de limpieza de los paneles; en caso de ser necesario, simplemente utilizar agua y algún detergente no abrasivo.

- Regulador: la simplicidad del equipo de regulación reduce sustancialmente el mantenimiento y hace que las averías sean muy escasas. Las operaciones que se pueden realizar son las siguientes: observación visual del estado y funcionamiento del regulador; comprobación del conexionado y cableado del equipo; observación de los valores instantáneos del voltímetro y amperímetro: dan un índice del comportamiento de la instalación.

- Acumulador: es el elemento de la instalación que requiere una mayor atención; de su uso correcto y buen mantenimiento dependerá en gran medida su duración. Las operaciones usuales que deben realizarse son las siguientes:

Comprobación del nivel del electrolito (cada 6 meses aproximadamente): debe mantenerse dentro del margen comprendido entre las marcas de "Máximo" y "Mínimo". Si no existen estas marcas, el nivel correcto del electrolito es de 20 mm por encima del protector de separadores. Si se observa un nivel inferior en alguno de los elementos, se deben rellenar con agua destilada o desmineralizada. No debe rellenarse nunca con ácido sulfúrico.

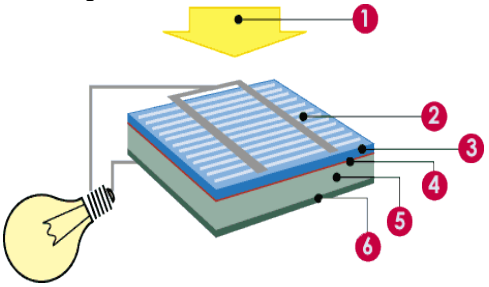
Al realizar la operación anterior debe comprobarse también el estado de los terminales de la batería; debe limpiarse de posibles depósitos de sulfato y cubrir con vaselina neutra todas las conexiones.

Medida de la densidad del electrolito (si se dispone de un densímetro): con el acumulador totalmente cargado, debe ser de 1,240 +/- 0,01 a 20 grados Celsius. Las densidades deben ser similares en todos los vasos. Diferencias importantes en un elemento es señal de posible avería.

1.4 Elementos que intervienen en un sistema autónomo

1.4.1 Celdas solares

Las celdas solares son fabricadas a base de materiales que convierten directamente la luz solar en electricidad. Hoy en día, la mayor parte de celdas solares utilizadas a nivel comercial son de silicio (símbolo químico: Si). El silicio es lo que se conoce como un semiconductor. Este elemento químico se encuentra en todo el mundo bajo la forma de arena, que es dióxido de silicio (SiO_2), también llamado cuarcita. Otra aplicación del silicio semiconductor se encuentra en la industria de la microelectrónica, donde es empleado como material base para los *chips*.



1. luz (fotones)
2. contacto frontal
3. capa negativa
4. capa de desviación
5. capa positiva
6. contacto posterior

Fig. 1.8: *Composición de una celda solar*

a) Estructura de una celda solar

Las celdas solares de silicio pueden ser de tipo monocristalinas, policristalinas o amorfas. La diferencia entre ellas radica en la forma como los átomos de silicio están dispuestos, es decir, en la estructura cristalina. Existe, además, una diferencia en la eficiencia. Por eficiencia se entiende el porcentaje de luz solar que es transformado en electricidad. Las celdas solares de silicio monocristalino y policristalino tienen casi el mismo y más alto nivel de eficiencia con respecto a las de silicio amorfo.

Una celda solar típica está compuesta de capas. Primero hay una capa de contacto posterior y, luego, dos capas de silicio. En la parte superior se encuentran los contactos de metal frontales con una capa de antireflexión, que da a la celda solar su típico color azul.

Durante la última década, se ha estado desarrollando nuevos tipos de celdas solares de materiales diversos, entre las que encontramos, por ejemplo, a las celdas de película delgada y a las celdas de CIS (diseleniuro de indio de cobre) y CdTe (telururo de cadmio) [10]. Estas están comenzando a ser comercializadas.

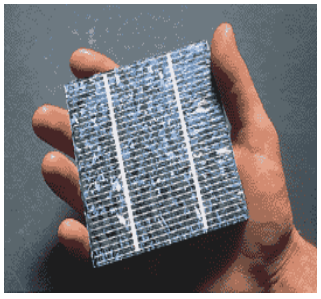


Fig. 1.9: *Celda solar policristalina*

b) Eficiencias de celda:

Tabla 1.1: *Eficiencia de diferentes tipos de celdas*

Monocristalina:	12-15 %
Policristalina:	11-14 %
Amorfa:	6-7 %
telururo de cadmio:	7-8 %

1.4.2 Paneles solares

a) Fabricación de un panel solar fotovoltaico

Un panel fotovoltaico está formado por un conjunto de células solares conectadas eléctricamente entre sí en serie y paralelo hasta conseguir el voltaje adecuado para su utilización.

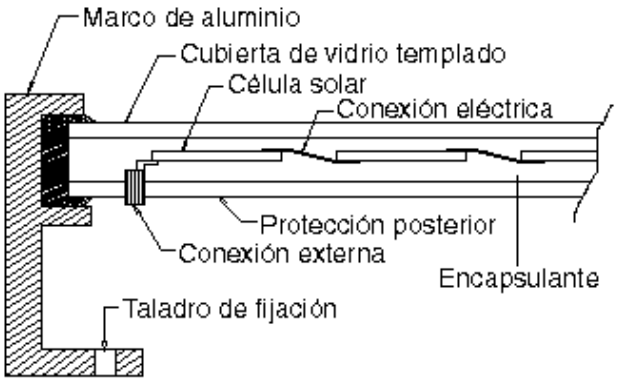


Fig. 1.10: *Corte transversal de un panel fotovoltaico*

Este conjunto de células está envuelto por unos elementos que le confieren protección frente a los agentes externos y rigidez para acoplarse a las estructuras que los soportan. Los elementos son los siguientes:

- Encapsulante, constituido por un material que debe presentar una buena transmisión a la radiación y una degradabilidad baja a la acción de los rayos solares.

- Cubierta exterior de vidrio templado, que, aparte de facilitar al máximo la transmisión luminosa, debe resistir las condiciones climatológicas más adversas y soportar cambios bruscos de temperatura.

- Cubierta posterior, constituida normalmente por varias capas opacas que reflejan la luz que ha pasado entre los intersticios de las células, haciendo que vuelvan a incidir otra vez sobre éstas.

- Marco de metal, normalmente de aluminio, que asegura rigidez y estanqueidad al conjunto, y que lleva los elementos necesarios (generalmente taladros) para el montaje del panel sobre la estructura soporte.

- Caja de terminales: incorpora los bornes para la conexión del módulo.

- Diodo de protección: impiden daños por sombras parciales en la superficie del panel.

Los paneles solares están compuestos por celdas solares. Dado que una sola celda solar no produce energía suficiente para la mayor parte de aplicaciones, se les agrupa en paneles solares, de modo que, en conjunto, generan una mayor cantidad de electricidad. Los paneles solares (también denominados módulos fotovoltaicos o FV) son fabricados en diversas formas y tamaños. Los más comunes son los de 50 Wp (*watt* pico), que producen un máximo de 50 *watts* de electricidad solar bajo condiciones de luz solar plena (un nivel de radiación aproximadamente de 1 kWh/m2), y que están compuestos por celdas solares de silicio. Dichos paneles miden 0,5 m2 aproximadamente. Sin embargo, se puede escoger entre una amplia variedad de paneles más grandes y más pequeños disponibles en el mercado. Los paneles solares pueden conectarse con el fin de generar una mayor cantidad de electricidad solar (dos paneles de 50 Wp conectados equivalen a un panel de 100 Wp).

Fig. 1.11: *Paneles solares*

b) Rendimiento de los paneles solares

Factores: Fundamentalmente de la intensidad de la radiación luminosa y de la temperatura de las células solares.

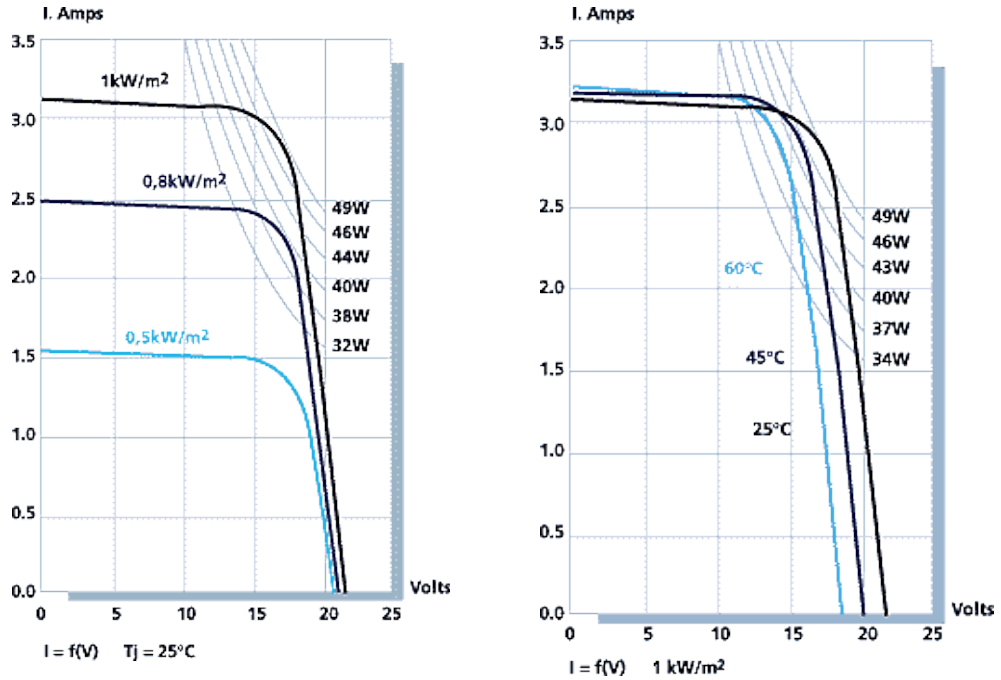


Fig. 1.12: *Variación de Intensidad y tensión con la radiación y la temperatura según potencia nominal*

La intensidad de corriente que genera el panel aumenta con la radiación, permaneciendo el voltaje aproximadamente constante. En este sentido, tiene mucha importancia la colocación de los paneles (su orientación e inclinación respecto a la horizontal), ya que los valores de la radiación varían a lo largo del día en función de la inclinación del sol respecto al horizonte.
El aumento de temperatura en las células supone un incremento en la corriente, pero al mismo tiempo una disminución mucho mayor, en proporción, de la tensión. El efecto global es que la potencia del panel disminuye al aumentar la temperatura de trabajo del mismo. Una radiación de 1.000 W/m2 es capaz de calentar un panel unos 30 grados por encima de la temperatura del aire circundante, lo que reduce la tensión en 2 mV/(célula * grado) * 36 células * 30 grados = 2,16 Voltios y por tanto la potencia en un 15% [5]. Por ello es importante colocar los paneles en un lugar en el que estén bien aireados.

La eficiencia de los paneles solares disponibles en el mercado fluctúa entre 5-15%. Esto significa que 5-15% de la energía de toda la luz solar que llega a la celda será, en efecto, transformada en electricidad. Los laboratorios de investigación en todo el mundo están desarrollando nuevos materiales con eficiencias mayores (hasta 30%). Los costos de producción son igualmente importantes. Algunas nuevas tecnologías (tales como las celdas de película delgada) permiten la producción a gran escala, lo que reduciría significativamente su costo.

Los paneles fotovoltaicos generan electricidad incluso en días nublados, aunque su rendimiento disminuye. La producción de electricidad varía linealmente a la luz que incide sobre el panel; un día totalmente nublado equivale aproximadamente a un 10% de la intensidad total del sol, y el rendimiento del panel disminuye proporcionalmente a este valor.

La mejoría del rendimiento con la incorporación de un sistema con seguimiento solar depende del clima y del tipo de aplicación. En condiciones ideales el rendimiento del sistema puede mejorar hasta un 40%, pero el mayor costo que supone no compensa el aumento que se consigue. Su aplicación se limita a aquellos casos en que el mayor rendimiento coincide con la mayor demanda (es el caso de sistemas de bombeo para el ganado en regiones muy secas).

c) Vida útil de un panel solar fotovoltaico

Teniendo en cuenta que el panel carece de partes móviles y que las células y los contactos van encapsulados en una robusta resina sintética, se consigue una muy buena fiabilidad junto con una larga vida útil, del orden de 30 años o más. Además si una de las células falla, esto no afecta al funcionamiento de las demás, y la intensidad y voltaje producidos pueden ser fácilmente ajustados añadiendo o suprimiendo células.
Además, los paneles van protegidos en su cara exterior con vidrio templado, que permite aguantar condiciones meteorológicas muy duras tales como el hielo, la abrasión, cambios bruscos de temperatura, o los impactos producidos por el granizo. Una prueba estándar para su homologación consiste en lanzar (con un cañón neumático) una bola de hielo de dimensiones y consistencia preestablecidas al centro del cristal.

1.4.3 Baterías

Una batería es esencialmente un recipiente lleno de químicos que producen electrones. Las reacciones químicas son capaces de producir electrones y este fenómeno es llamado reacción electroquímica. En los llamados sistemas solares autónomos o sistemas fotovoltaicos domiciliarios (SFD), las baterías almacenan electricidad que será utilizada durante la noche. Asimismo, suministran electricidad durante periodos de escasez o ausencia de luz solar, necesaria para que el panel solar produzca energía. La duración del periodo que puede ser cubierto está determinada por la demanda de electricidad y el tamaño de la batería de almacenamiento.

Las baterías están disponibles en diversas formas y tamaños. Las de 12V son las más utilizadas. Si las baterías son nuevas y son del mismo tipo y tamaño, pueden ser conectadas para incrementar la capacidad del almacenamiento de batería.

Fig. 1.13: *Baterías*

a) Funcionamiento

Si se examina una batería, esta tiene dos terminales. Una terminal está marcada (+) positivo mientras la otra está marcada (-) negativo. En una AA, o C (baterías más comunes) los extremos son los terminales. En una batería de auto existen dos grandes tubos que actúan de terminales.

Los electrones se agrupan en la terminal negativa de la batería. Si se conecta un cable entre las terminales positivas y negativas, los electrones pasarán de la terminal negativa a la positiva tan rápido como puedan (y descargarán a la batería muy rápido -esto también tiende a ser peligroso, especialmente con baterías grandes, así que no es algo que debería hacer-). Normalmente se coloca algún tipo de artefacto a la batería con el cable. Este artefacto puede ser una bombilla, un motor, un circuito electrónico como un radio, etc.

Dentro de la batería misma, una reacción química produce electrones, y la velocidad de la producción de electrones hecha por esta reacción (la resistencia interna de la batería) controla cuántos electrones pueden pasar por las terminales. Los electrones pasan de la batería al cable, y deben viajar de la terminal negativa a la positiva para que la reacción química se lleve a cabo. Es por eso que una batería puede guardarse por un año y todavía conserva su energía plenamente -a menos que los electrones corran hacia la terminal positiva, la reacción química no se efectuará-. Una vez que se conecte el cable, la reacción empieza.

b) Características que definen su comportamiento
Son fundamentalmente dos: la capacidad en Amperios hora y la profundidad de la descarga.

Capacidad en Amperios hora: Los Amperios hora de una batería son simplemente el número de Amperios que proporciona multiplicado por el número de horas durante las que circula esa corriente.
Sirve para determinar, en una instalación fotovoltaica, cuanto tiempo puede funcionar el sistema sin radiación luminosa que recargue las baterías. Esta medida de los días de autonomía es una de las partes importantes en el diseño de la instalación.

Teóricamente, por ejemplo, una batería de 200 Ah puede suministrar 200 A durante una hora, ó 50 A durante 4 horas, ó 4 A durante 50 horas, o 1 A durante 200 horas.
No obstante esto no es exactamente así, puesto que algunas baterías, como las de automoción, están diseñadas para producir descargas rápidas en cortos períodos de tiempo sin dañarse. Sin embargo, no están diseñadas para largos períodos de tiempo de baja descarga. Es por ello que las baterías de automoción no son las más adecuadas para los sistemas fotovoltaicos.

Existen factores que pueden hacer variar la capacidad de una batería:

- Ratios de carga y descarga. Si la batería es cargada o descargada a un ritmo diferente al especificado, la capacidad disponible puede aumentar o disminuir. Generalmente, si la batería se descarga a un ritmo más lento, su capacidad aumentará ligeramente. Si el ritmo es más rápido, la capacidad se reducirá.

- Temperatura. Otro factor que influye en la capacidad es la temperatura de la batería y la de su ambiente. El comportamiento de una batería se cataloga a una temperatura de 27 grados. Temperaturas más bajas reducen su capacidad significativamente. Temperaturas más altas producen un ligero aumento de su capacidad, pero esto puede incrementar la pérdida de agua y disminuir el número de ciclos de vida de la batería.

Profundidad de descarga: La profundidad de descarga es el porcentaje de la capacidad total de la batería que es utilizada durante un ciclo de carga/descarga.

Las baterías de "ciclo poco profundo" se diseñan para descargas del 10 al 25% de su capacidad total en cada ciclo. La mayoría de las baterías de "ciclo profundo" fabricadas para aplicaciones fotovoltaicas se diseñan para descargas de hasta un 80% de su capacidad, sin dañarse. Los fabricantes de baterías de Níquel-Cadmio aseguran que pueden ser totalmente descargadas sin daño alguno.

La profundidad de la descarga, no obstante, afecta incluso a las baterías de ciclo profundo. Cuanto mayor es la descarga, menor es el número de ciclos de carga que la batería puede tener.

c) Protección de las Baterías.
Por lo general, las baterías son la parte más delicada de un sistema solar y la primera en ser reemplazada. A continuación, se presenta algunas recomendaciones para ayudar a extender el tiempo de vida de la batería:

1. El uso de un controlador de carga es altamente recomendable. Éste desconecta las cargas cuando la batería se encuentra casi completamente descargada. Todos los sistemas solares domiciliarios estándar cuentan con un controlador de carga.
2. Asegúrese de que haya relación entre el número de paneles solares, el tamaño de las baterías y el número de cargas eléctricas (luces, artefactos eléctricos) y sus respectivos consumos.
3. Observe su controlador de carga para verificar el estado de carga de la batería (cuán cargada se encuentra). Por lo general, el controlador está provisto de un indicador luminoso rojo, que se enciende cuando la batería está descargada, y uno verde, que se enciende cuando está completamente cargada. Procure que el indicador verde permanezca encendido el mayor tiempo posible. Esto extenderá el tiempo de vida de la batería.
4. Dé mantenimiento a su batería (llénela con agua destilada) 3 veces al año como mínimo (si no se trata de una batería sin necesidad de mantenimiento).
5. Si tiene la oportunidad de cargar al máximo la batería utilizando un cargador /generador, hágalo (una vez al mes), pues esto ayuda a extender el tiempo de vida de la batería.
6. Nunca ignore las indicaciones del controlador de carga con el fin de extraer hasta la última gota de energía de la batería. Esto la arruinaría.

Esta tesis tiene por objetivo principal hacer duradera la vida útil de la batería ya que en un sistema fotovoltaico sus condiciones óptimas se deterioran más rápido. Por lo tanto es pertinente estudiar la naturaleza interna que genera las facultades que una batería posee.
Es algo generalizado el uso de las baterías de Plomo ácido en los sistemas fotovoltaicos. Por lo tanto centraremos el estudio en este tipo particular.

1.4.3.1 Baterías de Plomo ácido.

Estas baterías se componen de varias placas de plomo en una solución de ácido sulfúrico. La placa consiste en una rejilla de aleación de Plomo con una pasta de óxido de Plomo incrustada sobre la rejilla. La solución de ácido sulfúrico y agua se denomina electrolito.
El material de la rejilla es una aleación de Plomo porque el Plomo puro es un material físicamente débil, y podría quebrarse durante el transporte y servicio de la batería.

Normalmente la aleación es de Plomo con un 2-6% de Antimonio. Cuanto menor es el contenido en Antimonio, menos resistente será la batería durante el proceso de carga. La menor cantidad de Antimonio reduce la producción de Hidrógeno y Oxígeno durante la carga, y por tanto el consumo de agua. Por otra parte, una mayor proporción de Antimonio permite descargas más profundas sin dañarse las placas, lo que implica una mayor duración de vida de las baterías. Estas baterías de Plomo-Antimonio son del tipo de "ciclo profundo".

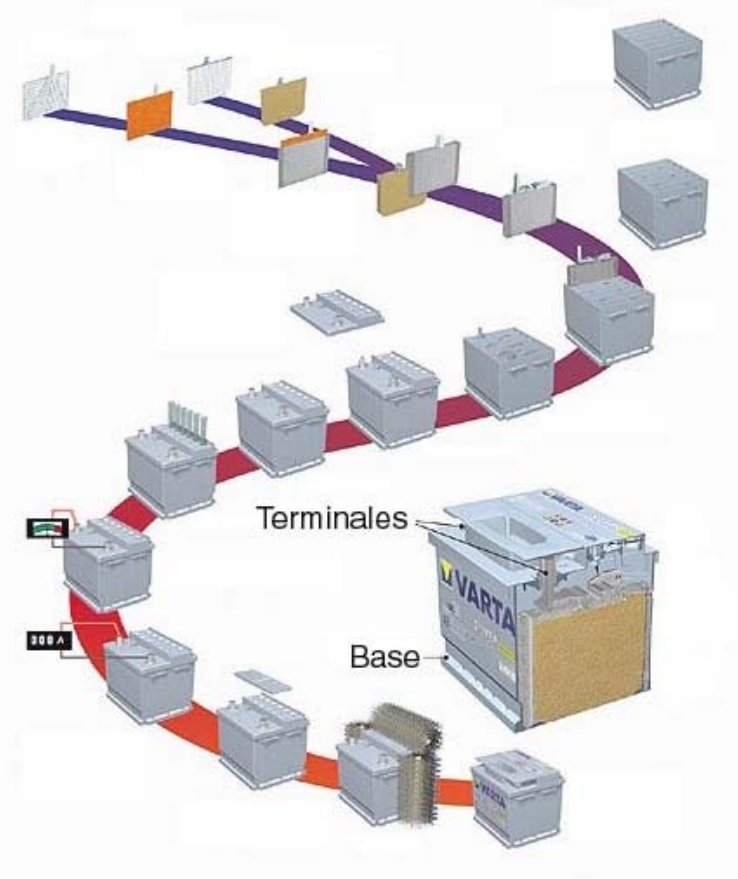


Fig. 1.14: *Construcción de una batería monoblock (VARTA)*

El Cadmio y el Estroncio se utilizan en lugar del Antimonio para fortalecer la rejilla. Estos ofrecen las mismas ventajas e inconvenientes que el Antimonio, pero además reducen el porcentaje de autodescarga que sufre la batería cuando no está en uso.
El Calcio fortalece también la rejilla y reduce la autodescarga. Sin embargo, el Calcio reduce la profundidad de descarga recomendada en no más del 25%. Por otra parte, las baterías de Plomo-Calcio son del tipo de "ciclo poco profundo".
Las placas positiva y negativa están inmersas en una solución de ácido sulfúrico y son sometidas a una carga de "formación" por parte del fabricante. La dirección de esta carga da lugar a que la pasta sobre la rejilla de las placas positivas se transforme en dióxido de Plomo. La pasta de las placas negativas se transforman en Plomo esponjoso. Ambos materiales son altamente porosos, permitiendo que la solución de ácido sulfúrico penetre libremente en las placas.
Las placas se alternan en la batería, con separadores entre ellas, que están fabricados de un material poroso que permite el flujo del electrolito. Son eléctricamente no conductores. Pueden ser mezclas de silicona y plásticos o gomas.
Los separadores pueden ser hojas individuales o "sobres". Los sobres son manguitos, abiertos por arriba, que se colocan únicamente sobre las placas positivas.

Un grupo de placas positivas y negativas, con separadores, constituyen un "elemento". Un elemento en un contenedor inmerso en un electrolito constituye una "celda" de batería.
Placas más grandes, o mayor número de ellas, suponen una mayor cantidad de Amperios hora que la batería puede suministrar.

Independientemente del tamaño de las placas, una celda suministrará sólo una tensión nominal de 2 voltios (para Plomo-ácido). Una batería está constituida por varias celdas o elementos conectados en serie, interna o externamente, para incrementar el voltaje a unos valores normales a las aplicaciones eléctricas. Por ello, una batería de 6 V se compone de tres celdas, y una de 12 V de 6.
Las placas positivas por un lado, y las negativas por otro, se interconectan mediante terminales externos en la parte superior de la batería.

Sulfatación de una batería de Plomo-ácido
Si una batería de Plomo-ácido se deja en un estado de descarga profunda durante un período prolongado de tiempo, se producirá su sulfatación. Parte del sulfuro del ácido se combinará con plomo procedente de las placas para formar sulfato de plomo.
Si la batería no se rellena con agua periódicamente, parte de las placas quedarán expuestas al aire, y el proceso se verá acelerado.
El sulfato de plomo recubre las placas de forma que el electrolito no puede penetrar en ellas. Esto supone una pérdida irreversible de capacidad en la batería que, incluso con la adición de agua, no se puede recuperar.

Las causas más habituales de sulfatación de una batería son:

- Dejarla descargada durante mucho tiempo.
- Añadir ácido puro al electrolito.
- Sobrecargas demasiado frecuentes.
- No haber añadido agua destilada en el momento oportuno.
- El trasvase de electrolito de unos vasos a otros.

Los síntomas más evidentes de sulfatación son:

- El densímetro registra siempre una densidad baja del electrolito, a pesar de que el elemento siempre se somete a la misma carga que los otros elementos.
- La tensión es inferior a la de los demás elementos durante la descarga y superior durante la carga.
- Es imposible cargar la batería a toda su capacidad.
- Las dos placas, positiva y negativa, tienen un color claro.
- En casos extremos, uno de los terminales sobresale más de lo normal debido a la deformación de las placas.

Clase de agua de reposición.
Únicamente agua destilada, o agua de lluvia. Debe guardarse en recipientes de vidrio bien limpios. El agua de lluvia, aunque es la mejor, debe recogerse sin que se ponga en contacto con metales (techos de zinc, etc), porque entonces adquiere impurezas. La recogida por un techo de tejas cerámicas o por una lona impermeable, por ejemplo, reúne buenas condiciones.

Diferencias entre las baterías de Plomo-ácido y las de Níquel-Cadmio.

Las baterías de Níquel-Cadmio tienen una estructura física similar a las de Plomo-ácido. En lugar de Plomo, se utiliza hidróxido de Níquel para las placas positivas y óxido de Cadmio para las negativas. El electrolito es hidróxido de Potasio.

El voltaje nominal de un elemento de batería de Ni-Cd es de 1,2 V, en lugar de los 2 V de los elementos de batería de Plomo-ácido.

Las baterías de Ni-Cd , aguantan procesos de congelación y descongelación sin ningún efecto sobre su comportamiento. Las altas temperaturas, tienen menos incidencia que en las de Plomo-ácido. Los valores de autodescarga, oscilan entre 3 y 6% al mes. Les afectan menos las sobrecargas. Pueden descargarse totalmente sin sufrir daños. No tienen peligro de sulfatación. Su capacidad para aceptar un ciclo de carga es independiente de la temperatura. El costo de una batería de Ni-Cd es mucho más elevado que el de una de Plomo-ácido; no obstante, tiene un mantenimiento más bajo y una vida más larga. Esto las hace aconsejables para lugares aislados o de acceso peligroso. Las baterías de Ni-Cd no pueden probarse con la misma fiabilidad que las de Plomo-ácido. Por tanto, si es necesario controlar el estado de carga, las baterías de Ni-Cd no son la mejor opción. El Ni-Cd presenta el llamado "efecto memoria": la batería "recuerda" la profundidad de descarga y reduce su capacidad efectiva. Esto se debe a que el compuesto químico que se forma en una placa cargada tiende a cristalizar, por lo que si se le deja el tiempo suficiente queda inutilizada, perdiendo su capacidad. Este proceso no es irreversible pero si de difícil reversión.

Efectos sobre la capacidad y el voltaje de la conexión en serie o paralelo.

Las baterías pueden conectarse en serie para incrementar el voltaje, o en paralelo para incrementar la capacidad en Amperios hora del sistema de acumulación.

Al conectar en serie/paralelo se incrementan tanto el voltaje como la capacidad.

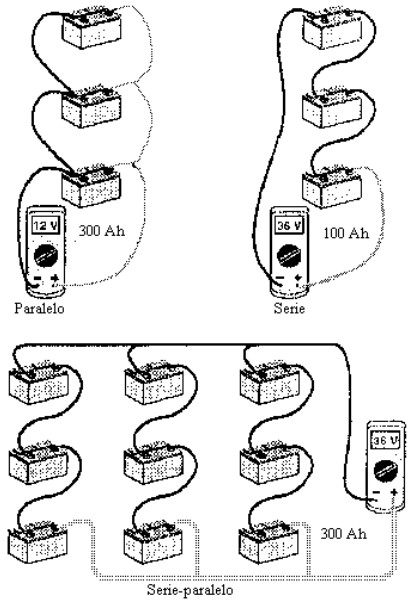


Fig. 1.15: *Baterías conectadas en paralelo, en serie y en serie paralelo*

Efecto de descargar rápidamente una batería de Plomo ácido.
En primer lugar, no se obtiene toda la energía que es capaz de proporcionar la batería. Por ejemplo, una batería descargada en 72 horas devuelve aproximadamente el doble de energía que si se descargarse en sólo 8 horas.
Además, las descargas rápidas producen deformaciones y la prematura desintegración de las placas de los elementos, que se depositan en el fondo de los recipientes en forma pulverulenta hasta llegar a cortocircuitar ambas placas, inutilizando la batería.

Efectos que produce el calor.
La elevación de temperatura es sumamente perjudicial para las baterías. Si la temperatura de los recipientes es superior a unos 40 grados, es necesario disminuir el régimen de carga.
Por esto para la instalación de las baterías se debe buscar un sitio donde la temperatura sea templada, evitando los lugares calientes. Es preciso también evitar temperaturas inferiores a 0 grados ya que entonces la resistencia interna de las baterías aumenta mucho.

Empleo de un regulador de carga en una instalación fotovoltaica
La función primaria de un regulador de carga en un sistema fotovoltaico es proteger a la batería de sobrecargas o descargas excesivas. Cualquier instalación que utilice cargas impredecibles, intervención del usuario, sistema de acumulación optimizado o infradimensionado (para minimizar inversión inicial), o cualquier otra característica que pueda sobrecargar o descargar excesivamente la batería, requiere un regulador de carga. La falta del mismo puede ocasionar una reducción de la vida útil de la batería y una reducción de la disponibilidad de carga.
Los sistemas con cargas pequeñas, predecibles y continuas pueden diseñarse para funcionar sin necesidad de regulador. Si el sistema lleva un acumulador sobredimensionado y el régimen de descarga nunca va a superar la profundidad de descarga crítica de la batería, se puede prescindir del regulador.

1.4.4 Regulador

Como se ha mencionado anteriormente, el objetivo del regulador es alargar la vida útil de la batería, evitando que esta sufra cargas y descargas profundas que estén fuera de sus límites permisibles.
La carga de la batería se mide en Ah (ver sección 1.4.3-b) y está directamente relacionado en proporción directa a la densidad del electrolito.
Entonces, una forma de medir la capacidad es a través de la medida de la densidad o gravedad específica del líquido contenido en el acumulador (electrolito). La densidad expresa cuanto pesa el electrolito en comparación con la misma cantidad de agua, y se mide con un densímetro o hidrómetro [15]. El densímetro más común es el utilizado para automoción, que indica la carga en porcentaje. Presenta el inconveniente de que está calibrado para el electrolito utilizado en acumuladores de arranque y no estacionarios, por lo que marcará siempre menos de lo real (50% para un acumulador estacionario completamente cargado).

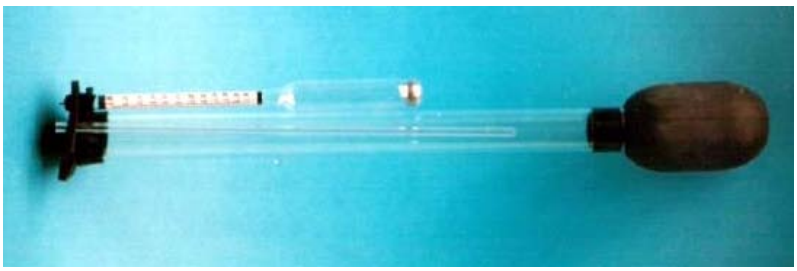


Fig. 1.16: *Un densímetro (sin ensamblar) de los usados en acumuladores estacionarios*

Es labor del regulador evitar las sobrecargas y descargas muy profundas; para realizar esto, instante a instante, el regulador debe medir de alguna manera el estado de carga de la batería y compararlo con niveles permisibles tanto inferiores como superiores.
Sería caro utilizar un densímetro como transductor y a través de este obtener una medida de la carga de la batería.
Hay una variable más, a la que la capacidad de carga se encuentra asociado en el proceso de carga o de descarga de una batería, veamos la siguiente gráfica deducible.

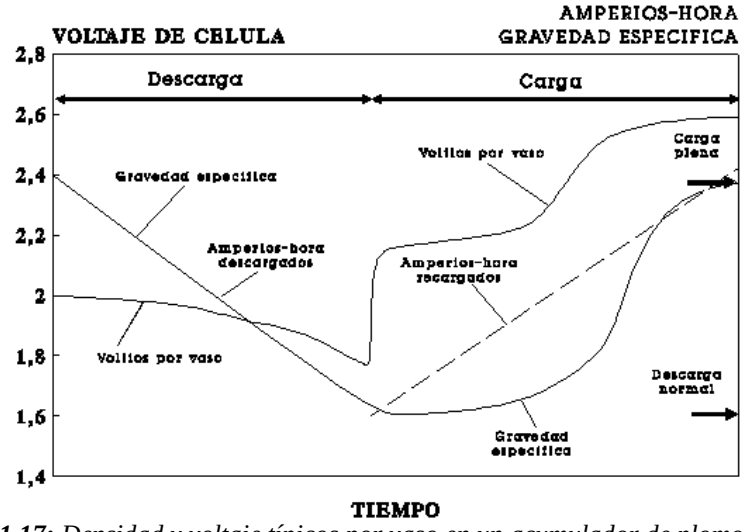


Fig. 1.17: *Densidad y voltaje típicos por vaso en un acumulador de plomo-ácido*

Cuanto mayor es la gravedad específica del electrolito, mayor es el estado de carga. El voltaje de cada vaso o celda, y por tanto el de la batería, es también mayor. Por lo tanto un buen acierto para medir el estado de carga de una batería en buen estado es medir el voltaje de carga de esta. Es seguro que este valor nos servirá para tener una idea del nivel de carga de la batería comparando la tensión con las correspondientes a un nivel de carga inferior y uno superior [10].

Tabla 1.2: *Estado de carga, densidad y voltaje de un acumulador de Plomo-ácido*
Niveles obtenidos de Isofotón

Estado	Densidad	Voltios/vaso	Voltios/conjunto
Cargada	1,265	2,12	12,70
Cargada 75%	1,225	2,10	12,60
Cargada 50%	1,190	2,08	12,45
Cargada 25%	1,155	2,03	12,20
Descargada	1,120	1,95	11,70

La tarea del regulador se simplifica en controlar constantemente la tensión de la batería durante su carga, cortando el paso de la corriente cuando esté totalmente cargada y detectar cuando la tensión de las baterías ha descendido y reanudar la carga de ésta.

Fig. 1.18: *Regulador de carga (Isofotón)*

El regulador es la parte fundamental de un sistema autónomo ya que permite el tránsito de corriente del panel hacia la batería y de la batería hacia la carga. Por lo tanto el regulador es el que protege a la batería de sobrecargas, bajas tensiones y de cortocircuitos. En la figura se muestra un regulador hecho por la empresa española “Isofotón” [10].
Los temas principales de la presente tesis son:

- Diseñar un regulador inteligente, es decir que esté gobernado por un microcontrolador, el cual mediante conversiones analógicas digitales obtendrá el nivel de carga (tensión) de la batería instante a instante, para así tomar una decisión. El conjunto de rutinas incluidas en el microcontrolador hará que éste posea un control sobre la situación de carga y descarga de la batería, haciendo esta situación la más conveniente para hacer duradera la vida de la batería.

- Mediante la comunicación entre la tarjeta reguladora y la tarjeta de adquisición de datos y la respectiva descarga de ésta a la PC, obtener una base de datos de las tensiones en la carga y descarga a través del tiempo. Mediante un adecuado proceso de los datos obtenidos se puede analizar las condiciones del lugar en donde este situado el panel y las condiciones de la batería.

A continuación se da el mínimo de especificaciones técnicas que un regulador debe cumplir:

a) Características Eléctricas

Tensión Nominal.
Intensidad máxima de generación.
Intensidad máxima de consumo.
Sobrecarga Admisible.
Autoconsumo.
Perdida máxima generación /consumo.

b) Características Constructivas

Tipo de regulación.
Selección de Batería. Tubular Abierta, *Hoppecke, Monoblock, Gel*.
Sistema de regulación. Igualación, Carga profunda, Flotación.
Señalización del estado de carga. Mediante *Leds*.
Desconexión del consumo por baja tensión de batería con rearme automático.
Alarmas por alta y baja tensión de batería, sobrecarga y cortocircuito mediante *Leds* y señal acústica.
Protección contra polaridad inversa en paneles, batería y consumo.
Protección contra sobrecarga temporizada en paneles y consumo.
Protección contra sobretensiones en paneles, batería y consumo.
Protección contra desconexión de batería.
Rango de funcionamiento a plena carga
Rearme manual en caso de cortocircuito, pulsado reset, previamente solucionado el cortocircuito.

Resumiendo estas especificaciones, determinamos los parámetros principales de un regulador:

Alarma Tensión Alta
Banda de Igualación
Tensión de Carga Profunda
Banda de Flotación
Tensión de Recarga Profunda
Alarma Tensión Baja
Tensión Desconexión consumo
Tensión Reconexión consumo

Teniendo como objetivo el cumplimiento de estas características se procedió al diseño del regulador que las cumpla (capítulo III). Es motivo fundamental de esta tesis hacer el control con un microcontrolador PIC 16c711, cuyas facultades harán que la tarjeta aumente en funcionalidad (*software* y *hardware*), ya que aparte del regulador, se ha implementado una tarjeta de adquisición de datos que nos permitirá almacenar datos de tensión de la batería para así relacionar la energía proporcionada del sol hacia el usuario, aproximando un nivel de corriente promedio relacionada con los amperios hora de la batería.
En el capítulo II se estudiará al microcontrolador PIC 16c711.

CAPÍTULO-2

DESCRIPCIÓN DEL MICRONCOTROLADOR PIC 16C711

El presente capítulo tiene como objetivo:

Una descripción detallada del microcontrolador PIC 16C711, en cuanto a *hardware* y *software*.

La implementación de sus registros y las funciones especiales que nos permiten especialmente la etapa de conversión analógica /digital.

Se darán también las razones del uso de este microcontrolador en específico, así como las ventajas y desventajas que presenta.

2.1 Introducción al Microcontrolador PIC 16C711.

Los microcontroladores PIC 16CXXX de *Microchip* se encuentran en aparatos muy diversos como son: programadores domésticos, telemandos, termostatos electrónicos, etc.
El motivo de su éxito se encuentra en su costo especialmente bajo, sus altas prestaciones y su reducido juego de instrucciones (RISC), así como su estructura interna del tipo *Harvard*.

HARVARD:

Casi todos los microcontroladores actuales utilizan la estructura de *Von Neuman*, en la que la memoria de programa contiene instrucciones y datos mezclados y se dispone de un único *bus* llamado *bus* de datos por el que circulan a la vez los códigos de las instrucciones y los datos asociados a ellas.
Esta arquitectura presenta algunos problemas cuando se demanda rapidez, con lo que es preferible utilizar la arquitectura denominada *Harvard* en la que las instrucciones y los datos están perfectamente diferenciados y se emplean *buses* distintos.
Esta estructura no altera el funcionamiento desde el punto de vista del usuario y la velocidad de ejecución de los programas es impresionante.

RISC:

Significa *Reduced Instruction Set Computer* (Ordenador con juego de instrucciones reducido)
Pero no solamente significa esto sino que debe disponer de una estructura "*pipeline*" , lo cual permite ejecutar como mínimo una instrucción mientras busca la siguiente, además también permite aumentar la velocidad de ejecución en relación a los microcontroladores clásicos denominados CISC (*Completex I.S.C*).
Además los circuitos RISC, deben ejecutar todas las instrucciones a la misma velocidad.

El PIC 16C711, es un circuito RISC con lo que gana en velocidad de trabajo y no es preciso trabajar con un centenar de instrucciones, sino tan solo 35, sin que esto disminuya las prestaciones.

2.2 Señales Disponibles

2.2.1 Alimentación

VSS : Masa general.
VDD: Alimentación (positiva) debe estar comprendida entre 4 y 6v.
En elementos con el sufijo HS estará entre 4,5 y 5,5v.
En modo SLEEP deberá tener una alimentación de 1,5v para que la RAM interna conserve sus contenidos.
En general, todas las reglas aplicables a la alimentación de microprocesadores son aplicables también aquí como por ejemplo seria el condensador de desacoplo (0,1uF).

2.2.2 Reloj (*Clock*)

Hay dos pines de reloj que son OSC1 y OSC2 pudiéndose utilizar varios tipos de reloj. En las versiones con ventana, el tipo de reloj utilizado se puede seleccionar mediante la programación de un registro especial accesible únicamente al programar el circuito, esto se puede hacer con reloj de cristal de cuarzo o como célula RC.
En las versiones sin ventana u OTP el reloj viene predeterminado en la cápsula teniendo cuatro encapsulados distintos en función de dicho reloj interno:

XT: Oscilador de cuarzo estándar con una frecuencia máxima de 4 Mhz.

HS: Lo mismo pero de hasta 20 MHz (*High Speed*).

RC: Oscilador RC funciona hasta a 4 MHz pero con una estabilidad menor que el cuarzo.

LP (*Low power*): Versión para cuarzo, especial para aplicaciones de bajo consumo, su frecuencia está limitada a 200 MHz[1].

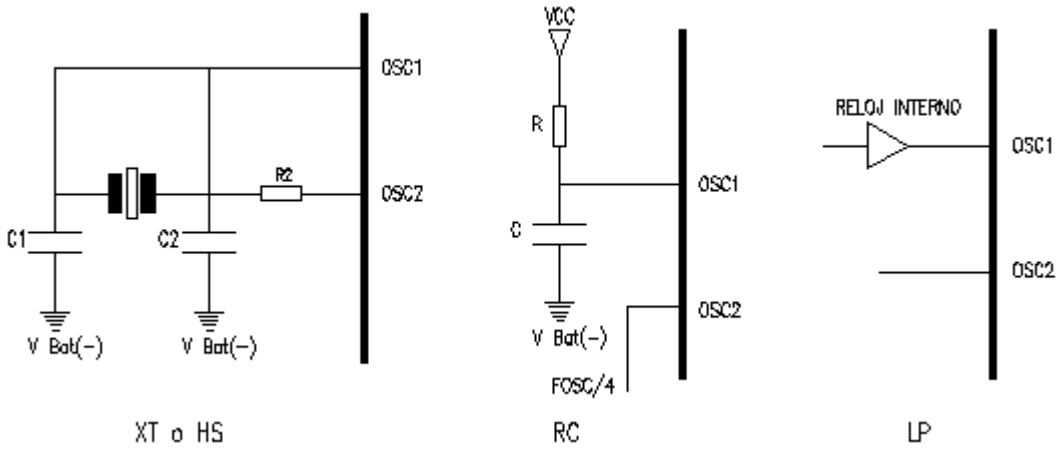


Fig. 2.1: *Clock*

[1] Para mayor información consultar el *data sheet* del PIC 16CXXX en la página *web* www.microchip.com.

2.2.3 RESET

La señal *reset* tiene como pin el denominado MLCR, y se le añade una circuiteria para el *reset* manual en el caso de que la velocidad de crecimiento de la señal de alimentación sea < 0,05 v/ms o si se requiere el acceso al reset de forma manual.

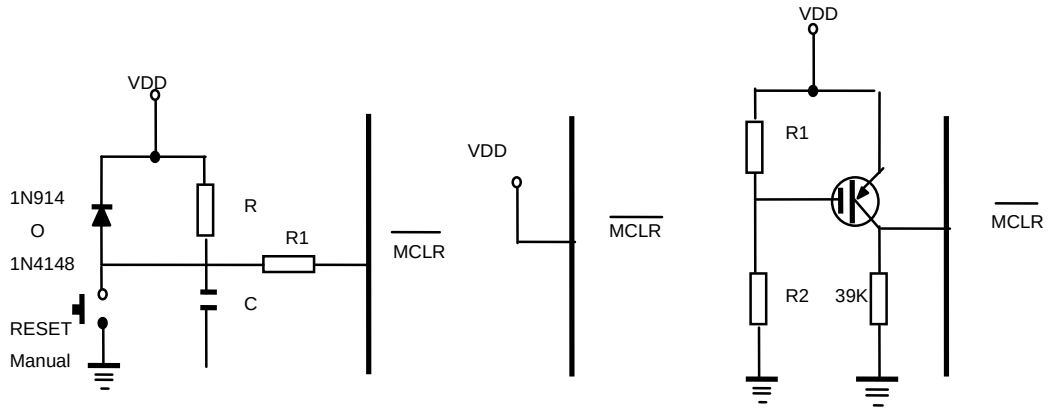


Fig. 2.2: *Circuito de reset manual, automático al conectarse a la tensión de alimentación y en caso de descenso de la tensión de alimentación.*

2.2.4 RA0 - RA3

Son los pines de entrada -salida del 0 al 3 del puerto A.

2.2.5 RA4

Pin 4 de entrada-salida del puerto A común con la entrada de reloj externo del temporizador T0.

2.2.6 RB0/INT

Entrada-salida o del puerto B común con la entrada de interrupciones externas.

2.2.7 RB1 - RB7

Entada-salida de la uno a la siete del puerto B.

a) RB4 - RB7

Soportan además la función de interrupción por cambio de estado.

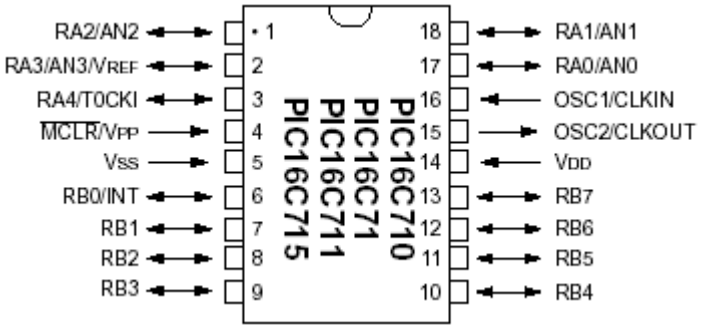


Fig. 2.3: *Patillaje del encapsulado del PIC 16C711*

2.3 ARQUITECTURA INTERNA

La memoria de programa esta organizada en palabras de 14 *bits*, el tamaño de la pila es de 8 niveles. Los registros están organizados siempre del mismo modo, estando en las direcciones bajas los registros especiales y de recursos y en las altas los de propósito general. El PC contiene 13 *bits* por lo que no se pueden direccionar 8 *Kbytes* de memoria, como no hay mas que 1 *Kbyte*, el hecho de direccionar mas allá de este valor hace que se vuelva al principio, como si los bits de mayor peso se ignoraran.

2.4 Registros Internos

Este gráfico representa la cartografía de la memoria de estos registros, obsérvese la utilización de dos páginas de memoria, la primera o página cero, contiene los registros fundamentales de 00 a 1F.
La segunda o página una, contiene registros menos importantes o asociados a registros de la página cero.

Tabla 2.1: *Organización de los registros internos.*

00	Direccionamiento indirecto	Direccionamiento indirecto	80
01	TMR0	OPTION	81
02	PCL	PCL	82
03	STATUS	STATUS	83
04	SFR	FSR	84
05	PORT A	TRIS A	85
06	PORT B	TRIS B	86
07		PCON	87
08	ADCON0	ADCON1	88
09	ADRES	ADRES	89
0A	PCLATCH	PCLATCH	8A
0B	INTCON	INTCON	8B
OC 2F	REGISTRO DE PROPOSITO GENERAL	IDEM PAGINA CERO	8C AF
30 7F			B0 FF
	PÁGINA 0	PÁGINA 1	

Así, el registro TRIS A asociado al PORT A se encuentra en la misma dirección dentro de la página.
Determinados registros figuran en las dos páginas, por lo que son accesibles de la misma forma y sin restricción, para facilitar de este modo la programación del circuito.

2.4.1 Registro INDF (Dirección 00) o registro de indirección

No tiene existencia física, por lo que no se podrá acceder a él. En realidad este registro sirve únicamente para especificar la utilización del direccionamiento indirecto, por ejemplo de la forma siguiente:

ADDWF INDF.W Significa que se va a añadir, al contenido del registro W, el contenido del registro apuntado por el registro FSR de dirección 04 (ver tabla 2.1). Se realiza entonces el Direccionamiento indirecto con respecto al contenido de FSR. Sirviéndose de INDF únicamente como modo de notación.

2.4.2 Registro PCL (Dirección 02)

Es el PC o contador de programa, o mejor dicho los 8 *bits* de menor peso del PC, puesto que el PC debe tener un tamaño de 13 *bits*, sus *bits* de mayor peso se llevan al registro PCLATCH, situado en las posiciones 0A y 8A.

Si el PC es destino de una instrucción, el contenido de PCLATCH se tiene en cuenta automáticamente.

Para las instrucciones GOTO y CALL, tiene lugar la misma operación, teniendo en cuenta el hecho de que el PC está codificado con 11 *bits* en la propia instrucción.
Los ocho registros de pila no aparecen en la gráfica anterior por no estar situados en el mismo espacio de memoria que los demás. Son registros de trece *bits* capaces de contener íntegramente al PC. Su utilización es automática, ya que el PC se introduce en la pila durante la ejecución de una instrucción CALL o de una interrupción, y se extrae de la pila durante la ejecución del retorno correspondiente. Estos registros de pila deben considerarse como un buffer de memoria circular, lo que significa que, si se introduce más de 8 valores del PC, el noveno valor tomará la posición del primero, y así sucesivamente.
Obsérvese que ningún *bit* de registro indica que la pila está llena, por lo que debe tener cuidado de que no se desborde, excepto si quiere realizar operaciones de retorno especiales.

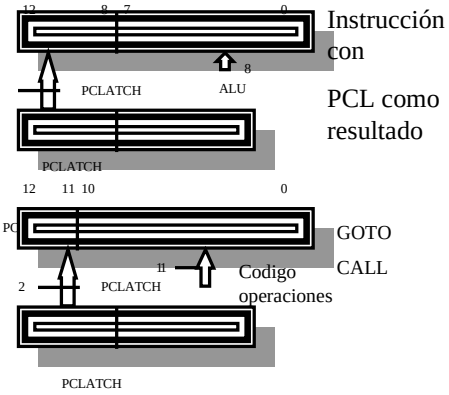
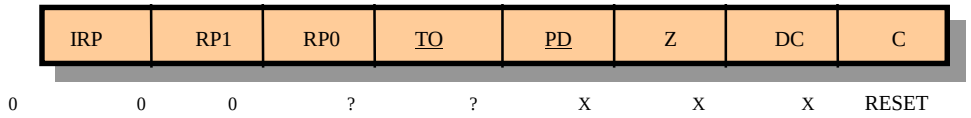


Figura 2.4: *Principio de determinación de los bits de mayor peso del PC por PCLATCH*

2.4.3 REGISTRO STATUS (Dirección 03) o Registro de Estado

Este registro contiene un cierto número de *bits* de estado de la unidad central, pero también los *bits* de selección de páginas, que ahora se denominan RP1 y RP0. Se pueden leer o escribir como cualquier otro registro, dando por supuesto que determinados *bits* de estado no se modificarán después de un intento de escritura. Cada *bit* de este registro tiene un significado particular que es el que sigue.



- **BIT 0** o *bit* de C (*Carry*). *Bit* de acarreo para las operaciones de suma y resta. Se pone a uno mediante las instrucciones ADDWF y SUBWF, si se genera un acarreo en el *bit* de mayor peso.
  Este *bit* también lo utilizan las instrucciones de rotación.
- **BIT 1** o *bit* DC (*Digit Carry*). Este es un *bit* de acarreo de dígito, para por ejemplo la aritmética en BCD. Se pone a uno con las instrucciones ADDWF y SUBWF, si se genera un acarreo del *bit* 3 al grupo de cuatro *bits* superior.
- **BIT 2** o *bit* Z (*zero*). Este *bit* se pone a uno si el resultado de la operación aritmética o lógica ejecutada es nulo.
- **BIT 3** o *bit* PD (*Power Down*). Este *bit* se pone a uno durante la conexión a la alimentación del circuito, o durante la ejecución de una instrucción CLRWDT relativa al temporizador *watchdog*. Se pone a cero mediante una instrucción SLEEP.
- **BIT 4** o *bit* TO (*Time Out*). Este *bit* se pone a uno durante una conexión a la alimentación o durante la ejecución de una instrucción CLRWDT o SLEEP. Se pone a cero cuando el temporizador *Watchdog* se desborda.
- **BIT 5 Y 6** o *bits* RP0 y RP1. Estos *bits* sirven para seleccionar las páginas de memoria de programa. La tabla 1.5 precisa su modo de utilización, que es perfectamente lógico sabiendo que cada página contiene 128 *bytes.*
- **BIT 7** o *bit* IRP. Este *bit* está previsto para un futuro direccionamiento de paginado indirecto, pero no se utiliza en el PIC 16C711. Tan solo se usa para compatibilidad con las futuras versiones, por lo que se debe poner a cero.

Tabla 2.2: *Programación de los bits de selección de página del STATUS*

RP1	RP0	PAGINA	DIRECCION
0	0	0	00 a 7F
0	1	1	80 a FF
1	0	2	100 a 17F
1	1	3	180 a 1FF

2.4.4 REGISTRO FSR (Dirección 04) o registro de selección de registro.

El contenido del SFR se utiliza para el direccionamiento indirecto. Este registro contiene 8 *bits*; no obstante, es preciso saber que en direccionamiento indirecto, puede construirse una dirección de 9 *bits* utilizando el contenido de este registro y el *bit* IRP del registro de estado. Esta función no se utiliza en el 16C711, sino que está prevista para extensiones futuras, por ello el que se no aconseje el uso del *bit* IRP.

2.4.5 REGISTRO PCLATCH (Dirección 0A y 8A)

Se cita este registro sólo como recordatorio, ya que lo vimos en el PCL.

2.4.6 REGISTRO INTCON (Dirección 0B y 8B)

Este registro sirve para el control global de las interrupciones y para indicar la procedencia de algunas de ellas, gracias a los *bits* de estado. Se dispone de cuatro potenciales recursos de interrupción.

- Una fuente externa a través del pin RB0/INT.
- El desbordamiento del temporizador 0.
- Un cambio de estado en los pines RB4 a RB7.
- Programación de la EEPROM de datos.

Cada *bit* del registro INTCON tiene un significado concreto que es el que sigue:

- **BIT 0** o *bit* RBIF (RB *Interrupt Flag*). Si se pone a 1, este *bit* indica un cambio de estado en una de las líneas de RB4 a RB7 del puerto B.
- **BIT 1** o *bit* INTF (*Interrupt Flag*). Si se pone a 1, este *bit* indica una interrupción provocada por la línea RB0/INT del puerto B.
- **BIT 2** o *bit* T0IF (*Timer* 0 *Interrupt Flag*). Si se pone a 1, este *bit* indica un desbordamiento del temporizador 0.
- **BIT 3** o *bit* RBIE (RB *Interrupt Enable*). Si se pone a uno, este *bit* autoriza las interrupciones provocadas por un cambio de estado de las líneas RB4 a RB7 del puerto B.
- **BIT 4** o *bit* INTE (*Interrupt Enable*). Si se pone a 1, este *bit* autoriza las interrupciones provocadas por la línea RB0/INT del puerto B.
- **BIT 5** o *bit* T0IE (*Time Out Interrupt Enable*). Si está a uno, este *bit* autoriza las interrupciones debidas al desbordamiento del temporizador 0.
- **BIT 6** o *bit* ADIE (*A/D conversion Interrupt Enable*). Si está a uno, este *bit* autoriza las interrupciones que proceden de la memoria EEPROM de datos.

- **BIT 7** o *bit* GIE (*Global Interrupt Enable*). Si se pone a uno, este *bit* autoriza todas las interrupciones que estén enmascaradas mediante sus *bits* individuales de activación, a cero las inhibe.

Cada activador individual debe ponerse a cero por *software*.
Solamente hay un vector indicador de interrupción (dirección 0004), por lo que se debe comprobar estos *bits* en el programa de interrupción para saber cual es la fuente de la misma.
Cuando llega una interrupción, el PIC pone el *bit* GIE a cero, de forma que no se perturbe el tratamiento de la interrupción en curso, debido a otras interrupciones eventuales. Este *bit* se pone automáticamente a uno al terminar el programa de interrupción, con la ejecución de la instrucción RETFIE. Los indicadores de interrupciones correspondientes permanecen funcionales incluso cuando no se han autorizado.

2.4.7 PUERTOS PARALELOS

Se dispone de dos puertos paralelos A y B. Las líneas de estos puertos se pueden programar individualmente como entradas o como salidas, y se utilizan casi de la misma forma.
Teniendo en cuenta que el encapsulado se propone con 18 pines, determinadas líneas de estos puertos se comparten con otros recursos internos.

a) Puerto A: Dispone de un ancho de 5 *bits*. El esquema interno de las líneas RA0 a RA3 se muestra en la Fig. 2.5. La salida está provista de un *buffer* CMOS, entrada y salida pasan por un *latch*.

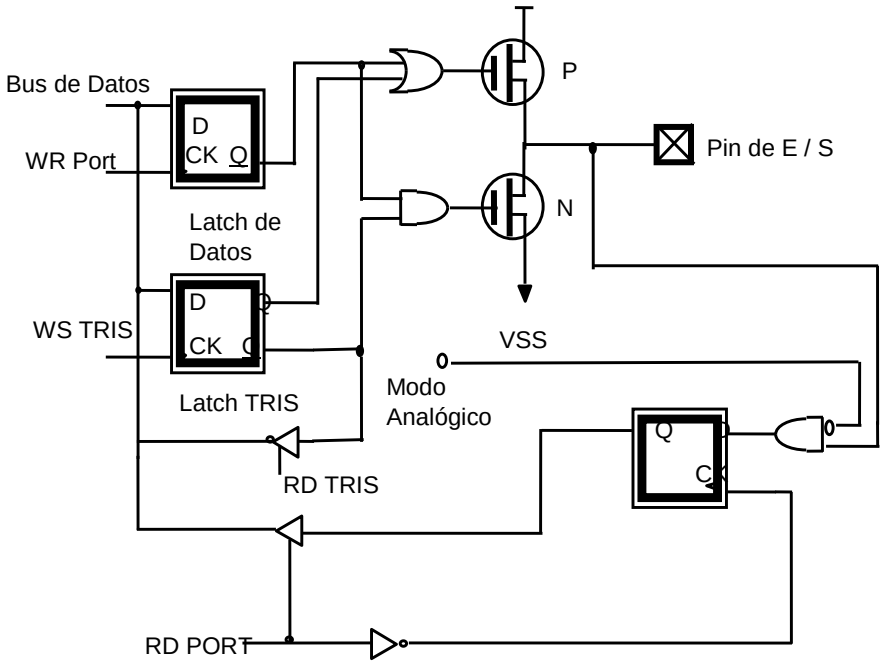


Fig. 2.5: *Esquema de las líneas RA0 a RA3 de la puerta A*

La línea RA4 adopta una estructura diferente (Fig. 2.6), su salida es de tipo dren abierto, y la entrada está provista de un *Trigger Schmitt*. Es común para la entrada externa del temporizador 0.

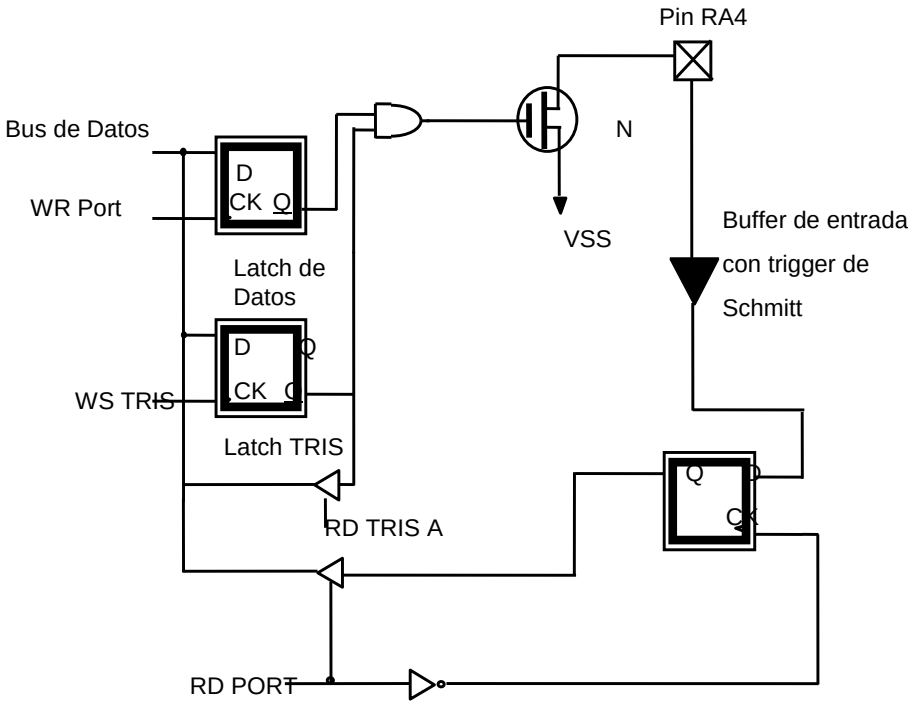


Fig. 2.6: *Esquema de la línea RA4 del puerto A*

El sentido de trabajo de todas las líneas de este puerto se controla mediante el registro TRISA, en el que un *bit* a cero activa la línea correspondiente como salida, y un *bit* a 1 como entrada, Evidentemente, después de un *reset*, todos los *bits* del registro TRISA quedan a uno.

b) Puerto B: El puerto B es un puerto bidireccional de 8 *bits* completo, en el que sólo una línea se comparte con otro recurso interno. Las líneas RB0 a RB3 adoptan la estructura interna indicada en la Fig. 2.7, mientras que las líneas RB4 a RB7 la de la Fig. 2.8. La razón de ser de esta diferencia, radica en el hecho de que es posible programar la generación de una interrupción durante un cambio de estado de cualquiera de las líneas RB4 a RB7.

Todas las líneas del puerto B disponen de una resistencia de *pull-up* de alto valor, conectada a la alimentación. Esta resistencia se puede activar o no gracias al *bit* RBPU del registro OPTION. Dicha activación afecta a todas las líneas del puerto B y se desactiva tras un *reset* al igual que las líneas configuradas como salida.

El sentido de trabajo de cada una de las líneas de este puerto es controlado por el registro TRISB, en el que un *bit* a cero hace que la línea correspondiente se active como salida, y un *bit* a 1 como entrada, tras un reset todos los *bits* se ponen a 1.

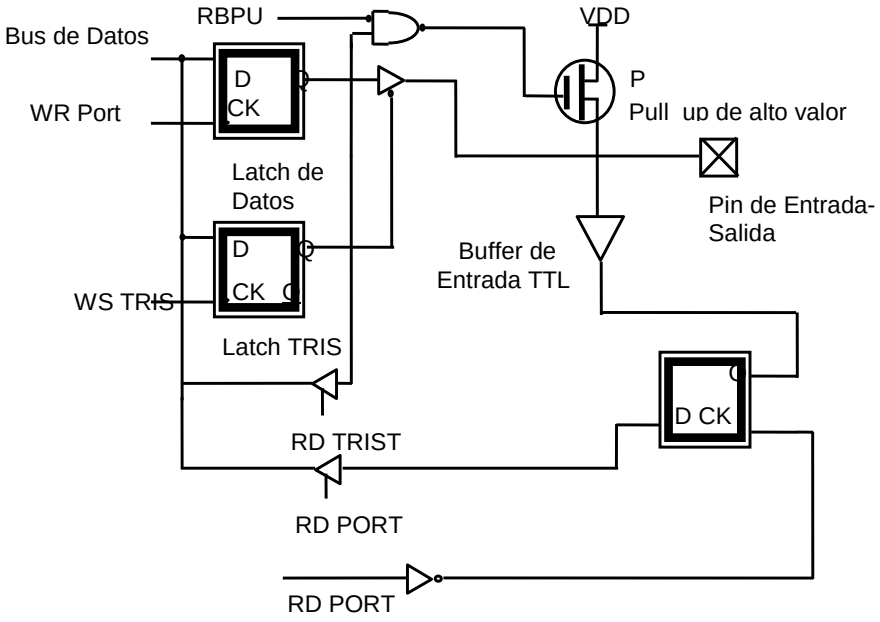


Fig. 2.7: *Esquema de las líneas RB0 a RB3 del puerto B*

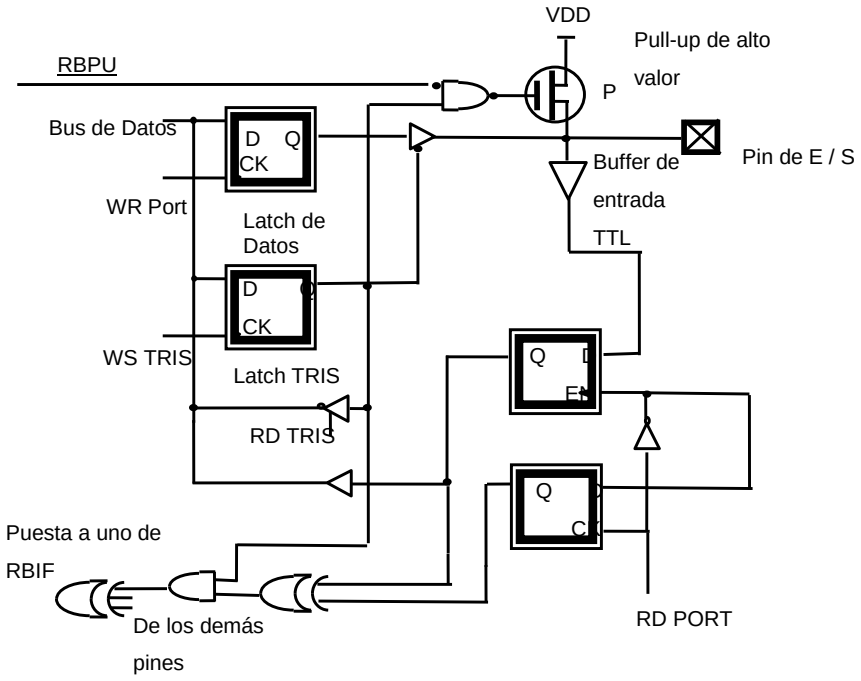


Fig. 2.8: *Esquema de la línea RB4 a RB 7 del puerto B*

2.4.8 Temporizador T0

El PIC 16C711 no contiene mas que un temporizador analógico. Genera una interrupción cada vez que el temporizador se desborda de FF a 00. El modo de funcionamiento y programación se realiza a través del registro OPTION.

Puede recibir su señal directamente, o a través de un predivisor, siendo hecha la selección mediante el *bit* PSA. Si se selecciona el predivisor, su tasa de división se programa a través de tres *bits*: PS0, PS1, PS2 (Fig. 2.9).
En todos los casos, la señal utilizada puede proceder de una cualquiera de las dos fuentes posibles, dependiendo del estado del *bit* de selección RTS. o procede del oscilador de reloj después de la división por cuatro, por lo que está a la frecuencia del reloj de instrucción, o del pin TOCKI. La puerta OR EXCLUSIVO, controlada por el *bit* RTE, permite seleccionar si el incremento del registro TMRO se producirá sobre el flanco de subida o bajada (Fig. 2.9).

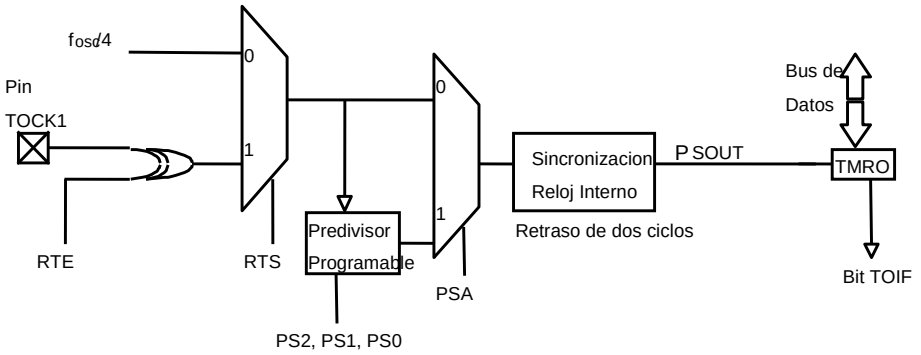


Fig. 2.9: *Esquema del temporizador T0*

Los *bits* RTE, RTS, PSA y PS0 a PS2 que se emplean para configurar este temporizador están contenidos en el registro OPTION. Cada *bit* de este registro tiene un significado especial que es el que sigue:

a) REG. OPTION

RBPU	INTEDG	RTS	RTE	PSA	PS2	PS1	PS0	OPTION
1	1	1	1	1	1	1	1	Reset

- **BITS 0, 1, 2.** *Bits* PS0, PS1, PS2. Estos *bits* sirven para definir la tasa de división del predivisor, como muestra la tabla, esta tasa difiere dependiendo de que el predivisor este asignado al temporizador o al *watchdog*.

Tabla 2.3: *Programación de los bits de selección del predivisor en el registro OPTION*

PS2	PS1	PS0	Temporizador 0	Watchdog
0	0	0	2	1
0	0	1	4	2
0	1	0	8	4
0	1	1	16	8
1	0	0	32	16
1	0	1	64	32
1	1	0	128	64
1	1	1	256	128

- **BIT 3** o *bit* PSA (*Prescaler Assignment*). Si este *bit* está a cero, el predivisor está asignado al temporizador 0. Si está a uno estará asignado al *watchdog*.
- **BIT 4** o *bit* RTE (*Timer Out signal Edge*). Si este *bit* está a cero, el contenido del registro TMR0 se incrementará en un flanco de subida de la señal aplicada al pin RA4/ToCK1; si está a uno, se incrementará con el flanco de bajada.
- **BIT 5** o *bit* RTS (*Timer 0ut signal source*). Si este *bit* está a cero, el temporizador utilizará el reloj interno. Si está a uno, utilizará la señal aplicada al pin RA4/T0CK1.
- **BIT 6** o *bit* INTEDG (*Interrupt edge*). Este *bit* define el sentido del flanco que provocará la interrupción a través del pin externo INT. Un uno activa el flanco de subida y un cero un flanco de bajada.
- **BIT 7** o *bit* RBPU (RB *Pull Up enable*). Si está a cero, este *bit* activa las resistencias de *pull up* a nivel alto, previstas en la entrada del puerto B.

(Estos últimos dos *bits* no se usan en el temporizador)

El registro TMRO se incrementa en una unidad con cada impulso de reloj seleccionado mediante el registro OPTION. Cada vez que llega al valor FF, vuelve a 00 generando una interrupción, si se ha autorizado, y continúa su ciclo indefinidamente.

El registro TMR0 se puede leer o escribir directamente con cualquier instrucción, con el fin de conocer su posición actual o para inicializarlo en un estado determinado. Es importante saber que después de cualquier escritura en este registro, es necesario un retardo de dos ciclos de instrucción para que se retome la incrementación. Este retraso es independiente de la fuente de reloj usada. Las instrucciones concernidas son MOVF TMRO o CLRF TMR0.
Para comprobar el paso por cero sin inferir en el desarrollo regular del recuento, es aconsejable utilizar, por ejemplo, una instrucción MOVF TMR0,W, que no hace mas que una lectura. El reloj interno deja de funcionar en el modo SLEEP, por lo que no se puede contar con sus interrupciones en este modo ni por consiguiente, que salga de este modo de funcionamiento por medio de dicha interrupción.
Obsérvese que todas las instrucciones que escriben en el TMR0 ponen a cero al predivisor, cuando éste está asignado al temporizador.

2.4.9 Temporizador *Watchdog*

Está provisto de su propio oscilador interno autónomo con célula RC interna, por lo que no necesita ningún componente externo y continúa funcionando incluso cuando el reloj del PIC se para durante, por ejemplo, la instrucción SLEEP.
Este temporizador cuenta de forma permanente, y cuando se desborda, es decir cuando llega a FF, hace que se genere un *reset* del microcontrolador. Si el PIC está en modo *sleep* cuando se produce este desbordamiento, tiene por efecto que se salga de este modo.
Si no se desea utilizar el *watchdog*, es posible desactivarlo escribiendo un 0 en un *bit* específico del circuito durante su programación. Este *bit* no es direccionable, pero todos los programadores saben acceder a él.
El tiempo típico de desbordamiento de este temporizador es de 18 ms, pero puede variar algo con la tensión de alimentación y la temperatura. Si no es muy largo, el predivisor del que hemos hablado puede asignarse al *watchdog* gracias al *bit* PSA del registro OPTION. En estas

condiciones, las tasas de predivisión definidas por este registro varían, pero teniendo en cuenta el máximo permitido de 128 ms, es posible obtener tiempos de desbordamiento de hasta 2.5 s.
Las instrucciones CLRWDT y SLEEP ponen a cero al *watchdog* y a uno el *bit* T0 del registro de estado. Esto evita que se generen *resets* indeseados, como por ejemplo cuando se está en modo *sleep*. Estas instrucciones también ponen a cero el predivisor, si este está asignado al *watchdog*.

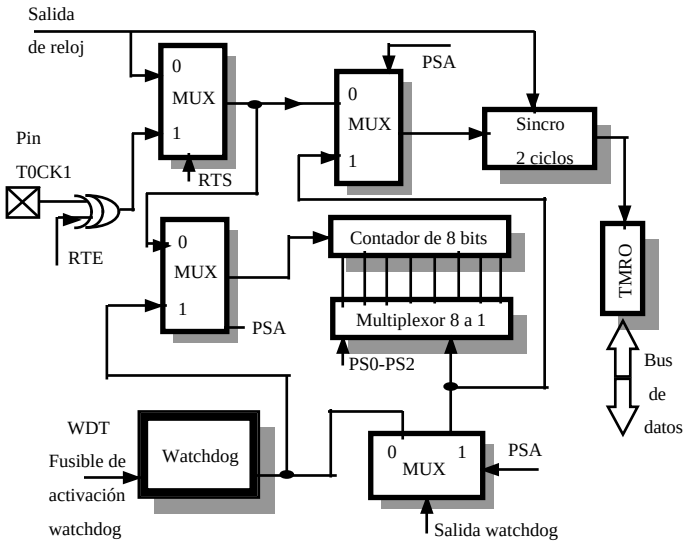


Fig. 2.10: *Temporizador Watchdog.*

2.4.10 Conversión analógica digital

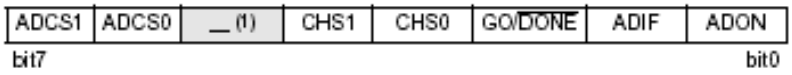
El módulo conversor analógico-a-digital (A/D) tiene cuatro entradas analógicas.
El A/D permite la conversión de una señal de entrada analógico a un correspondiente número digital de 8 *bits* (refiérase a la Aplicación Note AN546 para el uso de Conversor de A/D en Anexo). La salida de el ***Sample and Hold*** es la entrada en el conversor, el cual genera el resultado vía aproximación sucesiva (después veremos este método de conversión con más detalle).
El voltaje analógico de referencia es seleccionable por *software*. Puede ser el voltaje de alimentación positivo del dispositivo (VDD) o el nivel de voltaje en el pin de RA3/AN3/VREF.
El conversor de A/D tiene una única forma de estar disponible para operar mientras el dispositivo está en el modo SLEEP. Para operar en el modo SLEEP, el reloj del conversor A/D debe derivarse del oscilador de RC interior del A/D.

El módulo de A/D tiene tres registros. Estos registros son:

•El A/D Resultado Registro (ADRES)
•El A/D Mando Registro 0 (ADCON0)
•El A/D Mando Registro 1 (ADCON1)

El registro ADCON0, mostrado en la figura, controla el funcionamiento del módulo A/D.
El registro ADCON1, mostrado en la Figura configura las funciones de los pines del puerto. Los pines del puerto pueden configurarse como entradas analógicas (RA3 también puede ser una referencia de voltaje) o como I/O digital.

a) Registro ADCON0.



BIT 7-6: ADCS1:ADCS0:*Bits* de selección del Reloj de conversión A/D.

00 = FOSC/2
01 = FOSC/8
10 = FOSC/32
11 = FRC (el reloj derivado de una oscilación de RC)

BIT 5: no implementado: Lea como '0 '.

BIT 4-3: CHS1:CHS0: Selección de *bits* para canal analógico

00 = canal 0, (RA0/AN0)
01 = canal 1, (RA1/AN1)
10 = canal 2, (RA2/AN2)
11 = canal 3, (RA3/AN3)

BIT 2: GO/DONE: el *bit* de estado de la Conversión A/D.
Si ADON = 1: Si ADON = 1:

1 = la conversión de A/D está en marcha (poniendo este momento empieza la conversión de A/D).
0 = la conversión de A/D no está en marcha (Este *bit* es limpiado automáticamente por el *hardware* cuando la conversión A/D está completa).

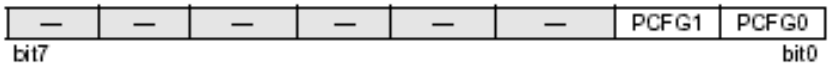
BIT 1: ADIF: El *bit* de Bandera de Interrupción de la Conversión Completa A/D.

1 = la conversión está completa (debe limpiarse en el *software*).
0 = la conversión no está completa.

BIT 0: ADON: A/D *bit* de operación.
1 = el módulo conversor A/D está operando.
0 = el módulo conversor A/D está apagado y no consume ninguna corriente.

Nota 1: *Bit* 5 de ADCON0 es un *bit* de escritura y lectura de Propósito General.

b) Registro ADCON1.



BIT 7-2: no implementado: Se lee como '0'.
BIT 1-0: PCFG1:PCFG0: A/D *Bits* de control para la configuración del puerto de conversión.

Tabla 2.4: *Programación de los bits de selección del canal de conversión A/D.*

PCFG1 : PCFG0	RA1 & RA0	RA2	RA3	VREF
00	A	A	A	VDD
01	A	A	VREF	RA3
10	A	D	D	VDD
11	D	D	D	VDD

A = Entrada Analógica.
D = Digital I/O (Entrada o salida).

El registro ADRES contiene el resultado de la conversión A/D.
Cuando la conversión A/D está completa, el resultado está cargado en el registro ADRES, el *bit* 2GO/DONE del registro ADCON0 se limpia, y el *bit* de interrupción de flanco ADIF es puesto a 1. El diagrama del bloque del módulo A/D es mostrado en la Figura siguiente.
Después de que el módulo de A/D se ha configurado como se desea, se debe seleccionar el canal de adquisición de datos analógicos antes que la conversión empiece.
Los canales de la entrada analógicos deben tener sus *bits* correspondientes de configuración (TRIS), seleccionados como entrada.
Para determinar el tiempo de adquisición, vea Sección 4.10.7. Después de que este tiempo de adquisición ha pasado, la conversión A/D puede empezar. Para realizar una buena conversión analógica deben seguirse los siguientes pasos:

- Configurar el módulo de A/D:

  •Configure los pines analógicos / el voltaje de referencia y digital I/O (ADCON1).
  •Selecionar el canal A/D de entrada (ADCON0).
  •Selecionar el reloj de conversión A/D (ADCON0).
  •Activar el módulo A/D (ADCON0).

- Configurar interrupción A/D (si se desea):

  •Limpiar el *bit* ADIF.
  •Poner a 1 el *bit* ADIE.
  •Poner a 1 el *bit* GIE.

- Espera el tiempo de adquisición requerido.
- Iniciar conversión:

  •Poner a 1 el *bit* GO/DONE (ADCON0)

- Espera que la conversión A/D se complete, por cualquiera de los métodos propuestos a continuación:

  •Probando que el *bit* GO/DONE esté limpio (en 0)
  •Esperando que suceda la interrupción A/D, si es que fue programada.

- Leer el resultado de la conversión A/D en el registro (ADRES), limpiar el *bit* ADIF si se requiere.

- Para la próxima conversión, ir al paso 1 o 2 como sea requerido. El tiempo de conversión A/D por *bit* es definido como TAD. Una espera mínima de 2TAD es requerido antes que la próxima adquisición empiece.

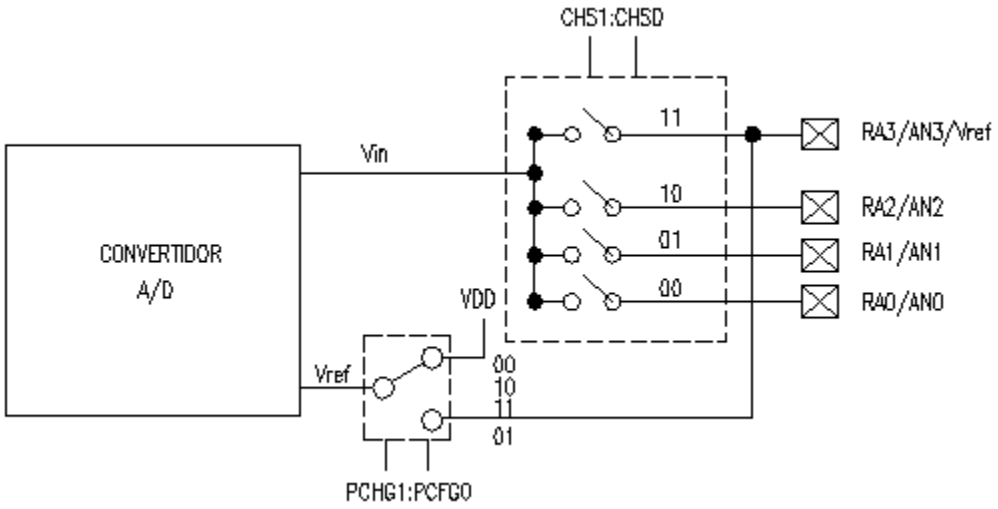


Fig. 2.11: *Canales de Conversión.*

c) Requisitos para la Adquisición A/D: Para que el conversor A/D, pueda reunir su exactitud especificada, la carga del condensador (CHOLD) debe estar totalmente al máximo y así obtener el voltaje que precisamente hay en el canal de entrada analógica.
El modelo de la entrada analógica se muestra en la Figura siguiente. La Impedancia de la fuente (RS) y la impedancia del interruptor de prueba interior (RSS) afectan directamente al tiempo requerido para cargar el condensador CHOLD. La impedancia del interruptor de prueba (RSS) varía si se supera el voltaje del dispositivo (VDD).
La impedancia de la fuente afecta la compensación del voltaje a la entrada analógica (debido a la fuga del pin de adquisición).
La impedancia máxima recomendada para las fuentes analógicas son 10 $k\Omega$. Después de que el canal de la entrada analógica es seleccionado se debe proceder a la adquisición antes de que la conversión empiece.
Para calcular el tiempo de adquisición mínimo, puede usarse una ecuación. Esta ecuación calcula el tiempo de adquisición siendo 1/2 el error de LSb (512 pasos para el A/D).
El 1/2 error de LSb es el error máximo permitido para que el A/D tenga su exactitud especificada.

Ecuación del Mínimo tiempo de Carga:

$$\text{Vhold} = (\text{Vref} - (\frac{\text{Vref}}{512})) \times (1 - e\Lambda - (\frac{\text{Tcap}}{\text{Chold}(\text{Ric} + \text{Rss} + \text{Rs})})) \quad 2.1$$

Dado: Vhold = (Vref/512), para 1/2 resolución de LSb
La ecuación anterior reduce a:

$$\text{Tcap} = -(512\text{pF}) \times (1\text{k}\Omega + \text{Rss} + \text{Rs}) \times \ln(1/511) \quad 2.2$$

El ejemplo 1 muestra el cálculo del mínimo tiempo de adquisición requerido TACQ.
Este cálculo es basado en las asunciones del sistema siguiente.

$Chold = 512\,pF$
$Rs = 10k\Omega$
1/2 error de LSb
$Vdd = 5V$
$Rss = 7k\Omega$
Temperatura máxima del sistema de aplicación = 50°C
$Vhold = 0 @\ T = 0$

EJEMPLO 1: Cálculo del mínimo tiempo para la adquisición [12].

TACQ = Tiempo de establecimiento del Amplificador + tiempo de carga del condensador + el Coeficiente de temperatura.

$$\text{Tacq} = 5\text{us} + \text{Tcap} + ((\text{Temp} - 25°\text{C}) \times (\frac{0.05\text{us}}{°\text{C}})) \quad 2.3$$

$$\text{Tcap} = -(\text{Chold}) \times (\text{Ric} + \text{Rss} + \text{Rs}) \times \ln(1/511) \quad 2.4$$

$$\text{Tcap} = -(512\text{pF}) \times (1\text{k}\Omega + 7\text{k}\Omega + 10\text{k}\Omega 0 \times \ln(0.002) \quad 2.5$$

$$-512\text{pF} \times 18\text{k}\Omega \times \ln(0.002)$$

$$-0.921\text{us} \times (-6.2364)$$

$$\text{Tcap} = 5.747\text{us}$$

$$\text{Tacq} = 5\text{us} + 5.747\text{us} + ((50°\text{C} - 25°\text{C}) \times (\frac{0.05\text{us}}{°\text{C}}))$$

$$10.747\text{us} + 1.25\text{us}$$

$$\text{Tacq} = 11.997\text{us}$$

Nota 1: El voltaje de la referencia (VREF) no tiene efecto en la ecuación desde que se cancela
Nota 2: La carga del condensador (CHOLD) no es descargado después de cada conversión.
Nota 3: La impedancia máxima recomendada para las fuentes analógicas es 10 $k\Omega$. Esto es requerido para encontrar la especificación del pin.
Nota 4: Después de que una conversión ha completado, un 2.TAD retraso que debe completar antes que la adquisición pueda empezar de nuevo.
Durante este tiempo el condensador no esta conectado con el canal elegido de entrada A/D.

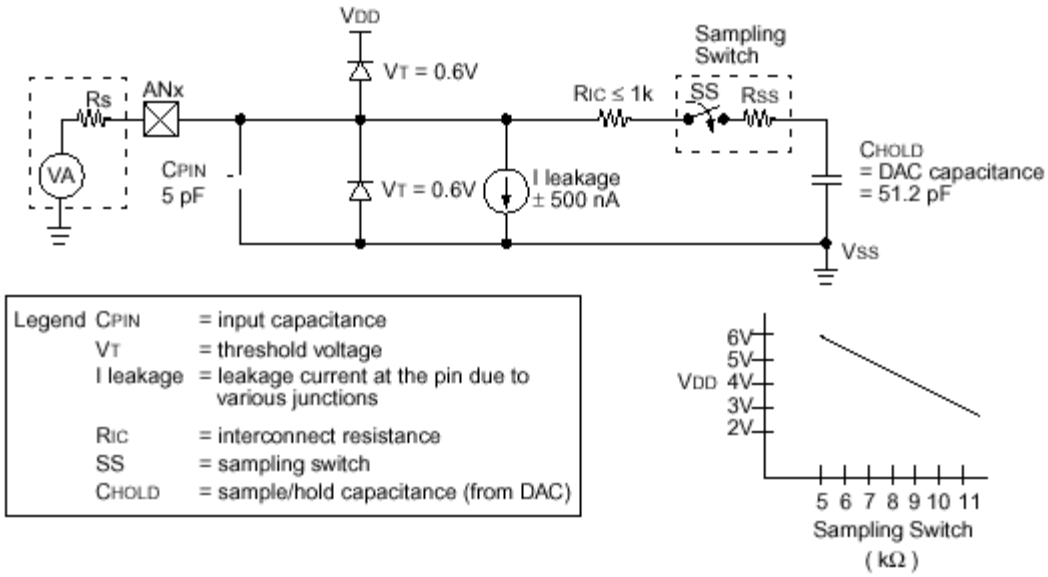


Fig. 2.12 *Circuito interno del convertidor A/D*

d) Selección del Reloj de Conversión A/D: El tiempo por *bit* de conversión A/D, se define como TAD. La conversión A/D requiere 9.5TAD por la conversión de 8-*bits*.
La fuente del reloj de conversión A/D es elegible por *software*.
Las cuatro posibles opciones para TAD son:

•2 T_{OSC}
•8 T_{OSC}
•32 T_{OSC}
•El oscilador RC interno

Para las conversiones A/D correctas, el reloj de conversión A/D debe seleccionarse para asegurar un tiempo de TAD mínimo de:

2.0 ms para el PIC16C71
1.6 ms para todos los otros dispositivos PIC16C71X

La tabla siguiente muestra el TAD resultante tiempos derivados del dispositivo que opera las frecuencias y la fuente de reloj seleccionada.

e) Configuración de los pines de Puerto de Analógico: Los registros ADCON1 y TRISA controlan la operación del puerto A/D. Los pines del puerto que se desean siempre y cuando las entradas analógicas tengan su TRIS correspondiente como entrada.
Si el bit de TRIS se aclara (salida), la salida digital (VOH o VOL) será convertido.
El funcionamiento de A/D es independiente del estado de los *bits* de CHS2:CHS0 y los bits de TRIS.

Nota 1: Al leer el registro del puerto, todos los pines, configurado como canales de entradas analógicos serán limpiados (un nivel bajo). Los pines configurados como entradas digitales, convertirán una entrada analógica la entrada. Los niveles analógicos en canales configurados para entradas digitales, no afectarán la exactitud de la conversión.

Nota 2: Los niveles analógicos en cualquier pin que está definido como entrada digital (incluso los pines AN7:AN0), pueden causar que el buffer de la entrada consuma más corriente sobrepasando la especificación de los dispositivos.

Tabla 2.5: *Programación de los bits de selección del tiempo de conversión.*

AD Clock Source (TAD)		Device Frequency			
Operation	ADCS1:ADCS0	20 MHz	5 MHz	1.25 MHz	333.33 kHz
2TOSC	00	100 ns(2)	400 ns(2)	1.6 µs	6 µs
8TOSC	01	400 ns(2)	1.6 µs	6.4 µs	24 µs(3)
32TOSC	10	1.6 µs	6.4 µs	25.6 µs(3)	96 µs(3)
RC(5)	11	2 - 6 µs(1,4)	2 - 6 µs(1,4)	2 - 6 µs(1,4)	2 - 6 µs(1)

Leyenda: Las celdas sombreadas están fuera del rango recomendado.

Notas

1:.La fuente de RC tiene un tiempo TAD típico de 4 ms.
2:.Estos valores violan el mínimo tiempo requerido TAD.
3:.Para tiempos de conversión más rápidos, se recomienda la selección de otra fuente del reloj.
4:.Cuando la frecuencia del dispositivo es mayor a 1MHz, se recomienda la fuente de reloj RC sólo para funcionamiento en modo SLEEP.
5:.Para los dispositivos de voltajes extendidos (LC), por favor refiérase a la sección de las Características técnicas Eléctricas en la hoja de datos correspondiente al PIC 16C711.

Conversiones A/D: El ejemplo 2 siguiente muestras cómo realizar una conversión A/D.
Los pines de RA se configuran como entradas analógicas.
La referencia analógica (VREF) es la alimentación del dispositivo (VDD).
La interrupción de A/D se habilita, y el reloj de conversión A/D es FRC.
La conversión se realiza en el pin RA0 (canal 0).

Limpiar el *bit* GO/DONE durante una conversión abortará la conversión actual. El registro ADRES no será actualizado con la conversión A/D parcialmente completada.

Es decir, el registro ADRES continuará conteniendo el valor de la última conversión completa (o el último valor escrito al registro de ADRES).
Después de que la conversión A/D es abortada, un tiempo 2TAD de espera es requerido antes de que la próxima adquisición se empiece.
Después de esta espera, una adquisición empieza automáticamente en el canal seleccionado.

Nota: El *bit* GO/DONE no debe estar en 1cuando se pone en operación el módulo A/D.

Ejemplo 2:

```
BSF     STATUS, RP0     ; Selección del banco 1.
CLRF    ADCON1          ; Configuración de las entradas A/D
BCF     STATUS, RP0     ; Selección del banco 0.
MOVLW   0xC1            ; Reloj RC, Empieza A/D , canal 0 se selecciona
MOVWF   ADCON0          ;
BSF     INTCON, ADIE    ; Habilita la Interrupción de A/D
BSF     INTCON, GIE     ; Habilita todas las interrupciones

---------------------------------------------------------------------------
; Asegura el tiempo requerido de muestreo para el canal seleccionado como entrada.
; Entonces la conversión puede empezarse.
---------------------------------------------------------------------------

BSF     ADCON0, GO      ; Empieza la Conversión A/D. El bit ADIF se
                        pondrá en 1 y el bit GO/DONE se limpia en la
                        realización de la Conversión A/D.
```

f) Conversión más rápida (Baja resolución): No todas las aplicaciones requieren un resultado con los 8-*bits* de resolución, pero pueden requerir un tiempo de conversión más rápido en cambio.
El módulo A/D les permite a los usuarios obtener alta velocidad de la conversión a costa de bajar la resolución.
Sin tener en cuenta la resolución requerida, el tiempo de adquisición es el mismo. Para acelerar la conversión, la fuente del reloj del módulo A/D, puede cambiarse para que el tiempo de TAD viole el tiempo mínimo especificó (vea las especificaciones eléctricas aplicables). Una vez el tiempo de TAD viola el mínimo tiempo especificado, todos los bits de la conversión A/D siguientes resultan los no válidos (vea Tiempo de la conversión A/D en la sección de las especificaciones eléctricas).
Las fuentes del reloj sólo pueden ser cambiadas entre las tres versiones del oscilador (no puede ser cambiada a RC).
La ecuación para determinar el tiempo antes de que el oscilador pueda cambiarse es como sigue:

$$\text{Tiempo de la conversión} = 2\text{Tad} + \text{N} \times \text{Tad} + (8 - \text{N}) \times (2\text{Tosc}) \qquad 2.6$$

Donde: N = el número de bits de resolución requeridos.

Desde que el TAD es basado en el dispositivo del oscilador, el usuario debe usar algún método (un cronómetro, lazos o bucles por *software*, etc.) para determinar cuando el oscilador A/D puede ser cambiado.

Ejemplo 3 muestra una comparación de tiempo requerido para una conversión con los 4-*bits* de resolución, contra la conversión de resolución de 8-*bits*.
El ejemplo es para dispositivos que operan a 20 MHz y 16 MHz (El reloj A/D se programa para 32TOSC), inmediatamente después de las 6TAD, el reloj A/D se programa para 2TOSC. 2TOSC viola el mínimo tiempo requerido TAD, entonces los 4-*bits* últimos no serán correctos.

Tabla 2.6:.*Tiempos de conversión para resoluciones de 4-bits y 8-bits*

	Freq (MHz)	Resolución	
		4-bit	8-bit
T_{AD}	20	1.6us	1.6us
	16	2.0us	2.0us
T_{OSC}	20	50us	50us
	16	62.5us	62.5us
$2T_{AD} + NxT_{AD} + (8-N)(2T_{OSC})$	20	10us	16us
	16	12.5us	20us

g) Funcionamiento del A/D Durante el modo Sep: El módulo A/D puede operar durante el modo SLEEP. Esto requiere que la fuente de reloj A/D sea RC (ADCS1:ADCS0 = 11).
Cuando se selecciona RC como fuente de reloj, el módulo A/D espera un ciclo de instrucción antes de empezar la conversión. Esto permite la ejecución de la instrucción SLEEP, lo cual elimina todo el ruido de la conversión.
Cuando la conversión se complete el bit GO/DONE se limpiará, es decir se pondrá a cero, y el resultado se carga en el registro ADRES.
Si la interrupción de A/D se habilita, el dispositivo saldrá del modo SLEEP. Si la interrupción de A/D no se habilita, el módulo A/D se apagará entonces, aunque el *bit* ADON permanecerá en 1.

Cuando la fuente de reloj A/D es otra opción de reloj (no RC), una instrucción de SLEEP causará que conversión presente sea abortada y el módulo A/D será apagado, aunque el *bit* ADON permanecerá fijo.

El modo SLEEP durante la conversión A/D hará que baje el consumo, lógicamente el reloj deberá ser RC.

h) Error de precisión del conversor A/D: La exactitud absoluta especificada para el conversor A/D incluye la suma de todas las contribuciones para el error de cuantización, error integral, error diferencial, error de escala completa, error del desplazamiento. Se define como la desviación máxima de una transición real contra una transición ideal para cualquier código.
El error absoluto del conversor A/D se especifica como <±1 LSb para VDD = VREF (por encima del rango de operación del dispositivo especificado). Sin embargo, la exactitud del conversor A/D será mala tanto como VDD diverja de VREF.

Para un rango dado de entradas analógicas, el código de la salida digital será el mismo. Esto es debido a la cuantización de la entrada analógica a un código digital. El error de cuantización es típicamente ± 1/2 LSb y es inherente en el proceso de conversión analógico digital.
La única manera de reducir el error de cuantización es aumentar la resolución del conversor A/D.
El error del desplazamiento mide la primera transición real de un código contra la primera transición ideal de un código. El error del desplazamiento cambia la función de transferencia entera.
El error del desplazamiento puede calibrarse fuera de un sistema o introducido en un sistema a través de la interacción del total de corriente de fuga y la impedancia de la fuente en la entrada analógica.
El error de ganancia mide la desviación máxima de la última transición real y la última transición ideal ajustada para el error del desplazamiento.
Este error aparece como un cambio en la función de transferencia. La diferencia está en el error de ganancia.

Nota: Para operar en modo SLEEP, la fuente de reloj para el módulo A/D debe ser RC (ADCS1:ADCS0 = 11).

El error de *full* escala es que esa escala no tiene en cuenta el error del desplazamiento. El error de ganancia puede calibrarse fuera por *software* [12].

El error de Linealidad se refiere a la uniformidad de los cambios del código. Los errores de Linealidad no pueden calibrarse fuera del sistema [12].

El error de no linealidad integral, mide la transición del código real contra la transición del código ideal ajustado por el error de ganancia para cada código.
La no linealidad diferencial mide la anchura del código real máxima contra la anchura del código ideal. Esta medida es sin ajustes.
En sistemas dónde la frecuencia del dispositivo es baja, se prefiere el uso del reloj A/D de tipo RC. Al moderarse a las altas frecuencias, TAD debe ser derivado del oscilador del dispositivo.
TAD no debe violar el mínimo y debe ser mayor a 8 ms para el funcionamiento preferido. Esto es porque TAD, cuando se deriva de TOSC, se mantiene fuera de las transiciones de fase del reloj.
Esto reduce, en una magnitud grande, los efectos de ruido alternante digital. Esto no es posible con el reloj derivado de RC. La pérdida de exactitud debido al ruido cambiante digital puede ser significante si muchos pines de I/O están activos.
En sistemas dónde el dispositivo entrará en el modo SLEEP, después de la salida de la conversión A/D, el reloj RC es requerido. En este modo, el ruido digital de los módulos en SLEEP se detiene.
Este método da una exactitud alta.

i) Efectos del RESET: Un *Reset* al dispositivo fuerza todos los registros a su estado de *reset* o de reestablecimiento. Esto obliga a apagar el módulo de A/D, y cualquier conversión se aborta. El valor que está en el registro de ADRES no se modifica para un *Power-on Reset*. El registro de ADRES contendrá datos desconocidos después un *Power-on Reset*.

j) Consideraciones de conexión: Si el voltaje de la entrada excede los valores estipulados (VSS o VDD) sobre 0.2V, entonces la exactitud de la conversión está fuera de especificación. Un filtro RC externo a veces se agrega para el anti-*aliasing* de la señal de entrada. El componente R debe seleccionarse para asegurar que la impedancia de la fuente total está por debajo de 10 $k\Omega$.

Cualquier componente externo conectado (vía alta-impedancia) a un pin de la entrada analógica (condensador, diodo del zener, etc.) debe tener una corriente de fuga muy pequeña al pin.

Nota: Debe tenerse cuidado al usar el pin RA0 en las conversiones A/D debido a su proximidad al pin OSC1.

k) Función de Transferencia: La función de transferencia ideal del conversor A/D es como sigue: la primera transición ocurre cuando el voltaje de la entrada analógica (VANO) es VREF/256[2].

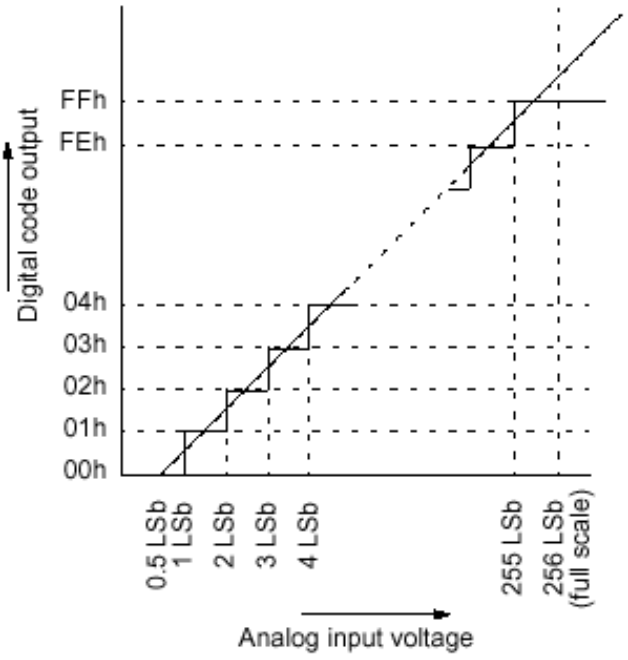


Fig 2.13: *Función de transferencia del convertidor*

[2] En este caso sólo se quiere una resolución de dos décimas. La precisión de la conversión es de 8 bits, hay otros microcontroladores de más bits de precisión.

l) Registros Asociados con la conversión A/D.

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Value on: POR, BOR	Value on all other Resets
0Bh,8Bh	INTCON	GIE	ADIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
89h	ADRES	A/D Result Register								xxxx xxxx	uuuu uuuu
08h	ADCON0	ADCS1	ADCS0	—	CHS1	CHS0	GO/$\overline{\text{DONE}}$	ADIF	ADON	00-0 0000	00-0 0000
88h	ADCON1	—	—	—	—	—	—	PCFG1	PCFG0	---- --00	---- --00
05h	PORTA	—	—	—	RA4	RA3	RA2	RA1	RA0	---x 0000	---u 0000
85h	TRISA	—	—	—	PORTA Data Direction Register					---1 1111	---1 1111

x = desconocido, u = no cambia, - = no implementado se lee como '0'.
Las celdas sombreadas no son usadas para la conversión A/D.

2.5 Reset e Interrupciones.

2.5.1 Fuentes de Reset.

Un *reset* puede ser provocado por diversas fuentes:

1. Una conexión a la alimentación del circuito.
2. Una acción sobre el pin MLCR mientras que el circuito está en modo normal.
3. Una acción sobre el pin MLCR mientras que el circuito está en modo SLEEP.
4. Un desbordamiento del *watchdog*, mientras el circuito está en modo normal.
5. Un desbordamiento del *watchdog*, mientras el circuito está en modo SLEEP.

El comportamiento del circuito y el estado de los registros afectados por un *reset* son diferentes dependiendo de la situación que se produzca. Es posible distinguir por *software* el origen del *reset*. Para ello, basta con leer los *bits* TO y PD del registro de estado como se muestra el la tabla adjunta:

Tabla 2.7: *Determinación del origen de un reset por medio de la lectura de los bits T0 y PD.*

TO	PD	ORIGEN DEL RESET
0	0	Watchdog en modo SLEEP
0	1	Watchdog en funcionamiento normal
--	0	MLRC en modo SLEEP
1	1	Alimentación
--	--	MLRC en funcionamiento normal

En todos los casos, al salir del modo SLEEP mediante el *watchdog* o una interrupción, el contador de programa PC se pone a 000. Por tanto, la primera instrucción ejecutable de su programa debe encontrarse en esta dirección.

Cuando se sale del modo SLEEP mediante el temporizador *watchdog*, el PC se incrementa en una unidad para pasar a la instrucción que sigue a la instrucción SLEEP, igual que cuando se sale de este modo mediante una interrupción (si el *bit* GIE del registro INTCON está a uno).

Tabla 2.8: *Estado de los registros después de los diferentes tipos de reset posibles.*

REGISTRO	DIRECCIONES	CONEXIÓN (a la alimentación)	RESET (watchdog en modo normal)	RESET (watchdog en modo sleep)	RESET MLCR EN MODO NORMAL	RESET MLCR EN MODO SLEEP	SALIDA DEL MODO SLEEP por interrupción
W	-----	XXXX XXXX	uuuu	uuuu	uuuu	uuuu	uuuu
INDIR	00	-----	-----	-----	-----	-----	-----
TRC	01	XXXX XXXX	uuuu	uuuu	uuuu	uuuu	uuuu
PC	02	0000	0000	PC+1	0000	0000	PC+1
STATUS	03	0001 uuuu	0000 1000	uuu00uuu	000uuuuu	00010uuu	uuu1ouuu
FSR	04	uuuu	0000 0000	uuuu	uuuu	uuuu	uuuu
PORT A	05	uuuu	0000 0000	uuuu	uuuu	uuuu	uuuu
PORT B	06	uuuu	0000 0000	uuuu	uuuu	uuuu	uuuu
TRIS A	85	---1 1111	---1 1111	---u uuuu	---1 1111	---1 1111	-uuuuuuu
TRIS B	86	1111 1111	1111 1111	uuuu	1111 1111	1111 1111	uuuu
OPTION	81	1111 1111	1111 1111	uuuu	1111 1111	1111 1111	uuuu
ADCON0	08	uuuu	0000 0000	uuuu	uuuu	uuuu	Uuuu
ADRES	09	uuuu	0000 0000	uuuu	uuuu	uuuu	Uuuu
ADCON1	88	---0 0000	---0 ?000	---uuuuu	---o ?000	---o ?000	---uuuuu
ADRES	89	-----	-----	-----	----	----	----
PCLATCH	0A	---0 0000	---0 0000	---uuuuu	---0 0000	---0 0000	---uuuuu
INTCON	0B	0000 000u	0000 0000	uuuu	000000 0u	oooo	Uuuu

(u) = no cambia (x) = desconocido (-) = no existe (?) = depende de otras condiciones

Las posibles fuentes de reset afectan de forma diversa a los contenidos de los diferentes registros de control, de estado o de datos. En la descripción de cada registro hemos indicado el estado de los bits después de un reset de alimentación. En la tabla anterior encontrará estas mismas indicaciones, pero más completas.
Es aconsejable examinar esta tabla después de un reset antes de cualquier utilización, excepto en el caso de que reinicialice cada registro.

2.5.2 Fuentes de Interrupción

Se dispone de cuatro fuentes de interrupción distintas, que se autorizan o no mediante la puesta a uno de los correspondientes *bits* del registro INTCON (cada una de las cuales dispone de su propio *flag* independientemente de que hayan sido autorizadas o no).
La siguiente figura muestra una representación electrónica de la lógica de interrupciones. Se ve claramente que cada *bit* de indicación de interrupción, se tiene en cuenta si la correspondiente puerta AND se abre mediante el bit de autorización, XXXE, que corresponde. También se pueden autorizar todas en bloque, o no, gracias al *bit* GIE del registro INTCON.
Ver como se puede hacer salir al PIC del modo SLEEP o generar una verdadera interrupción si está en modo normal.

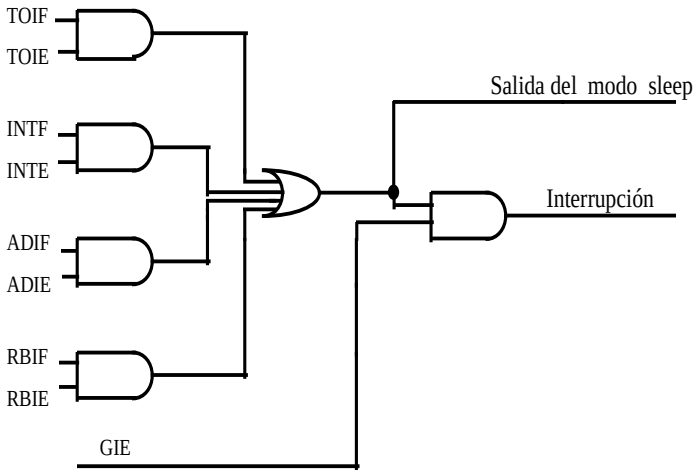


Fig. 2.14: *Esquema del principio de la lógica de interrupción.*

Existe un único vector de interrupción en la dirección 004. Sea cual sea la interrupción, el PC se carga mediante 004 y a continuación, debe comprobar los diferentes indicadores para saber cuál es el elemento que interrumpe.
Durante esta operación, el bit GIE está a cero, para impedir que se tenga en cuenta cualquier nueva interrupción. Después de la ejecución de una instrucción de retorno RETFIE, se pone automáticamente a uno. No está previsto ningún mecanismo de puesta a cero de los bits de indicación de interrupción, por lo que es su programa de tratamiento de interrupciones el que debe encargarse de poner estos bits a cero. Si no es así, no podrá salir de ellas.
Durante la interrupción, el único registro que se salvaguarda en la pila es el PC. Si se desea preservar el contenido de otro registro, tendrá que hacerlo de forma manual.

2.6 Modos de Direccionamientos

Los modos de Direccionamientos son muy pocos y como veremos a continuación, muy a menudo son parte integrante de la propia instrucción.

2.6.1 Inmediato

El dato manipulado por la instrucción se codifica con la propia instrucción. En este caso, el dato en cuestión se denomina literal, nombre que mantendremos a lo largo de todo el capítulo,

con el fin de ser coherentes con las denominaciones adoptadas por *Microchip* en todos sus documentos.

MOVLW k Coloca el literal *k*, que es un valor cualquiera codificado con 8 *bits*, en el registro. de trabajo *w*.

2.6.2 Directo

Es el modo más usado, ya que como se ha visto en los capítulos anteriores, la memoria RAM está dividida en registros específicos y en un conjunto de registros de propósito general. Este modo consiste en codificar el nombre del o de los registros en cuestión directamente en la instrucción.

MOVWF *f* Desplaza el contenido del registro *w* al registro *f*

El registro *f* se referencia mediante su número codificado con 5 o 7 bits, siendo este número en realidad la dirección del *byte* de la RAM correspondiente, después de la utilización del mecanismo de paginación de la memoria, si es necesario.

2.6.3 Bit a Bit

Manipulación de un bit individual en cualquier registro. Este modo de direccionamiento no se utiliza nunca solo, sino que siempre va emparejado con el modo de direccionamiento directo.

BCF *f* , *b* Pone a cero el bit número *b* del registro *f*

El registro *f* se codifica como ya indicamos anteriormente, y *b* es el número del *bit* de este registro, codificado sólo con tres *bits*, ya que no puede variar más que de 0 a 7.

2.6.4 Indirecto

Es el modo más potente y de el ya hemos hablado anteriormente en los registros INDF y FSR. Recordemos rápidamente que el registro FSR es el de selección de registro, en el que se introduce el número del registro direccionado. Lo único extraño del PIC es el modo de notación correspondiente que utiliza el registro INDF, como por ejemplo en:

MOVWF *f* Desplaza el contenido de *w* al registro apuntado por el registro FSR.

La notación *fo* únicamente sirve para indicar el direccionamiento indirecto y por tanto, la utilización del registro FSR como puntero. Incluso cuando esto le parezca pobre, sobre todo si está acostumbrado a otros microcontroladores, éstos son todos los modos de direccionamientos del PIC 16C711. Sin embargo, teniendo en cuenta la eficacia de la estructura de los circuitos y su juego de instrucciones, estos modos le permitirán realizar un buen trabajo.

Antes de concluir este apartado, observe que nos hemos olvidado del tradicional modo RELATIVO, el cual permite en las arquitecturas clásicas los saltos para por ejemplo, los bucles condicionales. Cuando lea la descripción de las instrucciones de bucle, comprenderá por qué no existe este modo de direccionamiento en este circuito.

2.7 Juego de Instrucciones

Con el fin de proporcionarle un texto tan práctico como sea posible, para cada instrucción que se presenta se indica:

- Su sintaxis, indicando la notación de forma precisa.
- Su codificación con 14 *bits*.

Este último punto, no le interesa en lo que concierne a la programación propiamente dicha, ya que estos bits los genera el ensamblador, por supuesto: pero le permitirán comprender cómo se llegan a codificar las instrucciones y los modos de direccionamiento. Además, también le permitirá comprender los límites de los tamaños impuestos a determinados datos, límites que se deben exclusivamente al número de bits que se pueden utilizar.

- Número de palabras y el número de ciclos de máquina utilizados. Puede así comprobar que el principio fundamental de la arquitectura RISC se respeta, con una instrucción por ciclo, salvo raras ocasiones.
- La representación esquemática de la operación realizada por la instrucción.
- Los bits del registro de estado a los que afecta esta operación.
- Comentarios extraídos de la documentación del fabricante o de nuestra experiencia personal cuando sus explicaciones son algo confusas.

La notación adoptada para los datos y direcciones manipuladas por las instrucciones es muy sencilla, y es como sigue:

- *f* representa un número de registro, codificado con siete bits
- *b* representa en número de *bit*, codificado con tres bits. El *bit* 0 es siempre el *bit* de menor peso.
- *k* representa un dato, el literal mencionado antes. EL número de *bits* que se emplea para su codificación es variable, dependiendo de la instrucción.

Un determinado número de instrucciones (ADDWF, ANDWF, etc.) utilizan una notación especial que es:

ADDWF *f,d*

f es el número del registro, como se sabe, y *d* puede tomar el valor 0 o 1, y cambia el comportamiento de la instrucción. Si d está a cero, el resultado se introduce en el registro W, mientras que si *d* está a uno, el resultado se introduce en el registro *f*. Las instrucciones de este tipo son, en realidad, dobles ya que, dependiendo del valor de *d*, realizan:

W operación *f* ----- W si *d* = 0 W operación *f* ----- *f* si *d* = 1

El segundo punto concierne a las instrucciones de bucle condicional:

BTFSC *f,b*

Ya sabemos que esta instrucción comprueba el *bit b* del registro *f*. El significado de esta instrucción es *Bit Test Skip if Clear*, por lo que hace un salto (*skip*) si el bit está a cero (*clear*). Veamos como ocurre esto:

- Si el *bit* está a uno, es decir, si la condición que se comprueba no se cumple, el programa continua su desarrollo normal de forma secuencial

- Si el *bit* está a cero, es decir, si la condición que se comprueba se cumple, el programa salta[3] a la instrucción que sigue a BTFSC en el programa.

Si desea, como generalmente es el caso, saltar a alguna otra parte del programa dependiendo del resultado de la prueba, generalmente adoptará la siguiente sintaxis:

GOTO *k* o **CALL** *k*

Procediendo de esta forma, si la condición es falsa, debe ejecutar la parte de programa llamada por GOTO o CALL, mientras que si la condición es verdadera, el programa continúa con la instrucción que sigue a GOTO o a CALL ya que, en ese caso, se salta justamente esta línea.
Obsérvese que este método, que puede parecer curioso a primera vista, en realidad es muy práctico para tratar los bits de los registros en función de condiciones externas. Así, si un interruptor está conectado a una línea de un puerto paralelo y en función de su estado, debe posicionar un relé controlado por otra línea del puerto paralelo, bastará con hacer:

BTFSC *f,b*

Donde *b* es el número de bit que corresponde al interruptor sobre el puerto en cuestión.

BCF *f,b*

Donde *b* es el número de bit que activa el relé del puerto en cuestión.
Este tratamiento especial de las instrucciones de prueba permiten comprender por qué, al principio del capítulo, le indicábamos que realmente el direccionamiento relativo no existía.

Las demás instrucciones son más clásicas, pero no obstante, nos detendremos un momento en las instrucciones CALL, que es una llamada a una subrutina, y GOTO, que es un salto incondicional.
En estos dos casos anteriores, la dirección llamada parece que está codificada con un número insuficiente de *bits*. En realidad, está codificada con 14 *bits*, pero los *bits* que faltan proceden del PCLATCH.
Tenga también cuidado en las llamadas a subrutinas, de no anidar más llamadas que los niveles que contiene la pila, ocho en este caso.

[3] Todo salto originará un retardo de dos veces el valor de la ejecución de cada instrucción.

Tabla 2.9: *Set de instrucciones.*

NEMÓNICO	OPERACIÓN	DESCRIPCIÓN	14 BITS MSb	LSb	EXAD	RELOJ	ESTADO
ADDWF f,d	w+f → d	Suma W y f	00 0111	dfff ffff	07ff	1	C,DC, Z
ANDWF f,d	w&f → d	AND entre W y f	00 0101	dfff ffff	05ff	1	Z
CLRF f	0 → f	Pone a cero f	00 0001	1fff ffff	018f	1	Z
CLRW -	0 → w	Pone a cero W	00 0001	0000 0011	0100	1	Z
COMF f,d	^f → d	Complementa f	00 0101	dfff ffff	09ff	1	Z
DECF f,d	f-1 → d	Decrementa f	00 0011	dfff ffff	03ff	1	Z
DECFSZ f,d	f-1 → d skip if zero	Dec f, salta si es 0	00 1011	dfff ffff	0Bff	1, (2)	NONE
INCF f,d	f+1 → d	Incrementa f	00 1010	dfff ffff	0Aff	1	Z
INCFSZ f,d	f+1 →d skip if zero	Inc f, salta si es 0	00 1111	dfff ffff	0Fff	1, (2)	NONE
IORWF f,d	w v f → d	OR inclusivo W con f	00 0100	dfff ffff	04ff	1	Z
MOVF f,d	f → d	Mueve f. mira si es 0	00 1000	dfff ffff	08ff	1	Z
MOVWF f,d	w → d	Mueve W a f	00 0000	1fff ffff	008f	1	NONE
NOP -	---	No operación	00 0000	0xx0 0000	0000	1	NONE
RLF f,d		Rota izq f con carry	00 1101	dfff ffff	0Dff	1	C
RRF f,d		Rota drch f con carry	00 1100	dfff ffff	0Cff	1	C
SUBWF f,d		Resta W con f	00 0010	dfff ffff	02ff	1	C,DC, Z
SWAPF f,d		4LSB con 4MSB	00 1110	dfff ffff	0Eff	1	NONE
XORWF f,d	w + f → d or exc	OR exc W con f	00 0110	dfff ffff	06ff	1	Z
Instrucciones en los registros bit por bit							
BCF f,d	o → f(d)	Borra f	01 00bb	bfff ffff	1bff	1	NONE
BSF f,d	1 → f(d)	Pone a 1 f	01 01bb	bfff ffff	1bff	1	NONE
BTFSC f,d		Salta si b(f)=0 1 instr	01 10bb	bfff ffff	1bff	1, (2)	NONE
BTFSS f,d		Salta si b(f)=1 1 instr	01 11bb	bfff ffff	1bff	1, (2)	NONE
Instrucciones con literales y de control							
ADDLW f,d	k + w → w	Añade literal k a W	11 111x	kkkk kkkk	3Ekk	1	C,DC, Z
ANDLW f,d	k & w → w	AND literal con W	11 1001	kkkk kkkk	39kk	1	Z
CALL f,d		Llama a subrutina	10 0kkk	kkkk kkkk	2kkk	2	
CLRWDT f,d		Borra el watchdog	00 0000	0110 0100	0064	1	TO, PD
GOTO f,d		Salta a nueva direcc	10 1kkk	kkkk kkkk	2kkk	2	NONE
IORLW f,d	k v w → w	OR incl literal con W	11 1000	kkkk kkkk	38kk	1	Z
MOVLW f,d	k → w	Carga literal en W	11	kkkk kkkk	30kk	1	NONE

			00xx				
RETFIE f,d		**Retorno interrupción**	**00 0000**	**0000 1001**	**0009**	**2**	**NONE**
RETLW f,d		**RETFIE + MOVLW**	**11 01xx**	**kkkk kkkk**	**34kk**	**2**	**NONE**
RETURN f,d	**TOS → PC**	**Retorno subrutina**	**00 0000**	**0000 1000**	**0008**	**2**	**NONE**
SLEEP f,d	**0→WDT stop oscila**	**Modo SLEEP**	**00 0000**	**0110 0011**	**0063**	**1**	**TO, PD**
SUBLW f,d	**k - w → w**	**Resta W con literal**	**11 110x**	**kkkk kkkk**	**3Ckk**	**1**	**C,DC, Z**
XORLW f,d	**k + w → w or excl**	**OR exc literal con W**	**11 1010**	**kkkk kkkk**	**3Akk**	**1**	**Z**
OPTION	**W→ OPTION regis**	**CargaW en reg *option***	**00 0000**	**0110 0010**	**0062**	**1**	**NONE**
TRIST f	**Tristate port f**	**Carga W en reg trist**	**00 0000**	**0110 0fff**	**006f**	**1**	**NONE**

ADDL*W*	***Añade el contenido de W al literal k, y almacena el resultado en W.***
ADDWF	**Añade el contenido de W al de f, y almacena el resultado en W si d = 0, y en f si d = 1.**
ANDLW	**Efectúa un AND lógico entre el contenido de W y el literal k, y almacena el resultado en W.**
ANDWF	**Efectúa un AND lógico entre el contenido de W y el contenido de f y coloca el resultado en W si: d = 0, y en f si d = 1.**
BCF	**Pone a cero el bit número b de f**
BSF	**Pone a uno el bit número b de f**
BTFSC como	**Si el bit número b de f es nulo, la instrucción que sigue a ésta se ignora y se tratará una NOP. En este caso, y solo en el, la instrucción precisará dos ciclos para ejecutarse**
BTFSS	**Si el bit número b de f está a uno, la instrucción que sigue a ésta se ignora y se tratará como una NOP. En este caso, y solo en el, la instrucción precisará dos ciclos para ejecutarse**
CALL	**Salvaguarda la dirección de vuelta en la pila y después llama a la subrutina situada en la dirección cargada en el PC. También hay que posicionar correctamente el registro PCLATCH antes de ejecutarla.**
CLRF	**Pone el contenido de f a cero y activa el bit Z.**
CLRW	**Pone el contenido del registro W a cero y activa el bit Z.**
CLRWDT	**Pone a cero el registro contador del temporizador Watchdog, así como el predivisor.**
COMF	**Hace el complemento de f bit a bit. El resultado se almacena de nuevo en: f si d = 1, y en W si d = 0 (en este caso, f no varía).**
DECF	**Decrementa el contenido de f en una unidad. El resultado se almacena de nuevo en: f si d = 1, y en W si d = 0 (en este caso, f no varía).**
DECFSZ	**Decrementa el contenido de f en una unidad. El resultado se almacena de nuevo en: f si d = 1, y en W si d = 0 (en este caso, f no varía). Si el resultado es nulo, se ignora la siguiente instrucción y en ese caso, ésta instrucción dura dos ciclos.**
GOTO	**Llama a la subrutina situada en la dirección cargada en el PC También hay que posicionar correctamente el registro PCLATCH antes de ejecutarla.**
INCF	**Incrementa el contenido de f en una unidad. El resultado se almacena de nuevo en: f si d = 1, y en W si d = 0 (en este caso, f no varía).**

INCFSZ	**Incrementa el contenido de f en una unidad. El resultado se almacena de nuevo en: f si d = 1, y en W si d = 0 (en este caso, f no varía). Si el resultado es nulo, se ignora la siguiente instrucción y en ese caso, ésta instrucción dura dos ciclos.**
IORLW	**Efectúa un OR lógico inclusivo entre el contenido de W y el literal k, y almacena el resultado en W.**
IORWF	**Efectúa un OR lógico inclusivo entre el contenido de W y el contenido de f, y almacena el resultado en: f si d = 1, y en W si d = 0**
MOVF	**Desplaza el contenido de f a f si d = 1 ó a W si d = 0 Este desplazamiento que parece inútil, permite comprobar el contenido de f con respecto a cero, ya que esta instrucción actúa sobre el bit Z.**
MOVLW	**Carga W con el literal k.**
MOVWF	**Carga f con el contenido de W.**
NOP	**Solamente consume tiempo de máquina, (un ciclo).**
OPTION	**Carga el registro OPTION con el contenido de W. Esta instrucción no debe usarse, tan solo existe por compatibilidad con circuitos superiores.**
RETFIE	**Carga el PC con el valor que se encuentra en la parte alta de la pila, asegurando así la vuelta de la instrucción. Pone a uno el bit GIE, para autorizar de nuevo que se tengan en cuenta las interrupciones. Dura dos ciclos.**
RETLW	**Carga W con el literal k, y después carga el PC con el valor que se encuentra en la parte superior de la pila, efectuando así un retorno de la subrutina. Dura 2 ciclos**
RETURN	**Carga el PC con el valor que se encuentra en la parte alta de la pila, efectuando así una vuelta de subrutina. Se trata de la instrucción RETLW simplificada. Dura 2 ciclos**
RLF	**Rotación de un bit a la izquierda del contenido de f, pasando por el bit de acarreo C. Si d = 1 el resultado se almacena en f, si d = 0 el resultado se almacena en W.**
RRF	**Rotación de un bit a la izquierda del contenido de f, pasando por el bit de acarreo C. Si d = 1 el resultado se almacena en f, si d = 0 el resultado se almacena en W.**
SLEEP	**Pone el circuito en modo sleep con parada del oscilador.**
SUBLW	**Sustrae el contenido de W del literal k, y almacena el resultado en W. La sustracción se realiza en complemento a dos.**
SUBWF	**Sustrae el contenido de W del contenido de f, y almacena el resultado en: W si d = 0 y en f si d = 1. La sustracción se realiza en complemento a dos.**
SWAPF	**Intercambia los cuatro bits de mayor peso con los cuatro de menor peso de f, y almacena el resultado en f si d = 1, o en W si d = 0.**
TRIS	**Carga el contenido de W en el registro tris del puerto f. Esta instrucción no debe usarse, tan solo existe por compatibilidad con circuitos superiores.**
XORLW	**Efectúa un OR lógico exclusivo entre el contenido de W y el literal k, y almacena el resultado en W.**
XORWF	**Efectúa un OR lógico exclusivo entre el contenido de W y el contenido de f, y almacena el resultado en f si d = 1 y en W si d = 0.**

2.8 Sistema de Desarrollo

2.8.1 Ensamblador y Compilador

En primer lugar, un sistema de desarrollo está formado por un ensamblador y uno o varios compiladores adaptados al lenguaje de alto nivel que se desea utilizar para programar.
El ensamblador traduce las instrucciones que se han escrito usando los nemónicos del lenguaje máquina, a código binario ejecutable por el microcontrolador. La secuencia de nemónicos se llama listado o código fuente del programa, mientras que el código binario se llama objeto o ejecutable.
El compilador traduce las instrucciones que se han escrito en lenguaje de alto nivel, que constituyen también lo que se llama listado o código fuente, a código binario ejecutable por el microcontrolador que constituye el código objeto.
En un sistema de desarrollo bien concebido, es primordial que el compilador utilizado para el lenguaje de alto nivel tenga una interfaz perfecta con el ensamblador, de forma que se puedan insertar subrutinas en lenguaje máquina en el mismo seno del programa de alto nivel.
Estos dos programas deben ejecutarse sobre una máquina denominada *host*, ó plataforma de desarrollo. Esta máquina puede ser de diversas formas, la solución más generalizada, es un compatible PC el cual permite minimizar la inversión.
Una vez que el programa se ha escrito y ensamblado o compilado sobre la máquina *host*, se está en posesión de un binario ejecutable. Es casi indispensable probar este programa, haciéndole funcionar en condiciones tan próximas como sea posible a las de utilización real. Para hacer esto, existen varias posibles soluciones.

2.8.2 Emulador y Simulador

Utilizar un emulador es la solución más costosa, es un montaje especial, que puede ser muy complejo y que se comporta exactamente como el microcontrolador al que reemplaza.
Como el emulador es una versión fragmentada del microcontrolador al que reemplaza, se tiene acceso a las distintas señales internas de éste y en particular, se puede saber por cuales direcciones pasa el programa, que ocurre en los diversos registros de los periféricos internos, etc. Del mismo modo, se puede introducir puntos de parada, para leer el estado de ciertas memorias o de ciertos registros, y ejecutar en tiempo real.
La segunda solución no siempre permite realizar todas las pruebas necesarias. Consiste en emplear un simulador, el cual es un programa escrito especialmente para el microcontrolador que se va a simular[4].
Por tanto, el simulador es un producto mucho más sencillo que el emulador, ya que no es más que un programa. Por tanto, su precio de costo es mucho más bajo.
El simulador realiza la ejecución del programa, aproximadamente, de 10 a 100 veces menos deprisa que lo haría el mismo programa directamente sobre el microcontrolador, por eso determinadas operaciones, en las que son necesarios tiempos muy precisos o críticos, no se

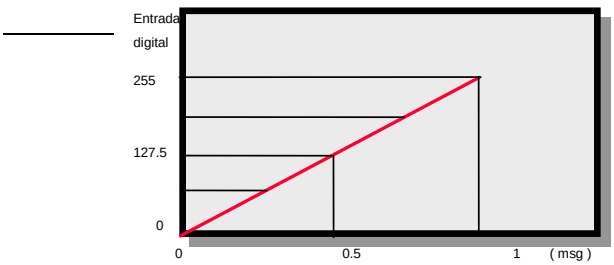
[4] Para el desarrollo de la presente tesis se usó el MPLAB que es un *software* compilador de lenguaje ensamblador para microntroladores PIC.

puede probar mediante la simulación. No obstante, bien utilizado permite desarrollar aplicaciones interesantes mediante una inversión mínima.

2.9 Desarrollo de un programa para una conversión A/D

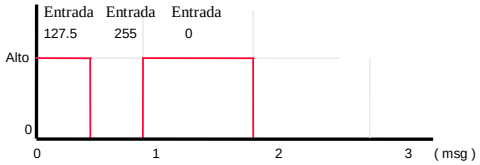
Diseñar un programa que de una frecuencia de salida de 1 KHz por el pin B0 del puerto B, el cual estará ofreciendo un pulso positivo proporcional a una entrada exterior, y el resto a cero hasta completar la duración total de la frecuencia anteriormente mencionada.
El pulso positivo tendrá una duración máxima de un milisegundo y su relación con la entrada será lineal como se muestra en la figura.
El valor máximo de la entrada, será de 255 en digital, que corresponderá con una duración de un milisegundo del pulso positivo, y el mínimo de un valor 0, que corresponderá con un pulso positivo de duración nula.

La entrada mencionada se dará por el puerto A.



Duración del pulso

Salida por el puerto en funcionamiento



Cada vez que se desborde el temporizador se decrementará un registro cargado inicialmente a FF en digital, con lo que estaré contando 256 x 255 pulsos de reloj (65280 pulsos).
Tengo que dar una frecuencia de salida de 1 KHz, por lo tanto, como la entrada máxima de 255 debe corresponder con un pulso positivo de un milisegundo, tendré:

65280ms --------- 1 KHz
1ms ---------- 0.015318Hz

Le asignaré una tasa de división de 256, por lo que la frecuencia del reloj deberá ser de 3.92156 Hz.

El registro OPTION, tras la puesta en marcha del circuito, están a uno todos sus bits, por lo que debo poner el bit 3 de este registro a cero para asignar dicha predivisión. El registro OPTION ocupa la posición 81 de la página uno dentro de los registros internos.

El PORT A inicialmente está configurado como entrada por tener todos los *bits* del registro TRISA a uno, y como quiero sacar los pulsos por el *bit* uno de dicho puerto, pondré a cero dicho *bit* del registro TRISA.
El registro TRISA ocupa la posición 85 de la página uno dentro de los registros internos.

```
INICIO:
        BCF 81,03           ; pone a cero el bit 3 del registro OPTION con lo que asigno el
```

```
                    predivisor al temporizador. PS0,PS1,PS2 tienen el valor 1,1,1 en
                    el registro OPTION.
        MOVLW FF    ; carga el registro de trabajo ( W ) con el literal FF.
        MOVWF 1C    ; carga el literal FF en un registro de propósito general ( 1C ).
        MOVF f6,01  ; pone el registro f6 ( PORT B ) en el registro W, actuando esta
```

Operación sobre el *bit* de cero en el caso de que en el PORT B estuviese el valor cero.

```
        BTFSC f3, 02    ; ignora la siguiente instrucción, tratándola como una operación
                        NOP                 ,
                        en el caso de que el bit 2 del registro f3
                        ( STATUS ) sea cero, es decir si la ultima  operación es distinta
                        de cero.
        GOTO INICIO     ; salta a inicio solamente en el caso de que la entrada del puerto
                        B sea  nula.
        BCF 85,01       ; Pone a cero el bit 1 del registro TRISA, asignándolo así como
                        salida.
        BSF f5, 01      ; pone a uno el bit 1 del PORT A, o bit RA1, que será la salida
                        del sistema.
        MOVWF 0C        ; carga el registro 0C con el contenido del registro W que es el
                        valor leído en el  PORT B.
CUENTA:
        GOTO ENTRADA    ; salta a la posición de memoria “entrada”

CUENTA1:
        BCF 0B, 02      ; pone a cero el bit 2 del registro INTCON
                         ( debe hacerse por software ).
        DEC 1C, 01      ; decrementa el registro 1C, el cual estaba inicialmente cargado
                        con el  literal FF.
        DECFSZ 0C, 01   ; decrementa 0C y guarda en 0C, el cual estaba inicialmente
                        cargado
                        con el valor del PORT B y saltará la siguiente instrucción en el
                        caso
                         de que el resultado sea cero.
        GOTO CUENTA     ; salta a la posición de memoria “cuenta”, solamente no leerá
                        esta  instrucción en el caso de que el registro 0C llegue a cero.
        BCF f5, 01      ; pone a cero el bit 1 del PORT A o bit RA1.
        GOTO FINAL      ; salta a la posición de memoria final.

ENTRADA:
        BTFSC 0B, 02    ; saltará la siguiente instrucción, si el bit 2 del registro INTCON
                        es cero,
                         es decir, en el caso de que no se halla producido overflow en el
                        temporizador.
        RETURN          ; retorna a la última llamada de subrutina producida. Carga el PC
                        con el
                        valor que se encuentra en la parte superior de la pila.     ( * )
```

```
        GOTO ENTRADA    ; salta a la posición de memoria "entrada".
        GOTO CUENTA1    ; ( * )

FINAL:
        BTFSS 0B, 02    ; saltará la siguiente instrucción, si el bit 2 del registro INTCON
                        es uno, es decir, en el caso de que si se halla producido overflow
                        en el temporizador.
        GOTO FINAL      ; salta a la posición de memoria final, esta instrucción se leerá en
                        el caso de que aun no se halla producido overflow.
        BCF 0B, 02      ; pone a cero el bit 2 del registro INTCON
                        (overflow flag)
        DECFSZ 1C, 01   ; decrementa el registro 1C, el cual estaba inicialmente cargado
                        con el literal FF y saltará la siguiente instrucción si el resultado
                        es cero.
        GOTO FINAL      ; salta a la posición de memoria final.
        GOTO INICIO     ; comienza de nuevo el programa.
```

(*) Los ocho registros de pila deben considerarse como un buffer de memoria circular, lo que significa que, si se introducen más de ocho valores del PC, el noveno valor tomará la posición del primero, y así sucesivamente.

La posición de memoria "final" se usa para completar el ciclo y que éste tenga la frecuencia deseada, y solamente se accederá a ella una vez se complete el ciclo de salida en alto del PORTA.
La posición de memoria "entrada" se usa para detectar el *overflow* del *timer*, y tendrá ciclos con la posición "cuenta" para ir decrementando los registros 0C y 1C[5].

En el siguiente capítulo estudiaremos el *Hardware* del regulador propuesto, basado en el microcontrolador estudiado en este capítulo.

[5] Consulte las aplicaciones de microcontroladores en la página de *microchip*, www.microchip.com, para más ejemplos.

CAPÍTULO-3

DESCRIPCIÓN DEL HARDWARE DEL SISTEMA PROPUESTO

El presente capítulo tiene como objetivo:

Una descripción detallada del regulador implementado, en cuanto a *hardware*.

Prácticamente se ven los criterios de diseño del regulador y el principio en el que se basa la respectiva regulación.

Se verá el diseño del *hardware* del regulador, la seguridad que brinda ante fallas comunes ya sea del panel o del usuario.

Las limitaciones de uso en cuanto a máxima tensión y corriente permisibles.

Se verán los componentes que intervienen en la comunicación PC –EEPROM, que son los que vendrían a conformar la tarjeta de adquisición de datos.

Se ha divido el respectivo estudio en 8 etapas fundamentales:

3.1 Principios de regulación.
3.2 Panel solar – batería.
3.3 Batería - Carga.
3.4 Indicadores.
3.5 Niveles permisibles.
3.6 Seguridad.
3.7 PIC-EEPROM.
3.8 EEPROM-PC.

3.1 Principios de regulación.

Para poder establecer el principio de regulación, veremos cuales son las tareas puntuales que el regulador diseñado cumple.

Lo que tiene que hacer la regulación es mantener la tensión de la batería entre unos límites establecidos. En el regulador que se ha implementado estos límites son puestos por *software* (esto se verá en el análisis del *software*), claro que existe un límite superior fijo (tope), pero que se puede variar también mediante *hardware.*

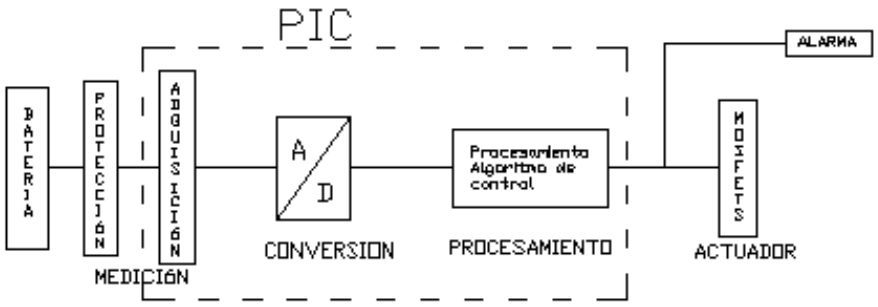


Fig. 3.1: *Para la regulación de la batería y la carga se utiliza un PIC 16C711.*

Para esto el regulador debe sensar (conversión analógica-digital) la tensión de la batería con buena **exactitud**[6] para que sepa su valor instante a instante y de esta manera hacer una excelente regulación.

Como la mayoría de las baterías son de 12V – 13V y corremos en un día claro la tensión en el panel puede llegar hasta 17 voltios, entonces escogemos los siguientes límites:

Límite inferior 11V, puede ser cambiado a 11.5 por *hardware* o y/o *software*
Límite superior 13.5V, puede ser cambiado a 14 por *hardware* y/o *software*.

[6] La exactitud depende del *hardware y software*. Para niveles de tensión mayores a 5 voltios, se debe dividir el voltaje mediante un partidor de tensión en base a resistencias de precisión de ± 0.5%.

3.1.1 Límite inferior

El límite inferior se escoge de 11 V para poder prolongar un mayor tiempo la vida de la batería, ya que se sabe por lo dicho en el primer capítulo que a medida que menos profunda es la descarga habrá menos grado de sulfatación por lo tanto habrá menos deterioro de la batería.
Tenemos que considerar algo muy importante, que es la **histéresis**[7] que ocurre cuando el regulador conecta la carga con la batería. Sucede que cuando la carga es conectada, está hará inmediatamente que la tensión de la batería baje levemente, por lo tanto si el límite inferior no es cambiado al instante, el regulador verá el voltaje de la batería menor que el límite inferior a la que fue conectada con la carga, por lo tanto se tendrá una situación de **histéresis** que posiblemente hará que el sistema permanezca en un lazo (como un lazo electrónico, del cual tardará en salir), lo que no es conveniente.
Para solucionar este problema hay dos alternativas:

a) El usuario podría esperar hasta que se supere en 1V la tensión inferior de la batería y recién usar la energía, lo que nos obligaría a usar un *display* o un LCD, entonces el costo de la tarjeta incrementaría y se tendrían que disponer de una mayor cantidad de salidas en el microcontrolador, por lo que se debería usar otro micro económicamente más costoso.
A todo esto se suma el riesgo de que el usuario no siga dichas instrucciones y se entraría en el mencionado lazo.

b) Mediante programa en el microcontrolador PIC, cambiar el límite inferior de desconexión inmediatamente después que el regulador haya conectado la carga con la batería. Esta es la mejor opción, pues se puede por programa, bajar el límite inferior y luego conectar al usuario, esto evitaría la histéresis mencionada.

Es decir que cuando la intensidad luminosa de los rayos solares que inciden en el panel solar hacen que éste tenga una tensión de 11 voltios, el microcontrolador, cerebro del regulador, hará que cambie el límite permisible inferior de 11 voltios a 10 voltios inmediatamente después que se active el **actuador** entre la batería y el usuario o carga.

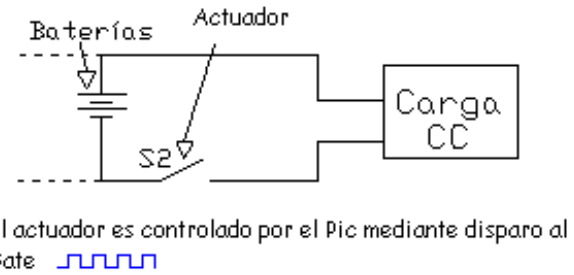


Fig. 3.2: *Posición del Actuador entre Batería y Carga*

[7] En el regulador propuesto, la diferencia de tensión debida a la histéresis es modificable por *software*. Esta flexibilidad hace funcional a esta tarjeta para cualquier capacidad de baterías ya que hay algunas cuya carga y descarga son más rápidas que otras.

Esta es la solución más adecuada que podemos darle a este problema que a veces en la mayoría de reguladores analógicos origina un problema de conmutación constante hasta que la tensión de la batería supera el valor de 11 voltios, que para esto lógicamente tendría que pasas un tiempo mayor y durante este tiempo el usuario no sabría si está en estado de conexión o desconexión.
En el análisis del *hardware* se verá en detalle al actuador.

3.1.2 Límite superior.

El límite superior de la batería es modificable por programa, eso depende mucho de las exigencias del usuario, pero no hay que olvidar que las baterías funcionan hasta un límite de tensión determinado por fabricación, por lo que es imprescindible que la carga de la batería esté totalmente controlada.
El límite escogido es 13.5 voltios. Cuando la tensión de la batería está dentro de los límites mínimo y máximo establecidos, no hay ningún problema, pero una vez que la tensión máxima es superada en un **mínimo valor**[8], el regulador debe estabilizar la tensión de la batería en 13.5, lógicamente mientras que el voltaje del panel se encuentre sobre éste valor.
Para lograr esta estabilidad de la batería en 13.5 voltios mencionada anteriormente, se utiliza un principio que se basa en la técnica de modulación por ancho de pulso.
Cuando la tensión en la batería pasa los 13.5 voltios, lo que se hace es conmutar el actuador a la misma frecuencia pero con diferentes *duty cicles* cada segundo, tal así que la onda positiva de 5 voltios que emite el PIC, disminuya segundo a segundo a medida que se disminuye el *duty cicle*. Lo que en realidad se está haciendo es cortando la conducción del actuador cada vez más si es que la tensión aún sigue siendo mayor a 13.5. Esto quiere decir que cada segundo, aparte de cortar la onda, se está comparando la tensión de la batería con el límite superior (13.5 voltios), cuando la batería esté en este límite o por debajo, el microcontrolador ya no bajará más el *duty cicle* de la onda de disparo al actuador (que está entre el panel solar y la batería), sino por lo contrario la subirá, haciendo un seguimiento de gran exactitud al límite superior (13,5 voltios).
Esto asegura una regulación muy efectiva, la cual no se limita a cortar el disparo total del actuador cada vez que la tensión en el panel supere los 13.5 voltios.

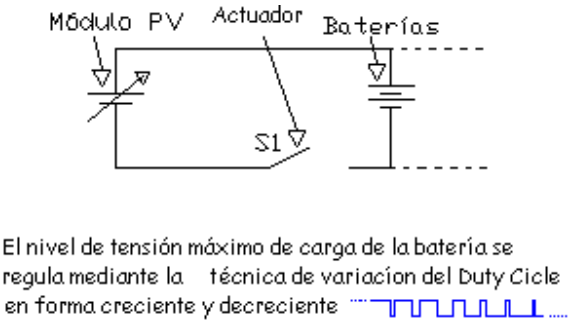


Fig. 3.2: *Técnica de Regulación de tensión*

[8] El mínimo valor, es la variación de la tensión de la batería en 10 segundos. Esto es porque en este tiempo la modulación corta definitivamente el pulso de activación del *mosfet* Panel – Batería.

Existen dos límites: inferior y superior, pero pasado el límite inferior se entra a una región que comprende entre 11.0 y 13.5 voltios en donde tenemos la certeza de que los dos actuadores están activados. A esta región se le llama **banda de flotación**[9], que es donde los puertos del microcontrolador permanecen inalterables, es decir no cambian de estado.

En el siguiente cuadro Resumo todas las características de regulación que he mencionado.

Tabla 3.1: *Características de Regulación*

Tensión 12V	***Batería***
Reconexión de consumo	11V
Desconexión de consumo	10V
Banda de Igualación	13.5 – 13.55V
Carga Profunda	13.5V
Banda de Flotación	11.0 – 13.5V
Alarma de baja	10.5V
Alarma de Alta	13.5V
Rearme de carga profunda	13.5V

La **banda de igualación** nos indica hasta cuanto se puede elevar el voltaje de la batería, mientras el regulador trata de estabilizar la tensión de la batería. Si es que se supera este valor sonará la alarma o se activará un *led* rojo indicador de alta.
Esta característica es inversa a la de los productos comerciales que usan el *led* verde como indicador de alta, pero esto como mejor parezca puede ser cambiado por programa. Debido a que en el regulador propuesto la regulación es automática, la tensión en la batería nunca llegaría a los 14 Voltios, su regulación es tan efectiva que como máximo le puede tomar 10 segundos para estabilizar la tensión en 13.5 voltios.
Entonces la banda de igualación en realidad es mucho más corta que la de un regulador normal, llega hasta el orden de las décimas, es decir está en un rango de 13.5 – 13.55, es más podríamos variar esta banda, cambiando la precisión de la conversión por *software* y por *hardware*.

[9] Hay autores de diferentes libros que definen como banda de flotación a un estado anterior pero próximo al de desconexión del panel.

3.2 Panel solar – batería

Las tierras (polaridad negativa) de la batería y del panel solar, se conectan por un dispositivo que anteriormente llamamos actuador, este actuador es un transistor MOSFET en modo de enriquecimiento. Cada vez que el microcontrolador activa el *gate* del MOSFET, éste conducirá y la batería se cargará.
Por el paso de la corriente y por su alta conmutación, estos dispositivos se calientan, por lo que necesitan un sistema de refrigeración externa. Lo que se hizo es aprovechar la empaquetadura o la carcasa y usarla como área de dispersión de calor, esto lo logramos fijando la parte metálica de los MOSFETS a la carcasa del regulador.
El transistor MOSFET que escogí es el IRFZ44 debido a su baja impedancia, del orden de los 17.5 $m\Omega$, la cual origina menos pérdidas. En la sección de anexos adjunto la hoja de datos de este dispositivo.

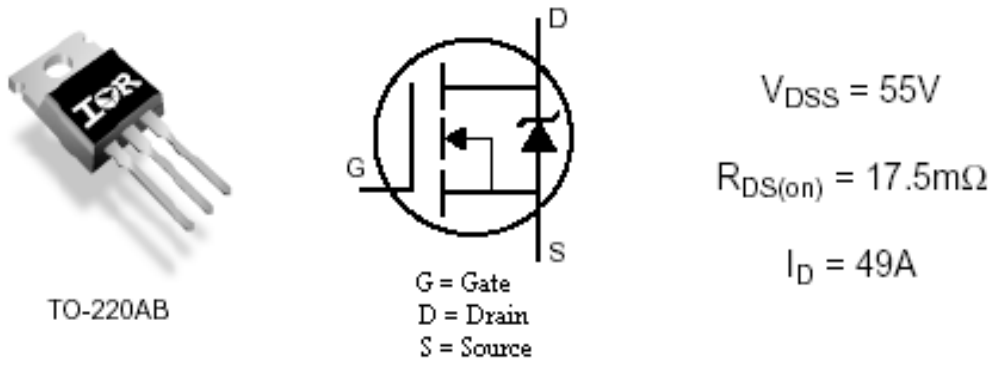


Fig. 3.3: *Mosfet IRFZ44, características principales.*

El problema del disparo: es que para hacer que el MOSFET conduzca se le tiene que dar una tensión *gate-source* V_{GS} de 5V. En la salida del microcontrolador tenemos 5 voltios, lo malo es que el MOSFET sólo conduce de *drain* a *source*, por lo que necesariamente la puerta *source* del MOSFET va a tener que estar al mismo potencial que la tierra del panel.

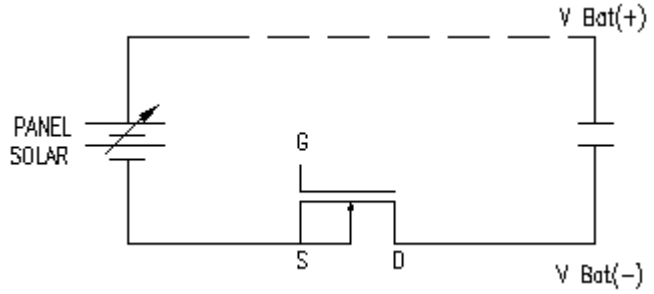


Fig. 3.4: *Transistor Mosfet conectado entre las dos polaridades negativas*

El transistor *mosfet* IRFZ44 en modo de enriquecimiento es el que controla el flujo de carga del panel hacia la batería mediante el disparo de su *gate*. El control del *gate* del MOSFET lo

hace el microcontrolador a través de sus puertos mediante un acondicionamiento de la señal que sale del PIC.

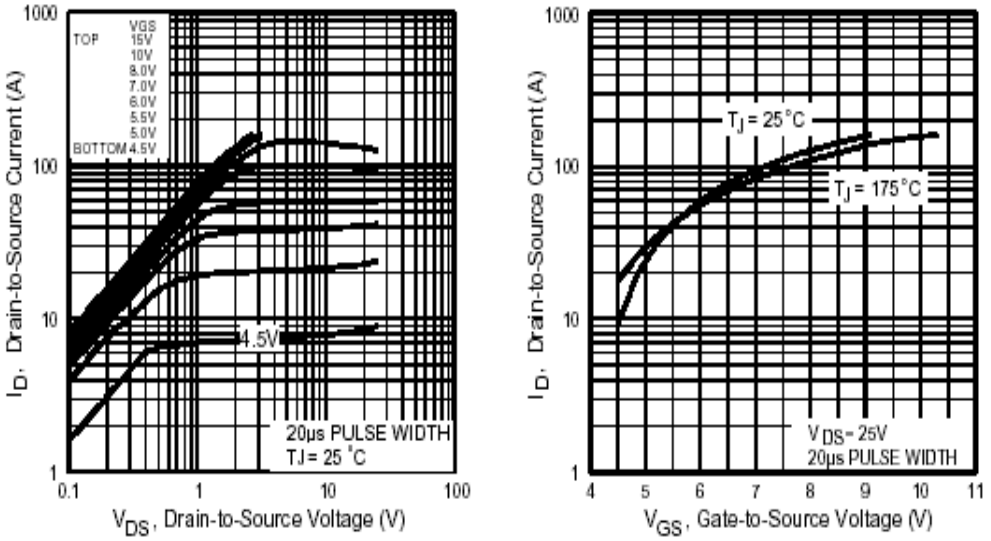


Fig. 3.5: *Características típicas de salida.*

Como podemos ver en las características típicas de salida, en el primer gráfico están las curvas correspondientes a I_D versus V_{DS}, para diferentes V_{GS}.
En esto tenemos que poner cuidado debido a que por el simple hecho de que V_{GS} sea 4.5 voltios e I_D sea 8 amperios, estaríamos en una zona dentro de la gráfica de muy baja pendiente entre los valores de 1V y 10V para V_{DS}. V_{DS} es el voltaje entre *drain* y *source* del MOSFET, que corresponden a la tierra de la batería (bat (-)) y la tierra del panel solar (Panel (-)). Es decir que de no tener cuidado se podría llegar a una diferencia de potencial de 10 voltios entre las dos tierras de la batería y del panel.
V_{GS} siempre tiene que ser mayor o igual a 4.5 voltios, la referencia de este voltaje es a la tierra del panel solar, ya que tiene el mismo potencial de la puerta S del MOSFET. Entonces si estamos en la situación anterior en donde la tierra del panel se encuentra en un potencial 10 voltios menor que la tierra de la batería, es decir:

$$V_{GS} = 10 \text{ voltios}$$

$$V_G - V_S = 10 \text{ voltios}$$

$$V_{bat(-)} - V_{panel(-)} = 10 \text{ voltios}$$

La tensión V_{GS} la sacamos de la batería aprovechando esa caída de tensión V_{DS}, pero este es un caso extremo peligroso, si la batería esta cargada al máximo (13.5 voltios), entonces la diferencia de potencial entre la polaridad positiva de la batería y la tierra del panel sería 23.5 voltios.
De hecho esta tensión no es en ese momento V_{GS}, debido a que el voltaje de la batería pasa por un limitador en base a resistencias (R8 y R9) pero que disminuye la tensión en este caso en sólo 5 voltios, por lo tanto la tensión V_{GS} sería 18.5 voltios, que lógicamente quemaría al MOSFET.

En la sección de protección de la tarjeta se ve los dispositivos de protección es el caso de que se llegue a esta situación, cosa no muy dable debido a que también existe en la tarjeta otros dispositivos de protección que limita la corriente I_D hasta 6 amperios y esto sólo produce una caída de tensión V_{DS} menor a 1 voltio.(Ver en la primera curva). El microcontrolador tiene como salida 5 Voltios en 1 lógico, pero estos 5 voltios son con referencia a la tierra de la batería más no del panel.
Por la caída de tensión que hay V_{GD}, la tierra del panel no tiene el mismo potencial que la tierra de la batería. Entonces es seguro que se implementó un sistema de disparo para el MOSFET, que se muestra a continuación.

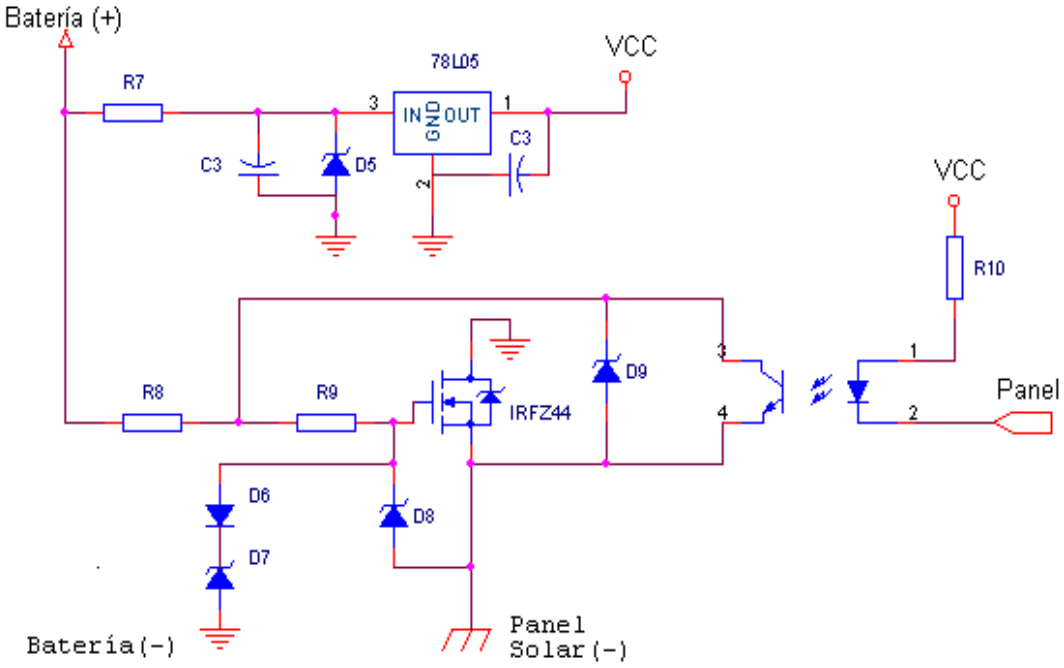
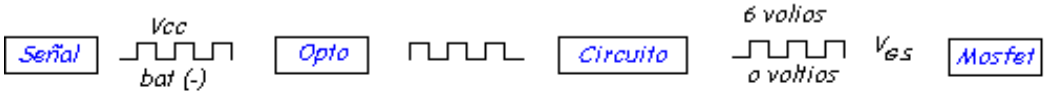


Fig. 3.5: *Circuito de disparo al Mosfet entre el panel solar y la batería.*

Como podemos apreciar en la figura, para acoplar la señal del microcontrolador (Panel) con la señal que debe ir al MOSFET, se usa un **opto acoplador**.
Cuando la señal de salida del microcontrolador que se encarga de activar la carga de la batería, está en 1 lógico (5 voltios), el **opto acoplador** deja de conducir, lo que llevaría a tener una señal de activación del MOSFET, proveniente de la tensión de la batería, que lógicamente es mayor o igual a 10 voltios. Debido a esta alta tensión he puesto un partidor de tensión (R8 y R9) que hace que el voltaje V_{GS} sea de 5 a 6 voltios.
Es decir una señal activa del microcontrolador (Panel) desactiva el **opto acoplador** y esto hace que se active el MOSFET, es decir que conduzca.



Vcc = Voltage de alimentación del Pic
bat (-) = Tierra de la batería
Vgs = Tensión de gate - source para el Mosfet

Fig. 3.6: *Secuencia de disparo al Mosfet entre el panel solar y la batería.*

Cuando la señal del microcontrolador es 0 voltios, el **opto acoplador** conduce lo que hace que el voltaje V_{GS} sea 0 ó negativo, es decir que el MOSFET no conduzca.

También se puede ver en el esquema del circuito, que la tensión de alimentación para el PIC se obtiene por medio de un **regulador de tensión**[10] (78L05), el cual tiene como entrada la tensión de la batería y como salida 5 voltios (en la sección de anexos está su hoja de datos).
Es decir cuando se hace por primera vez la instalación solar, necesariamente el voltaje de la batería tiene que ser mayor a 5 voltios, debido a que para valores menores a 5 voltios como entrada al regulador, no se obtendría en el regulador la tensión de salida en 5 voltios. Lo que pasa es que el regulador es un limitador de tensión, consta de varios diodos **zener**, cuya tensión **zener** suman 5 voltios.
Luego, la batería nunca bajará de los 10 voltios, debido a que el microcontrolador desactivará al usuario justo cuando la batería baje de 10 voltios.

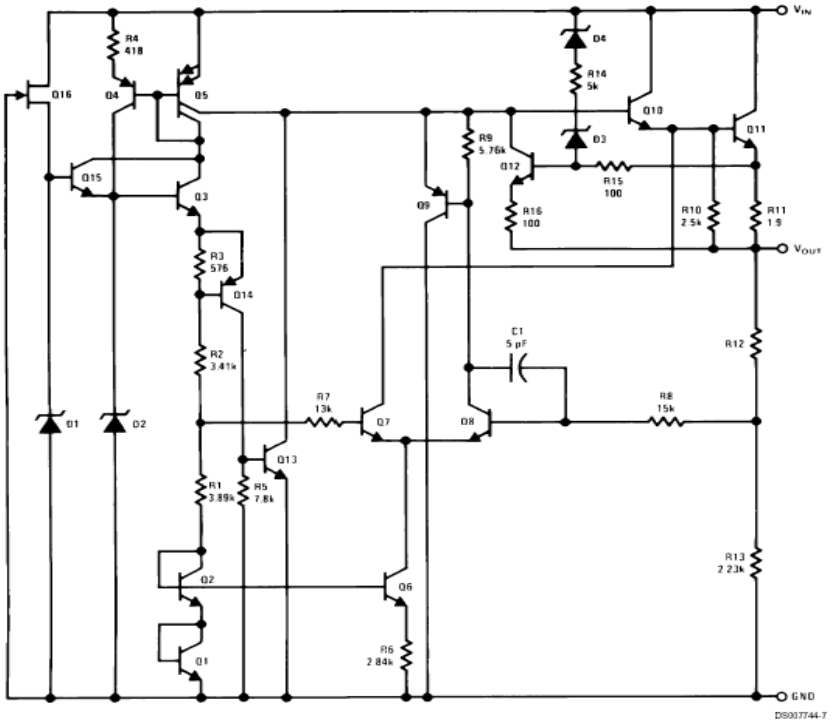


Fig. 3.7: *Circuito interno del regulador*

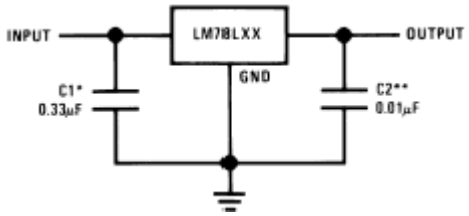


Fig. 3.8: *Esquema de conexión*

[10] Este regulador de tensión trabaja a partir de una entrada de tensión mínima, igual a 9 voltios. Esto indica que la tensión de la batería nunca debe bajar de este valor; de ser así el sistema se desactivará.

EL regulador prácticamente estabiliza la tensión de salida en 5 voltios, teniendo como entrada tensiones mayores o iguales a 5 voltios. La mayoría de reguladores se diferencian en la exactitud y en la potencia, es decir Máxima corriente que pueden entregar.
La potencia consumida por el PIC depende instantáneamente de la tarea asignada, es decir, la corriente en los puertos depende de la orden ejecutada, puede ser que el puerto este configurado como entrada entonces le corresponde una determinada corriente máxima, pero también puede que esté configurado el puerto como salida que le corresponde otra corriente máxima (ver la sección de anexos).

Como ya se sabe el microcontrolador PIC 16c711, es el dispositivo fundamental de la tarjeta construida, es el elemento inteligente de la tarjeta, por lo tanto le debemos mayor cuidado.
El PIC es prácticamente un microcomputador diseñado para una tarea específica.
En él están encriptados buses de direcciones, de datos, memoria EEPROM, RAM y aparte en especial nuestro PIC posee un convertidor analógico digital (Ver Hoja de datos en anexos).

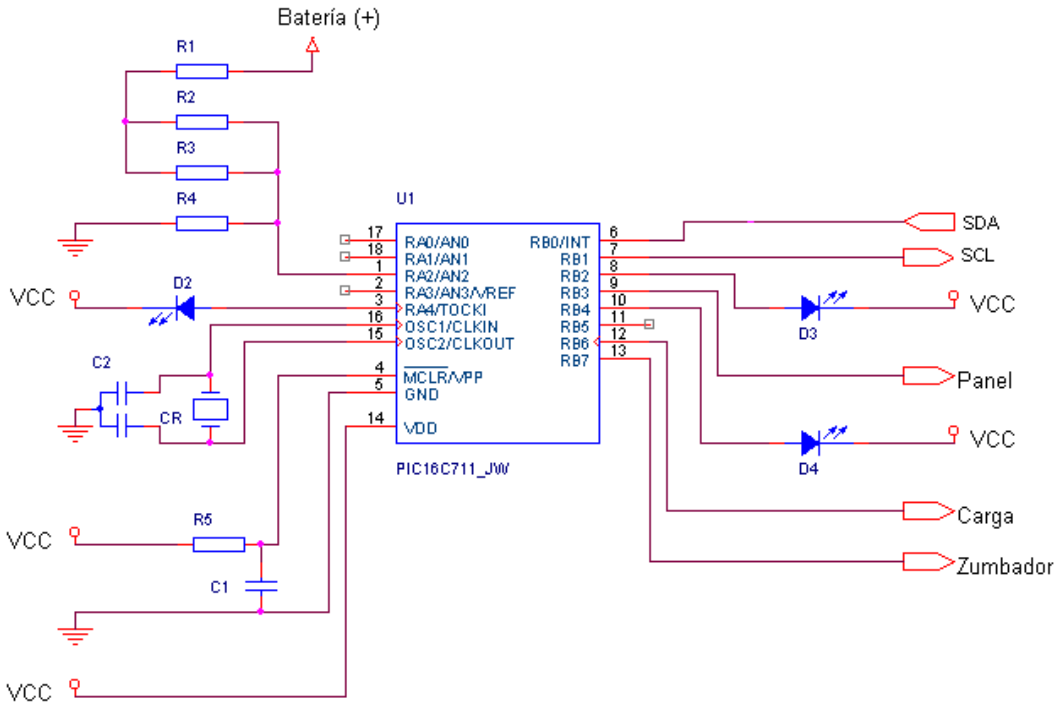


Fig. 3.9: *Conexión de los pines del PIC*

3.3 Batería-Carga.

La conexión entre la Batería y la carga, se realiza mediante otro MOSFET IRFZ 44 que es controlado directamente por el PIC.
La tierra de la batería esta conectada con la puerta source del MOSFET y la puerta *drain* esta conectada a la tierra de la carga.
La conexión y desconexión es gobernada por el *gate* que esta conectado directamente al Pin RB6 del PIC, etiquetado como *Load*.

Para seguridad de la máxima diferencia de tensión entre el *gate* y la tierra de la carga, usamos un diodo zener de 33 voltios de ruptura y un diodo que no permite la circulación de corriente desde la tierra de la batería al *gate* del MOSFET.
Para seguridad de la tensión entre el *gate* y la tierra de la batería se usa un zener de 15 voltios de ruptura. Esto es por el máximo voltaje V_{GS} que permite el MOSFET.

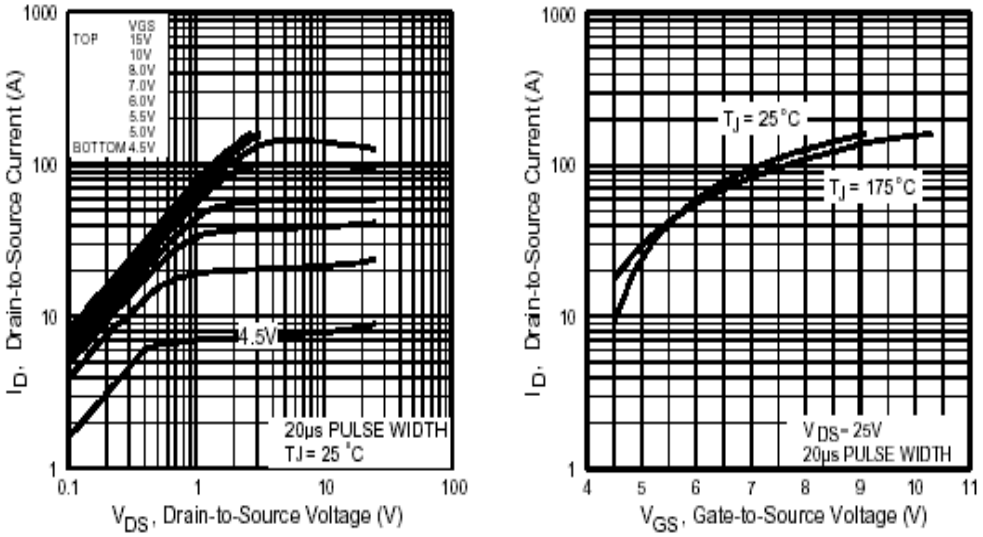


Fig. 3.10: *Voltaje Gate-Source del Mosfet en función de la corriente I_D*

Como vemos en la gráfica del MOSFET para un Vgs de 5 voltios, Ids puede pasar los 6 amperios y Vds es 0.15 voltios, pero para 4.5 voltios o menos, para Ids mayor a 6 amperios, Vds puede llegar a ser mayor a 10 voltios.

$$V_{DS} = V_D - V_S$$

V_S = Tierra de la batería.

V_D = Tierra de la Carga.

$$V_{GS} = 5V = V_G - V_S = V_G - V_D + V_{DS}$$

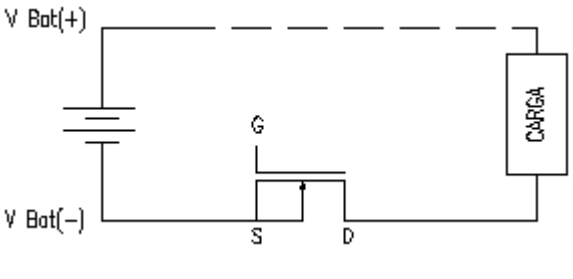
$$V_{DS} = 10\ V$$

$$V_D = V_G + 5V.$$

$$V_D > V_G$$

Por eso es necesario el diodo que evite la circulación entre la tierra de la carga y el *gate*.
La conexión y desconexión se hace con el respectivo disparo del PIC, directamente al MOSFET mediante el pin *Load*. La relación es directa, es decir un nivel el alto en este pin significa una conducción del MOSFET y un nivel bajo, para el flujo de electrones.

Este MOSFET es disparado con nivel alto cuando se debe conectar la carga a la batería y viceversa.



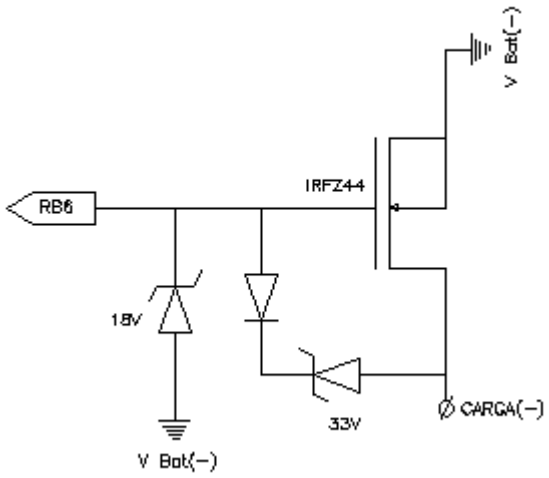


Fig. 3.11: *Conexión del mosfet entre Bat. – Carga*

Fig. 3.12: *Protección del mosfet*

3.4 Indicadores.

Los señalizadores empleados son tres *leds*, verde, ámbar y rojo, aparte una alarma que viene a ser un zumbador.
Los estados lógicos de estos *leds* corresponden a una indicación de nivel de nivel de tensión y del estado de carga de la batería.

Fig. 3.11: *Conexión del mosfet entre Bat. – Carga*

RB4	RB2	RA4	
Verde	Ámbar	Rojo	V (Batería)
1	1	0	V < 11V
0	1	0	11.5 <V< 13.5
0	1	1	V > 13.5

El zumbador señaliza la batería con carga baja, justo antes de desconectar al usuario.

3.5 Niveles Permisibles.

Para la conversión analógica digital es necesario un divisor de tensión, para así poder tener valores de voltajes permisibles en la entrada del canal A/D.

$$Vpin = Vbat \times \left(1 - \frac{(R1 + R2 + R3)}{(R1 + R2 + R3 + R4)}\right)$$

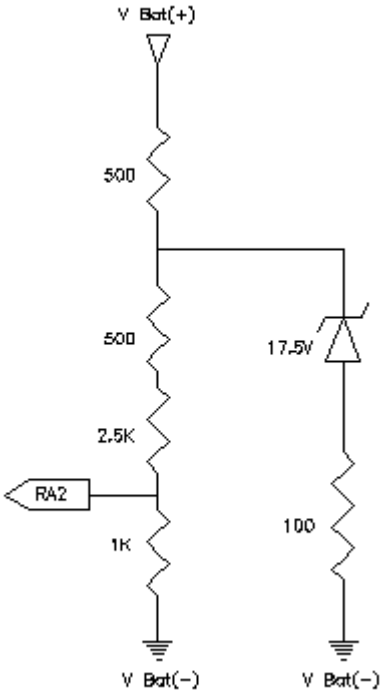
$$Vpin = Vbat \div \left(\frac{(R1 + R2 + R3 + R4)}{R4}\right)$$

$R1 = 500\Omega$
$R2 = 500\Omega$
$R3 = 2.5K\Omega$
$R4 = 1K\Omega$

Entonces el factor de división de tensión queda:

$$\left(\frac{(R1+R2+R3+R4)}{R4}\right) = 3.5$$

Por lo tanto la tensión máxima es 3.5 x 5 Voltios, lo que equivale a 17.5 voltios.
Es decir el voltaje de la batería no debe exceder este valor ya que de hacerlo se quemaría el microcontrolador. Para prevenir esto, el circuito de conversión analógica digital contiene un diodo zener que limita la tensión de entrada a 17.5 voltios, una vez superado este valor.
El circuito resulta lo siguiente:



El voltaje en RA2 se ve afectado por las resistencias que se muestran, es decir que la precisión de la conversión depende de la exactitud de las resistencias, en este caso son de +/- 1%.

El diodo Zener no se activa hasta que la tensión de la batería llegue hasta su voltaje zener de 17.5 V, valor que se ha escogido con el propósito de proteger al microcontrolador.

Ya que el partidor de tensión tiene un factor de 3.5, para un valor mayor a 17.5 voltios en la batería, en la entrada RA2 del PIC el voltaje sería mayor a 5 voltios.

La alimentación del microcontrolador se toma de la batería, mediante el regulador LM78L05, se mantiene la tensión de salida de este en 5 voltios con un rango de variación de ± %5.
En la siguiente tabla se puede ver algunas de las características del regulador usado.

Fig. 3.13: *Entrada analógica*

Tabla 3.2: *Características del Regulador LM78L05*

Símbolo	Parámetro	Condiciones	Min	Tip	Max	Unidades
Vo	Salida	7V<Vin<20V	4.75	5	5.25	V
ΔVo	Regulación	1mA<Io<100mA		20	60	mV
Vin(Min)	Mínimo			6.7	7	V

Nota: Disipación de Potencia ≤ 0.75 W.

Si se quiere mayor detalle del regulador, referirse a la sección de anexos.
Una tensión buena en la entrada del regulador es 10 voltios; con este valor estaríamos seguros que la tensión de salida es 5V. Pero este caso no es dable ya que el voltaje en la batería varía constantemente y puede incrementar hasta 14 voltios.
Como valor permisible para este dispositivo, viendo su tabla de datos es como máximo 20 Voltios. Para seguridad se ha limitado en la tarjeta el valor máximo de entrada al regulador 15V, lógicamente esto también se hizo con un zener de 15V.

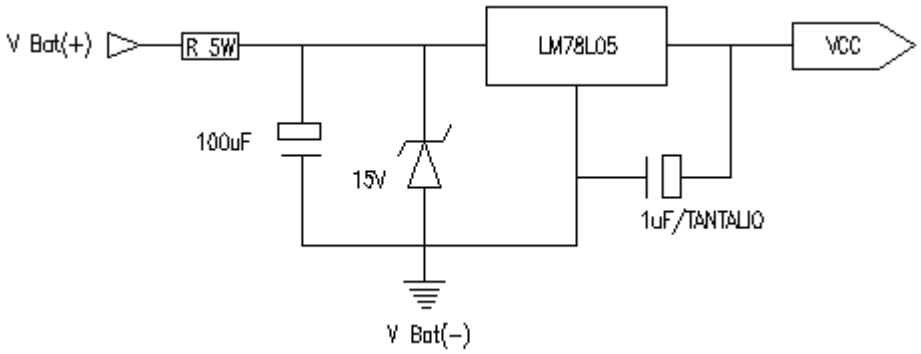


Fig. 3.14: *Protección del regulador*

3.6 Seguridad

Esta tarjeta esta pensada para un sistema fotovoltaico que usa un panel y una batería con unos límites de corriente permisibles, tanto para la batería (Descarga) como para el panel (Carga). En el caso de falla y se supere esta corriente, a modo de fusible se usan dos diodos de potencia de 6 amperios, que se queman cuando la corriente supera 6 amperios.

3.6.1 Sistema de Carga

El panel solar posee una corriente máxima, la cual de ser superada, ocurrirían daños irreparables al panel.
En el caso de un corto circuito habría mayor amperaje que el admisible pero inmediatamente el diodo se quemaría y evitaría mayor daño, se podría poner un fusible pero se estaría perdiendo unos cuantos *watts* por lo que no conviene para este sistema que trata de aprovechar al máximo la energía del sol.

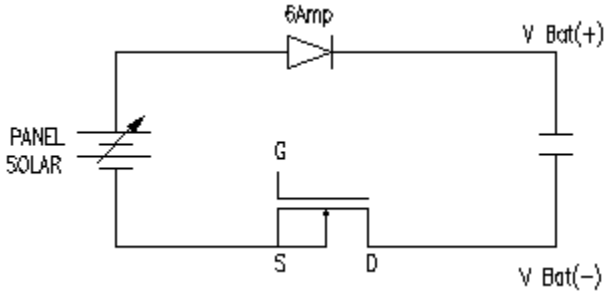


Fig. 3.15: *Diodo de protección contra corrientes mayores a 5 amperios (Sistema de Carga)*

3.6.2 Sistema de Descarga

En la descarga el que establece la corriente es el usuario o la carga que se conecte a la batería. Las baterías se miden por su capacidad que esta relacionada con el amperaje y el tiempo que se demora en descargarse entregando un determinado amperaje. Teóricamente una batería de 200 Ah, entregando 1 amperio se demoraría 200 horas en descargarse. Como se ha mencionado anteriormente la corriente depende del usuario o de la carga conectada. Aquí entra a tallar la profundidad de la descarga de la batería es decir la rapidez con que se descarga la cual se mide en amperios. Mientras la descarga es más profunda, su capacidad de carga disminuye.
En una descarga profunda en primer lugar, no se obtiene toda la energía que es capaz de proporcionar la batería. Por ejemplo una batería descargada en 72 horas devuelve aproximadamente el doble de energía que si se descargarse en sólo 8 horas. Además las descargas rápidas producen deformaciones y la prematura desintegración de las placas de los elementos, que se depositan en el fondo de los recipientes en forma pulverulenta hasta llegar a cortocircuitar ambas placas, inutilizando la batería. La rapidez da descarga se mide en amperios que absorbe el usuario de la energía. [11] Para controlar este valor lo hemos considerado como máximo de 6 amperios; por lo que un diodo de 6 amperios entre los polos positivos evitaría que el valor de la corriente sea perjudicial para la batería.

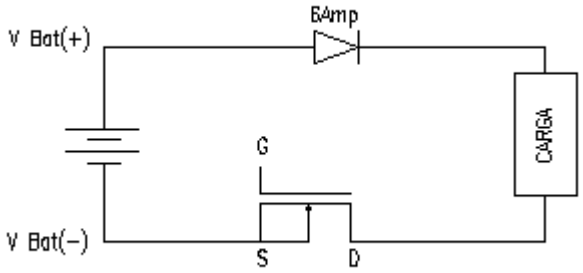


Fig. 3.16: *Diodo de protección contra corrientes mayores a 5 amperios (Sistema de Descarga)*

Es algo particular en los MOSFETS, el calentamiento, pues la potencia que disipan es directamente proporcional con la corriente que circula por ellos. Es por esta razón que se eligió el IRFZ44 ya que debido a su baja impedancia, la potencia que disipa es menor. Esta potencia disipada que en otras palabras es el calentamiento producido, va sumándose y calienta cada vez más al *mosfet.*
El *mosfet* tiene una aleta, que no es más que una superficie óptima de disipación.
Esta aleta va conectada a la caja de la tarjeta que esta al contacto con el aire.
Esto ventilará el MOSFET y evitará su sobre calentamiento.

3.6.3 Diodos de Bloqueo

La función principal de los diodos de bloqueo es la de impedir que el flujo de potencia se invierta.

[11] Existen baterías que permiten mayor descarga profunda sin disminuir su capacidad, el amperaje máximo de descarga esta fijado según el tipo de batería.

En los sistemas fotovoltaicos hay momentos de sombra, hasta días enteros en donde el panel recibe niveles bajos de luz. Puede darse el caso de que en estos momentos el voltaje del panel sea menos al de la batería, por lo que se invertiría el flujo. El diodo evitaría este cambio de dirección ya que este sólo conduce en una sola dirección: del panel hacia la batería.
De igual modo para el sistema de carga, puede ser que el usuario se conecte a una fuente más potente de energía, lo que originaría el flujo inverso entonces se estaría cargando a la batería pero no controladamente, como lo hace la tarjeta.
Recordemos que el objetivo no es proveer un sistema de carga sino controlar la carga y la descarga adecuándola a los parámetros más óptimos que alargan la vida útil de la batería.

3.7 PIC-EEPREOM

La EEPROM utilizada fue la memoria 24LC32A de 4 *kBytes* y de 8 pines, a continuación se detalla la utilidad de cada pin:

3.7.1 A0, A1, A2 Dirección del Chip: Las entradas A0.A2 son usadas por la EEPROM 24LC32A para la operación de selección múltiple del dispositivo y se conforman por un estándar *bus* de 2 líneas. Los niveles aplicados para estos pines definen el dispositivo a usar, esto es si hay más dispositivos conectados en paralelo al *bus*. Un dispositivo particular es seleccionado transmitiendo los *bits* correspondientes (A2, A1, A0) en el *byte* de control.

3.7.2 SDA Línea Bidireccional de Datos y de Direccionamiento: Éste es un pin Bidireccional usado para transferir direcciones y datos hacia o desde el dispositivo. El *hardware* de esta línea bidireccional consta de un colector abierto como terminal, por consiguiente el *bus* SDA requiere una resistencia de *pullup*[12] para VCC (10KΩ típico para 100 kHz, 2 KΩ para 400 kHz). Como la frecuencia usada para el *clock* es menor a 100KH entonces en el *hardware* he hecho el *pullup* con una resistencia de 10 KΩ.
SCL y SDA filtran el ruido y dejan pasar un entrada limpia. Esto es porque incorporan un *Schmitt triggers* [14] para 400 kHz (Modo Rápido). Es decir toda señal superpuesta con una frecuencia mayor a 400 kHz será filtrada, por lo tanto define a un filtro pasa baja cuyo límite es 400kHz.

3.7.3 SCL Reloj Serial: Esta entrada se usa para sincronizar la transferencia de datos.

3.7.4 WP: Este pin debe ser conectado a VSS o VCC.
Estando conectado a VSS, la operación normal de memoria es posibilitada (lectura y escritura de la memoria entera 000-FFF).
Estando conectado a VCC, las operaciones de escritura están inhibidas. La memoria será protegida contra escritura. Las operaciones de lectura no son afectadas.

[12] Mayor información del *pullup* para colector abierto, la puede encontrar en la página www.semiconductors.com de *philips*.

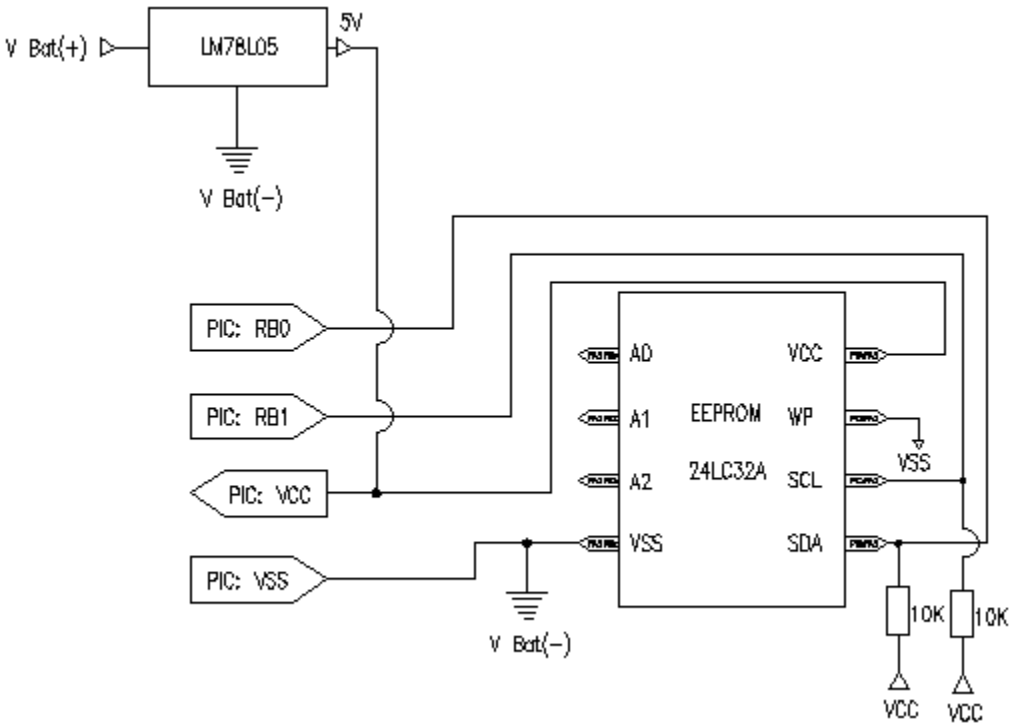


Fig. 3.17: *Hardware para la comunicación I2C entre el PIC y la EEPROM*

3.8 EEPROM - PC

Para transmitir los datos a la PC, la manera más fácil, menos costosa y rápida, es hacer la comunicación directamente entre la EEPROM y la PC.
Sería engorroso, hacerlo con el método tradicional que es el de leer los datos de la EEPROM con el PIC e ir mandándolos a la PC con el respectivo acondicionamiento de señal mediante el protocolo RS-232. Esto implica costo del *software* necesario para que el PIC lea los datos de la EEPROM y mandarlos a la PC, lógicamente en respuesta a la petición de esta
Adicionalmente a esto está el programa en la PC, para manejar una interfase que te permita tener control sobre el puerto serial.
Para una comunicación directa entre la PC y la EEPROM, sólo necesitaríamos lo último mencionado en *software*, es decir el control del puerto de la PC para poder hacer una petición de datos a la EEPROM y para poder mediante un *buffer* almacenar los *bytes* leídos. El *software* implementado será explicado con mayor detalle.
Los pines del puerto serial usados para la comunicación son:

Pin 5 (CND); Línea de tierra, esta va conectada a la tierra de la EEPROM.
Pin 7 (RTS): Línea de *Clock* generado por el *master* (PC).
Pin 8 (CTS): Línea de datos (SDA) que se leen del esclavo (EEPROM.).
Pin 6 (DSR): Línea de datos que se mandan al esclavo (EEPROM).

Los pines de salida del puerto serial, manejan niveles de tensión de ± 12V, entonces no podemos conectar la PC a la EEPROM sino se regula dicha tensión.

Para dicha regulación se usa las líneas RTS y DSR como entrada a un MAX 232[13], cuya característica es acoplar el buen funcionamiento de protocolos como el RS 232 y el I2C.

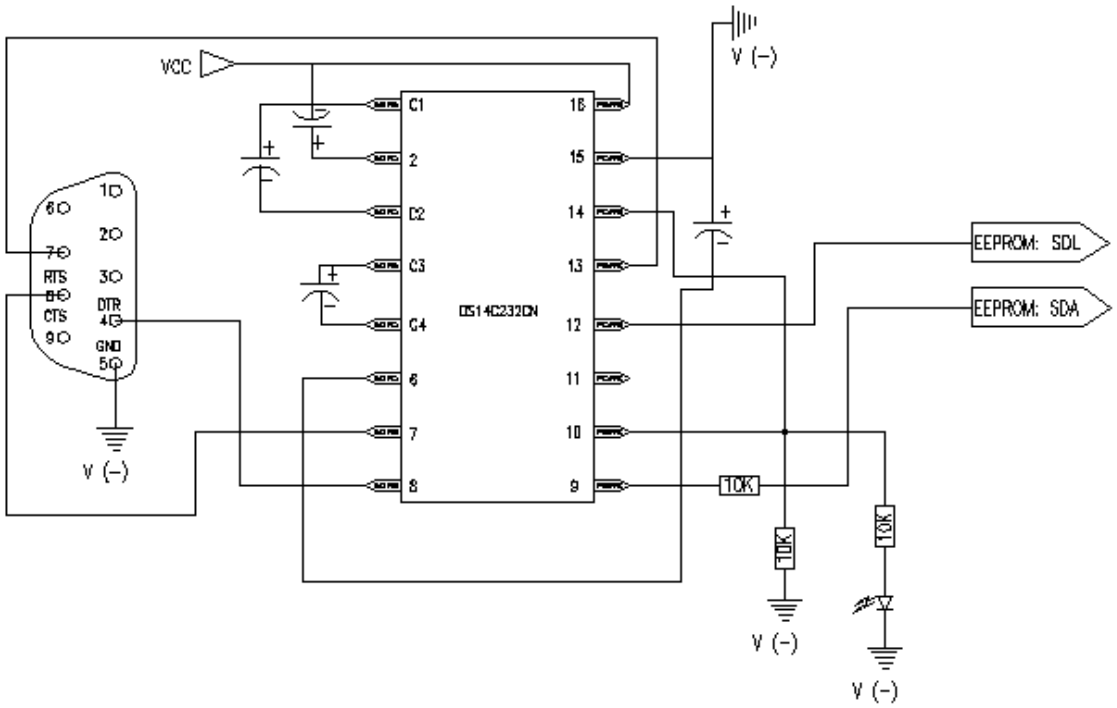


Fig. 3.18: *Hardware para la comunicación I2C entre la EEPROM y la PC*

La línea SDA es una línea bidireccional, la EEPROM recibe datos que la computadora manda por el Pin6 (DSR). Pero la EEPROM mando datos también por la línea SDA y la computadora los recibe por el Pin6 (CTS).
Cuando la EEPROM este mandando datos, DSR estará o bien en 1 o en 0 lógico, a la vez los datos mandados fluctúan constantemente entre 0 y 1 lógico. La razón del diodo es evitar que la tensión sea inestable ya que en la misma línea ambos terminales estarán a diferentes tensiones.
Los esquemas generales (Planos) se encuentran en el apéndice A.

[13] En la presente tesis se usa como MAX 232, al *chip* DS14C232CN.

CAPÍTULO-4

DESCRIPCIÓN DEL SOFTWARE DEL SISTEMA PROPUESTO

El presente capítulo tiene como objetivo:

Una descripción detallada del regulador implementado, en cuanto a *software*.

Se verá en detalle la implementación del protocolo para la comunicación del PIC con la tarjeta de adquisición de datos. Así mismo se estudiará la comunicación entre la PC y la tarjeta de adquisición de dato implementada.

Se verá los diagramas de bloques del *software* implementado, la planificación de tareas comunes que el microcontrolador debe realizar en un mismo tiempo (*Real Time Kernel*).

El siguiente estudio se dividirá en 3 etapas:

4.1 Regulación.
4.2 Comunicación Regulador – EEPROM.
4.3 Comunicación PC – EEPROM.

4.1 Regulación.

La regulación que se haga debe dar al sistema el perfil siguiente:

Tabla 4.1: *Perfil de Regulación*

Tensión 12V	***Batería***
Reconexión de consumo	11V
Desconexión de consumo	10V
Banda de Igualación	13.5 – 13.55V
Carga Profunda	13.5V
Banda de Flotación	11.0 – 13.5V
Alarma de baja	10.5V
Alarma de Alta	13.5V
Rearme de carga profunda	13.5V

Existen baterías más resistentes a las descargas que otras y por lo tanto diferentes en sus niveles óptimos de tensión permisibles. Los niveles mencionados en el cuadro son ajustables por *software* al requerimiento del sistema, en esto radica la flexibilidad de la tarjeta. El microcontrolador nos ofrece en su uso varias bondades como son:
Conversión A/D, un temporizador y varios servicios de interrupciones. Para implementar el programa se ha explotado estas bondades que bajo la configuración de sus respectivos registros de control, se pueden acoplar perfectamente al programa que sigue una secuencia estructurada.

4.1.1 Conversión A/D.

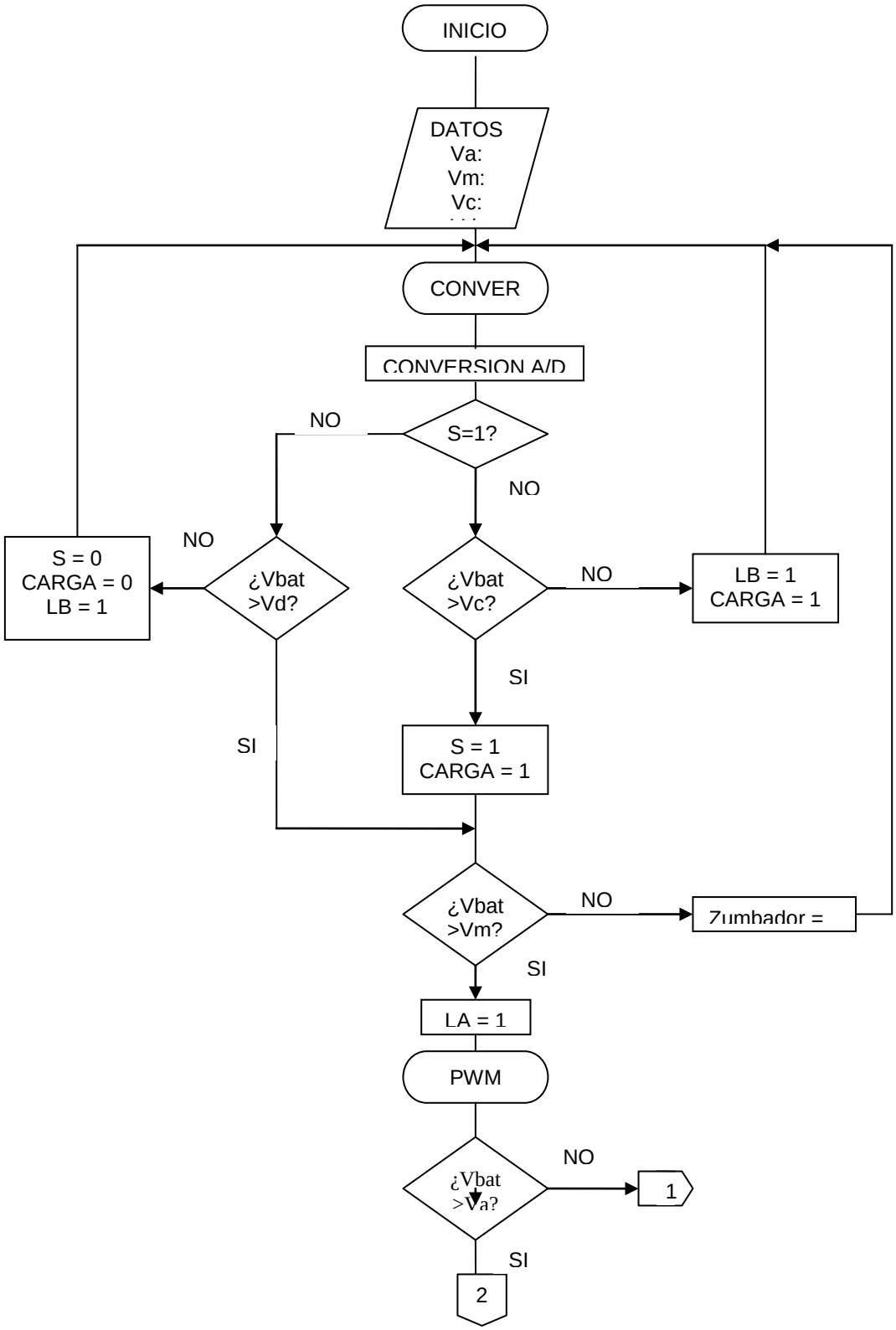
Como el nombre lo dice, se usa para obtener instante a instante el valor del voltaje de la batería y de esta manera ver si esta dentro de los valores predeterminados en el cuadro de perfil que inicialmente se dieron como datos en el programa.

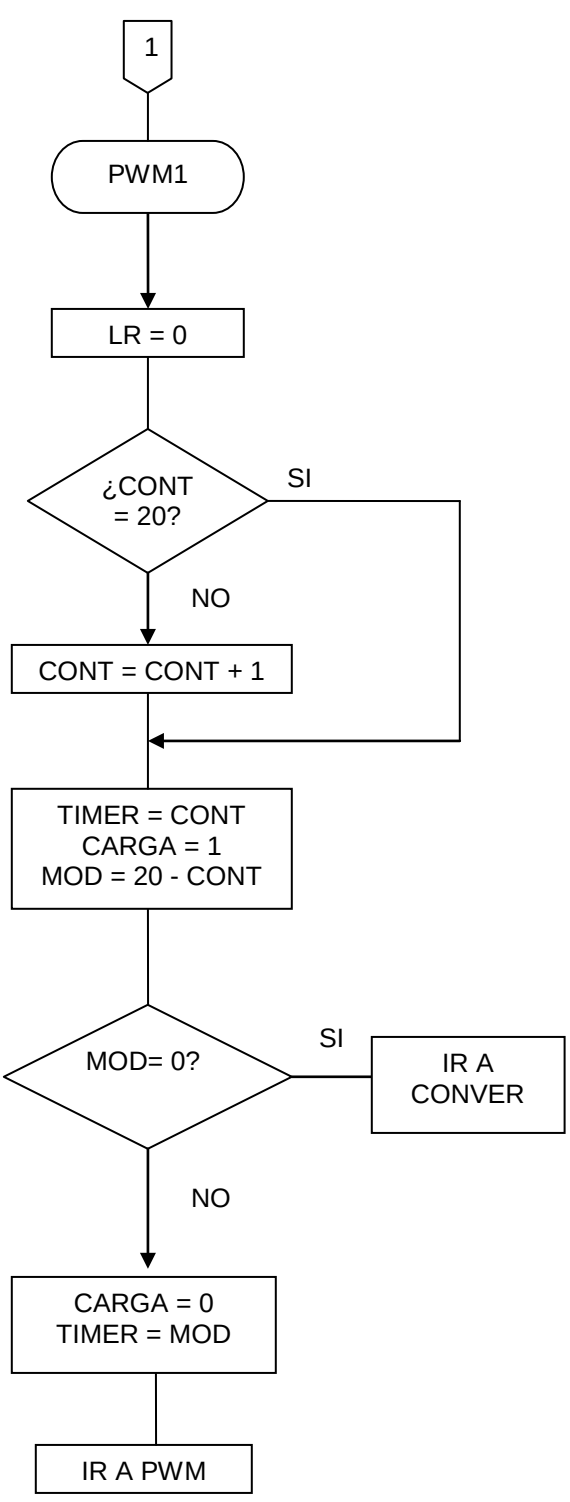
4.1.2 Temporizador e Interrupciones:

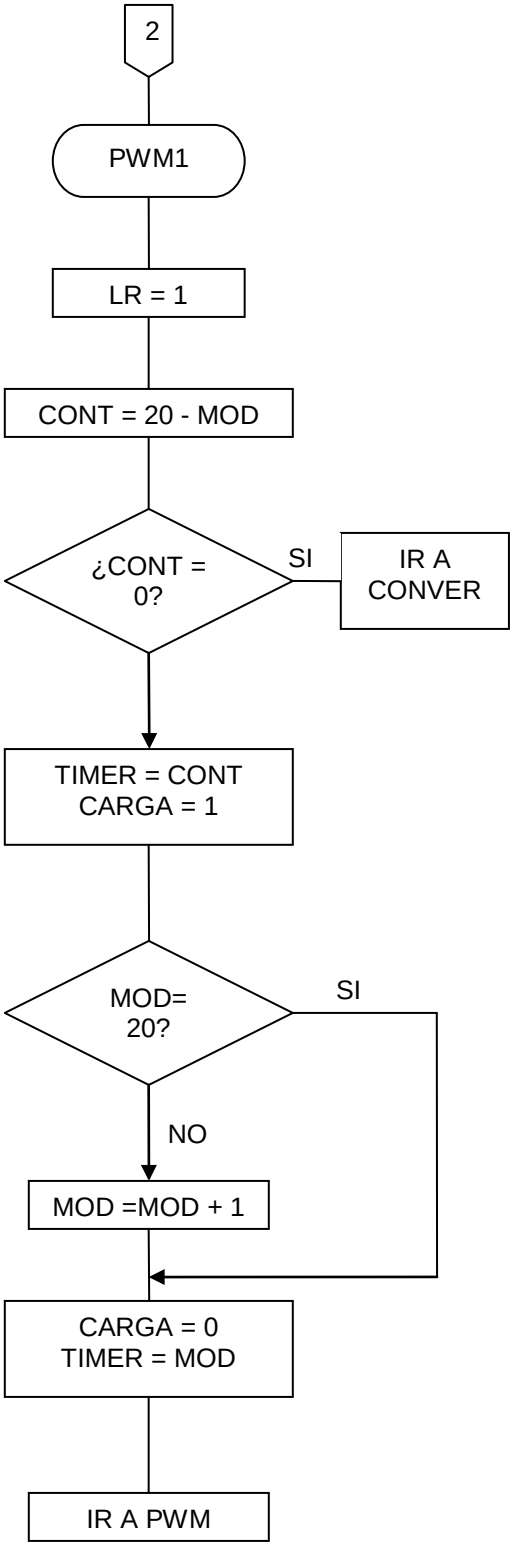
Se usa el servicio de interrupciones que se activa al llenarse el byte del temporizador, esto en el programa sucede cada 50 ms, la cual pasado este tiempo, el puntero del programa va a una dirección (0x004) fija y preestablecida por el propio microcontrolador, en esta dirección se debe hacer la rutina que se quiere cumplir cada vez que suceda una interrupción.
Lo mencionado en el cuadro son las características que definen la buena regulación del sistema.
El *software* implementado debe ser el indicado para poder asegurar dichos valores. El microcontrolador se programa con lenguaje ensamblador (Ver programa en anexos).

Como todo programa, este cumple un diagrama de flujo, que se muestra a continuación.

DIAGARMA DE FLUJO DE LA REGULACION




1
PWM1
LR = 0
¿CONT = 20?
SI
NO
CONT = CONT + 1
TIMER = CONT
CARGA = 1
MOD = 20 - CONT
MOD= 0?
SI
IR A CONVER
NO
CARGA = 0
TIMER = MOD
IR A PWM


2
PWM1
LR = 1
CONT = 20 - MOD
¿CONT = 0?
SI
IR A CONVER
TIMER = CONT
CARGA = 1
MOD= 20?
SI
NO
MOD =MOD + 1
CARGA = 0
TIMER = MOD
IR A PWM

4.2 Comunicación Regulador – EEPROM:

La comunicación entre el PIC y la EEPROM se realiza bajo el protocolo I2C, protocolo que creo la *philips*[14] para estandarizar un tipo de comunicación entre sus dispositivos integrados.
Para poder entender el programa que esta en la sección de anexos de este estudio, hay que entender detalladamente en que consiste el protocolo de comunicación I2C.

4.2.1 Descripción Funcional. El protocolo I2C usa un bus de datos bidireccional, de recepción y de transmisión de datos. Un dispositivo que envía datos por el bus está definido como transmisor, y el dispositivo que recibe datos como aparato receptor. El bus debe controlarse por un dispositivo maestro que genera el *Clock* Serial (SCL), controla la vía de entrada del bus, y genera el PRINCIPIO y EL FIN de acuerdo a algunas condiciones, mientras el receptor funciona como esclavo. Ambos maestro y esclavo pueden operar como el transmisor o el aparato receptor pero el maestro es el dispositivo que determina que modo es activado.

4.2.2. Características del Bus de Datos. El siguiente protocolo de bus haya estado definido:
• La transferencia de datos puede ser iniciada sólo cuando el bus no está ocupado.
• Durante la transferencia de datos, la línea de datos debe permanecer estable cuando quiera que la línea del reloj esté ALTO. Cambiar en la línea de datos mientras la línea del reloj está ALTO será interpretado como una condición de PRINCIPIO o de PARADA.
Consecuentemente, las siguientes condiciones del bus han estado definidas (la Figura 3-1).

4.2.3 Bus no Ocupado (A). Ambos línea de datos y línea del reloj permanecen en nivel ALTO.

4.2.4 Iniciar Transferencia (B). Una en la línea de datos (SDA) mientras el reloj (SCL) está ALTO determina una condición de PRINCIPIO (START). Todas las órdenes deben ser precedidas por una condición de PRINCIPIO.

4.2.5 Detener Transferencia de Datos (C). Un cambio de nivel de bajo a alto en la línea SDA mientras el reloj (SCL) está ALTO determina una condición de PARADA. Todas las operaciones deben ser acabadas con una condición de PARADA.

4.2.6 Datos Validos (D). El estado de la línea de datos representa datos válidos cuándo, después una condición de INICIO, la línea de datos es estable para la duración del período ALTO de la señal del reloj.
Los datos en la línea deben variarse durante el período BAJO de la señal del reloj. Hay un pulso del reloj por cada bit de datos.
Cada transferencia de datos es iniciada con una condición de INICIO y terminada con una condición de ALTO. El número de los bytes transferidos entre INICIO y PARADA es determinado por el dispositivo maestro.

4.2.7 Reconocimiento (ACK). Cada dispositivo receptor, cuando son direccionados, tienen la obligación de generar una señal de reconocimiento después de la recepción de cada byte. El dispositivo maestro debe generar un pulso adicional del reloj que se asocie con el bit de reconocimiento. Un dispositivo haga el reconocimiento debe poner en nivel bajo la línea SDA

durante el pulso del reloj de tal manera que la línea SDA esta en nivel BAJO estable durante el período ALTO del pulso del reloj. Por supuesto, el los tiempos de secuencia entre onda de SDA y onda de SCL deben tomarse en cuenta y estos tiempos dependen de las especificaciones de ambos; maestro y esclavo. Durante la operación de de lectura de datos, un Maestro debe dar señas de un fin de datos para el esclavo y esto lo hace al NO generar un bit de reconocimiento después del último byte. En este caso, el esclavo (24LC32A) dejará la línea de datos en ALTO y esto le facultaría al *Master* generar la condición de PARADA.

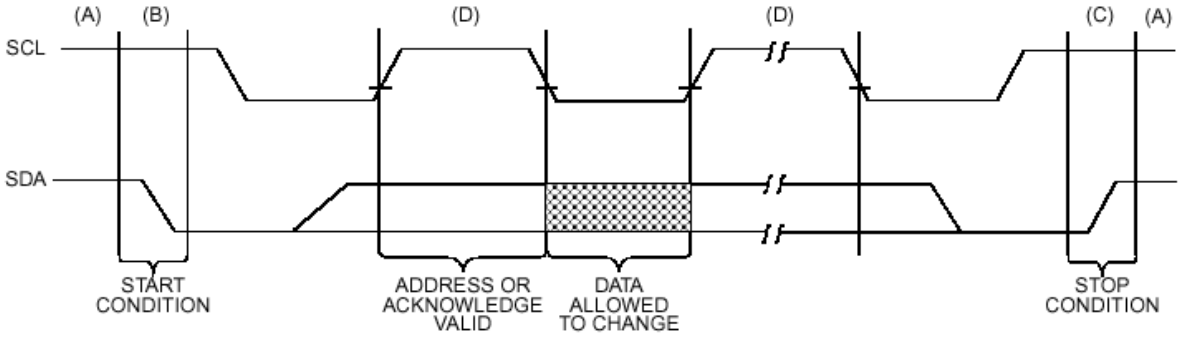


Fig. 4.1: *Transferencia de datos en el Bus serial*

4.2.8 Direccionamiento del Dispositivo.

El *Byte* de control es el primer *byte* admitido después de la condición de INICIO del dispositivo maestro. El *Byte* de control consta de un código de control de 4 *bits*. Los siguientes tres *bits* del *byte* de control son los *bits* de selección de dispositivo, ellos son usados por el dispositivo maestro para seleccionar uno de los 8 esclavos con el que desea entablar la comunicación, haciendo de una respectiva selección de los *bits* (A2, A1, A0).Estos en el esclavo están conectados físicamente o a tierra o a Vcc. El último *bit* del *byte* de control define la operación a ser realizado. Cuando es seteado a uno, se selecciona una operación de lectura, y cuando es seteado a cero se selecciona una operación de escritura. Los siguientes dos bytes recibidos definen la dirección del puntero de la memoria a leer o escribir en el esclavo del primer *byte* de datos (la Figura 3-3).

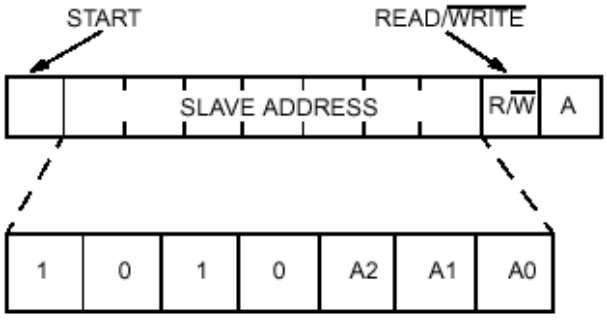


Fig. 4.2: *Control Byte*

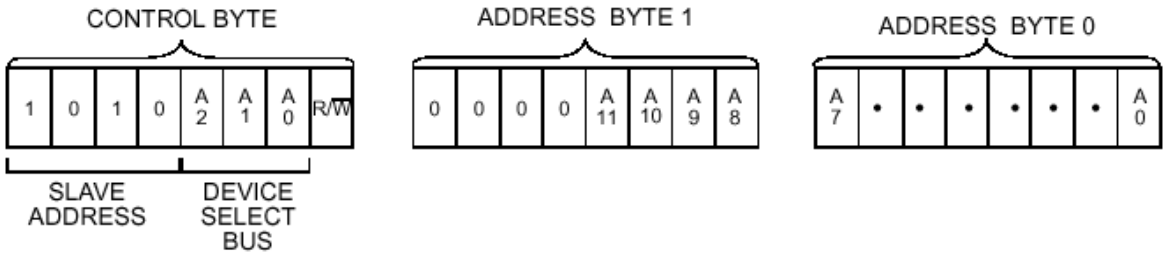


Fig. 4.3: *Byte de Control y de Direccionamiento para el caso un memoria de 4Kbytes*

4.2.9 Operación de Escritura

a) Escribiendo un *Byte*. Siguiendo la condición de INICIO dada por el master, el código de control (cuatro *bits*), el dispositivo selecto (tres *bits*), el *bit* R/W debe ser puesto a nivel lógico bajo por el *master*. Esto indica al dispositivo receptor que un *byte* de dirección seguirá después de que él haya generado un bit de reconocimiento (ACK) durante el noveno ciclo de reloj. Por consiguiente, el siguiente *byte* transmitido por el Master es el segundo *byte* de la dirección esto ubicará un puntero en la memoria del esclavo. El siguiente *byte* es el *byte* de la dirección menos significante. Después de recibir la señal de reconocimiento el dispositivo maestro transmitirá el *byte* de datos que desea escribir en la dirección de memoria que apunto.
El esclavo manda un *bit* de reconocimiento de nuevo (ACK) y el master genera una condición de PARADA. Esto inicia un ciclo de escritura interna dentro del esclavo, y durante este tiempo el esclavo no generará un reconocimiento.

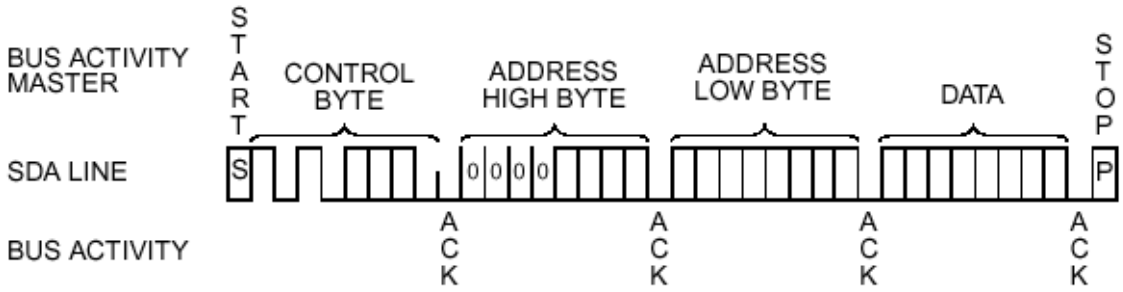


Fig. 4.4: *Escribiendo un Byte*

b) Escribiendo una Página. El *byte* de control, la dirección de palabra y el primer *byte* de datos son transmitidos al esclavo de la misma manera que al escribir un *byte*. Pero en lugar de generar una condición de PARADA, el *Master* transmite hasta n *bytes* que son temporalmente almacenados *buffer* del esclavo y estará escrito en la memoria después de que el *Master* haya transmitido una condición de PARADA. Después de reciba cada bloque, los cinco *bits* menos significativos del puntero de la dirección son internamente incrementados en uno.
La cantidad n de *bytes* secuenciales a escribir depende de la capacidad del *buffer* del dispositivo receptor. Sí el *Master* transmite más de 32 *bytes* antes de generar la condición de PARADA, los datos previamente recibidos serán sobre-escritos. Al igual que con al escribir un *byte*, una vez que la condición de PARADA es recibida, un interno ciclo de escritura comenzará.

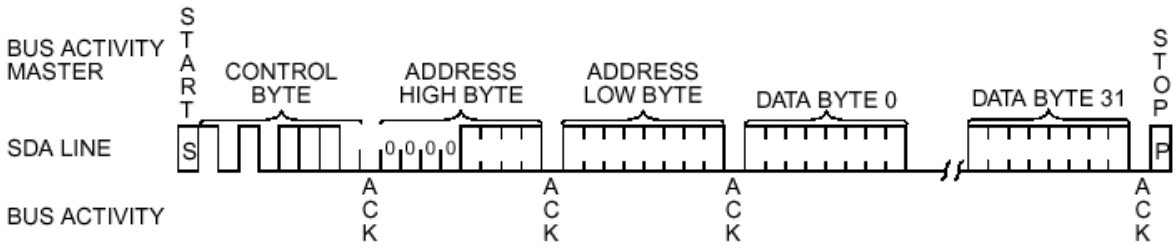


Fig. 4.5: Escribiendo una página de 32 bytes

c) Espera por el *Bit* de Reconocimiento. Desde que el dispositivo no envía un *bit* de reconocimiento podemos usar esto para determinar cuando el ciclo esta completo. Recordemos que una vez que se genera la condición de PARADA, el esclavo empezará su ciclo de escritura interna. Luego el *master* para iniciar otra comunicación manda la condición de INICIO y el control *byte*, luego se hace un *Polling* (ACK) para saber sí el esclavo ya terminó su ciclo interno de escritura. Esto implica al *Master* enviar una condición INICIO seguida por el *byte* de control para escribir (R/W = 0). Si el esclavo está todavía ocupado, entonces NO ACK será devuelto. Si el ciclo es completo, entonces el dispositivo devolverá ACK y el *Master* puede otra vez entablar comunicación.

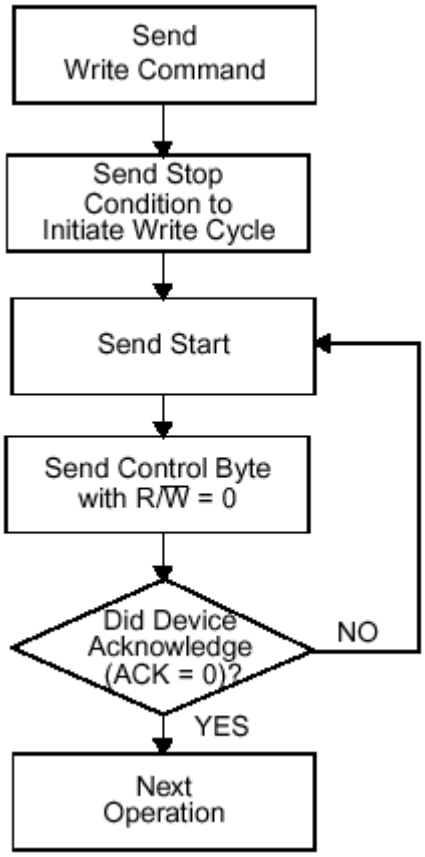


Fig. 4.6: *Diagrama de flujo para el Polling del ACK*

4.2.10 Operación de Escritura. Las operaciones de lectura son iniciadas de la misma manera que las operaciones de escritura a excepción del *bit* R/W que es 1. Hay tres tipos básicos de lectura: Leer la dirección actual, leer al azar, lectura secuencial.

a) Lectura de la dirección Actual del Puntero de la Memoria. Las memorias EEPROM contienen un contador de direcciones que mantiene la dirección de la última palabra accedida +1. Por consiguiente, si la vía de entrada previa (ya sea una operación de lectura o escritura) fue dirigida a la palabra n (n es cualquier dirección legal), entonces la siguiente dirección actual de datos será la dirección n + 1. Para obtener este dato de la siguiente posición de la memoria a ocupar (n+1), se debe de leer como se explicó anteriormente, después del *control byte* de escritura, se debe esperar el reconocimiento del esclavo y luego se debe recibir el dato de la dirección (n+1) en un *array* de 8 caracteres.
Luego el *Master* no debe hacer reconocimiento y esto generará una condición de PARADA.

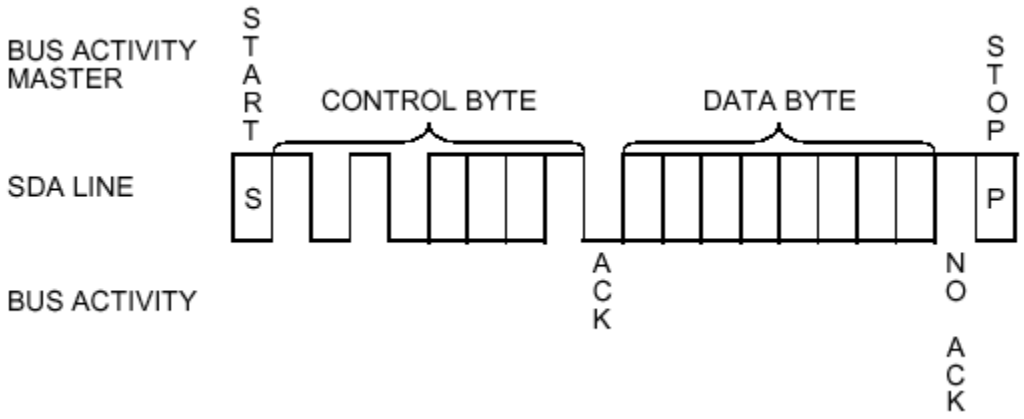


Fig. 4.7: *Lectura de la dirección actual*

b) Lectura al Azar y Secuencial. Las operaciones de lectura al azar permiten que el master acceda para leer a cualquier posición de la memoria de manera aleatoria. Para realizar este tipo de operación, primero debe estar colocada la dirección del dato a leer. Esto se realiza mandando la dirección que se desea leer en una operación de escritura (R/w= 0). Después que la dirección es mandada, después del reconocimiento, el *Master* debe generar un *bit* de INICIO, esto termina la operación de escritura pero no antes de que el puntero se haya colocado en la dirección deseada.
Luego el *Master* envía el *byte* de control una vez más pero con el R/W= 1, esto generará la condición de escritura y el esclavo generará un bit de reconocimiento y luego de esto el esclavo mandará el *byte* de datos. Sí se desea sólo leer un dato el *master* debe de enviar un *bit* de no reconocimiento poniendo la línea SDA a 1, esto generará la condición de PARADA (lectura aleatoria). Pero sí se desea leer más datos en forma secuencial, se puede seguir con el respectivo reconocimiento (lectura secuencial).

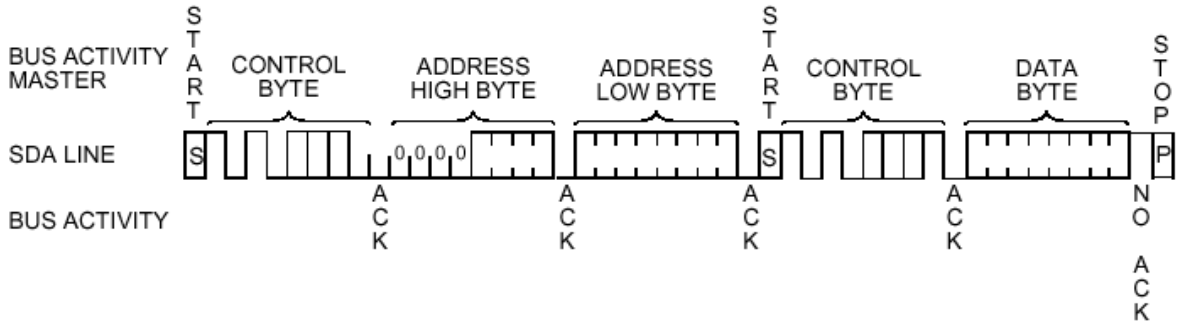


Fig. 4.8: *Lectura Aleatoria*

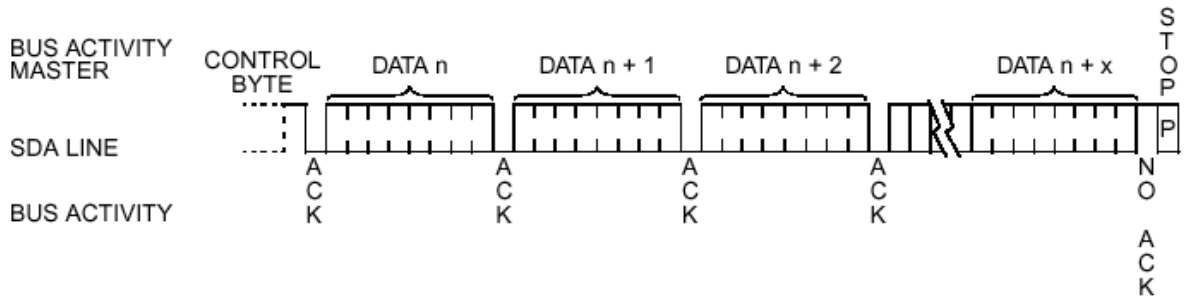


Fig. 4.9: *Lectura Secuencial*

Implementación de la comunicación.
Para realizar esta comunicación, centrar nuestra atención en 8 pequeñas rutinas que conjuntamente ejecutadas desde un programa principal, permitirán realizar una excelente comunicación I2C entre el PIC 16c711 y la memoria EEPROM 24LC32A.

4.2.11 Rutinas.

Abría que definir unas subrutinas de llamada, que están conformadas por las ejecuciones por bloque que están separados por *bits* de reconocimiento ACK o NO ACK.

1) START BIT: *Bit* de inicio (S).
2) CONTROL BYTE: 7-*Bit* de direccionamiento (*Slave Address*) + 1 *Bit* de Lectura (0) o escritura (1).
3) ACK: *Bit* de reconocimiento (A).
4) NO ACK: *Bit* de no reconocimiento (NA)
5) DBR (*Data Byte Reception*): Este es el dato de la dirección de la última palabra accedida + 1 que se lee de la memoria para proseguir con el llenado sin reemplazar los datos existentes.
6) AHB & ALB: Son los dos *bytes* de direccionamiento de la memoria, entre los dos tiene que existir un ACK para el reconocimiento.
7) DBS (*Data Byte Send*): Mensaje o dato octeto que se quiere mandar, pueden ser desde 1 hasta 32 *bytes* sin *bit* de reconocimiento.
8) STOP BIT: *Bit* de parada (P).

Los pines que se usarán son RB0 y RB1, como SCL y SDA respectivamente.

$RB0 \Rightarrow SDA$

$RB1 \Rightarrow SCL$

Esto se declara al comienzo de todo el programa. Antes del ORG 0.
La dirección de la memoria la tenemos que tener en cuenta, debido a que existe riesgo de que se sobre escriban los datos. Como la memoria EEPROM del 24LC32A es de 4K x 8Bits, entonces necesitamos 11 *bits*, para poder direccionarla. Por lo tanto tenemos que escoger 2 *Bytes* de la memoria RAM del PIC, para poder tener 11 *bits,* de hecho habrá un *Byte* en el que sólo se escribirán los 3 *bits* menos significativos.

En la memoria EEPROM sólo se pueden escribir 4 x 1024 *bytes*. Cada *Byte* va a representar una tensión que se almacenará cada minuto que este conectada la carga, es decir lo máximo de datos que podríamos almacenar antes de limpiar la RAM son 4096 datos, que vienen a ser datos de voltios en 4096 minutos, suponiendo que el usuario esté conectado por 14 horas al día, entonces se podrían almacenar valores de tensión durante 5 días. Este *Data Loger* se usaría para determinar el comportamiento de la batería y del medio en donde se localiza el panel, se puede hacer una relación directa entre la potencia consumida y la potencia presente en la radiación mediante el factor eficiencia del panel y de esta manera determinar la radiación directa en esa zona.

Tabla 4.2: *Niveles de Tensión en que la que la batería esta en constante descarga*

	Bajo 10 o 11 voltios	sobre13.5 voltios
Mosfet panel-batería	Activo	No Activo
Mosfet batería-carga	No Activo	Activo

Cuando empieza el día y la batería se carga hasta llegar a 11 voltios, se activa la carga pero cuando el voltaje de la batería es menor a 10 voltios, la carga se desconecta del sistema. Esto nos indicaría las horas de trabajo de la batería[14].
Se podría plantear en código lo siguiente.

a) Condición de inicio
Cuando el voltaje de la batería es 10.9 voltios y el *mosfet* de batería-carga está ***desactivado***, quiere decir que recién comienza el día y la batería ha empezado a trabajar, claro el error sería el tiempo que ha pasado para que la batería llegue de 10.9 a 11 voltios, tiempo que es corto.

b) Condición de parada
Cuando el voltaje de la batería es 10.1 y el *mosfet* de batería-carga está ***activado***, quiere decir que la carga está a punto de ser desconectada. Estaríamos en fin del trabajo de la batería, pues una vez desconectada la carga, la batería tendría que cargarse a 11 voltios para que la carga se vuelva a ser conectada, lo que representaría el comienzo de otro día.

[14] Los límites de conexión y desconexión de la carga pueden ser cambiados por *software*.

Tener el control de las horas de trabajo de la batería es como si se llevara un control de las horas de sol utilizadas, en función de la capacidad de la batería Ah. Se deduce esto porque así la batería se quede cargada hasta la noche en donde naturalmente no hay horas de sol, la carga que posee vino del sol, lo cual indirectamente, es su energía pero ahora medida en capacidad del que la almacena, Ah.

Por otro lado, tener un registro de la descarga por día Sol de la batería serviría para ver que tan profunda fue la descarga; cuanto más rápida es la descarga, menor es el tiempo de duración de la carga de la batería por día. Una mayor descarga impedirá una mayor carga de la batería, esto es perjudicial y poco a poco se podría ir disminuyendo la capacidad de carga de la batería hasta quedar obsoleta.

Para llevar el control del voltaje y del tiempo se configuró que se haga una interrupción a la ejecución del programa cada 50ms, esta interrupción manda al puntero del programa a una posición determinada por el microcontrolador en donde hay una subrutina que permitirá ir sumando este valor del tiempo y así se irá contando hasta llegar a un minuto.

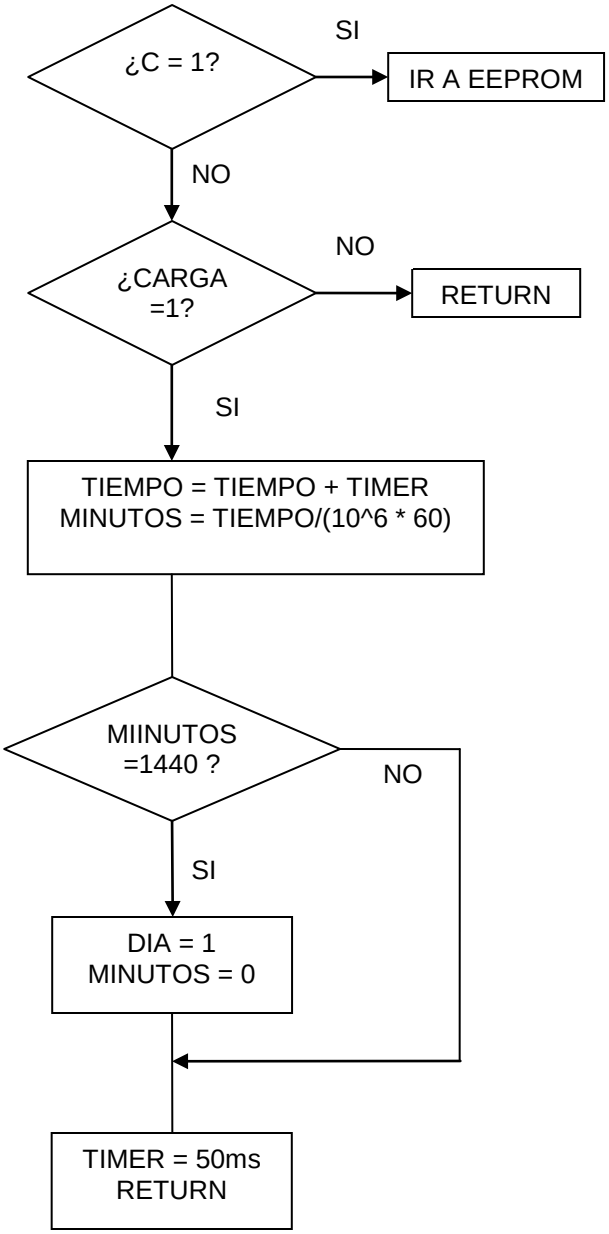
De esta manera constantemente se están ejecutando las siguientes tareas al mismo tiempo. La conversión, la regulación, la comunicación y la activación de los indicadores. Son varias tareas al mismo tiempo, esto es lo que forma la base de los sistemas operativos y se le denomina *Real Time Kerne*[15]*l*.

Con estas premisas se elaboró el programa que se basa en el diagrama de bloques que se muestra en la siguiente página.

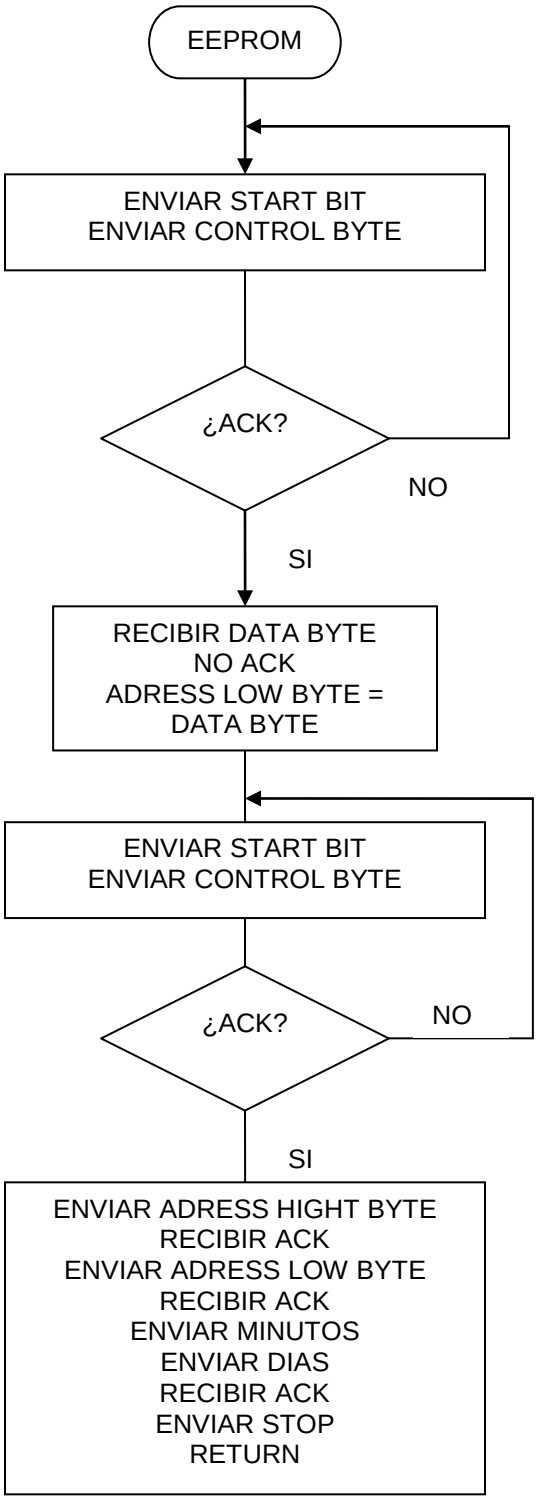
[15] Para más información leer cualquier bibliografía de sistemas operativos.

DIAGRAMA DE FLUJO DE LA TEMPORIZACION DE HORAS DE DESCARGA Y COMUNICACIÓN ENTRE EL PIC Y LA MEMORIA EEPROM

TEMPORIZADOR DE DESCARGA



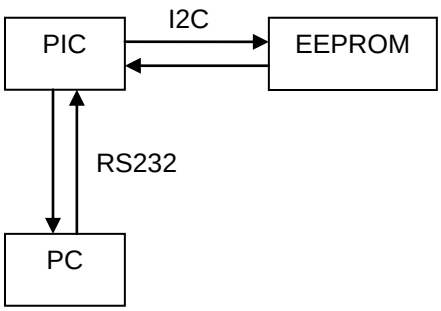
PIC - EEPROM



4.3 Comunicación PC – EEPROM.

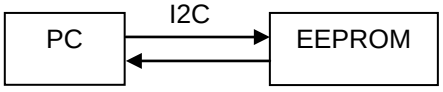
La abstracción de datos de la PC puede ser de la siguiente manera:

4.3.1 Protocolo RS232:



Esta comunicación tiene dos etapas, en la primera el PIC es *Master* y la EEPROM, de esta manera abstrae los datos guardados. En la segunda la PC es *Master* y el PIC es esclavo. Esta comunicación sugeriría un mayor gasto computacional ya que abría que programar los dos protocolos en el PIC. También nos obliga a llevar una PC portátil al lugar de la regulación ya que no puede ser lo contrario, debido a que la tarjeta se encuentra constantemente regulando.

4.3.2 Protocolo I2C:



Con este protocolo sólo se comunicarían la PC con la EEPROM, usando la PC como *Master* y la EEPROM como esclavo. Sólo se programaría en la PC para la abstracción de los datos bajo el protocolo I2C.

Por la opción es usando I2C entre PC y EEPROM, otra ventaja es que esto evitaría llevar una portátil al lugar del regulador; es por esto que el *data loger*, esta en una tarjeta separada extraíble del regulador, para así poder trasladarla con facilidad.
Este *data loger*, como hemos visto en capítulo anterior, tiene dos puertos de comunicación, para poderla conectar al regulador y a la PC cuando ase quieran conectar los datos.
Como convertidor de protocolos en niveles de tensión se usa un MAX 232:

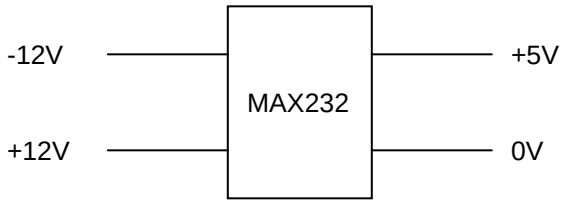
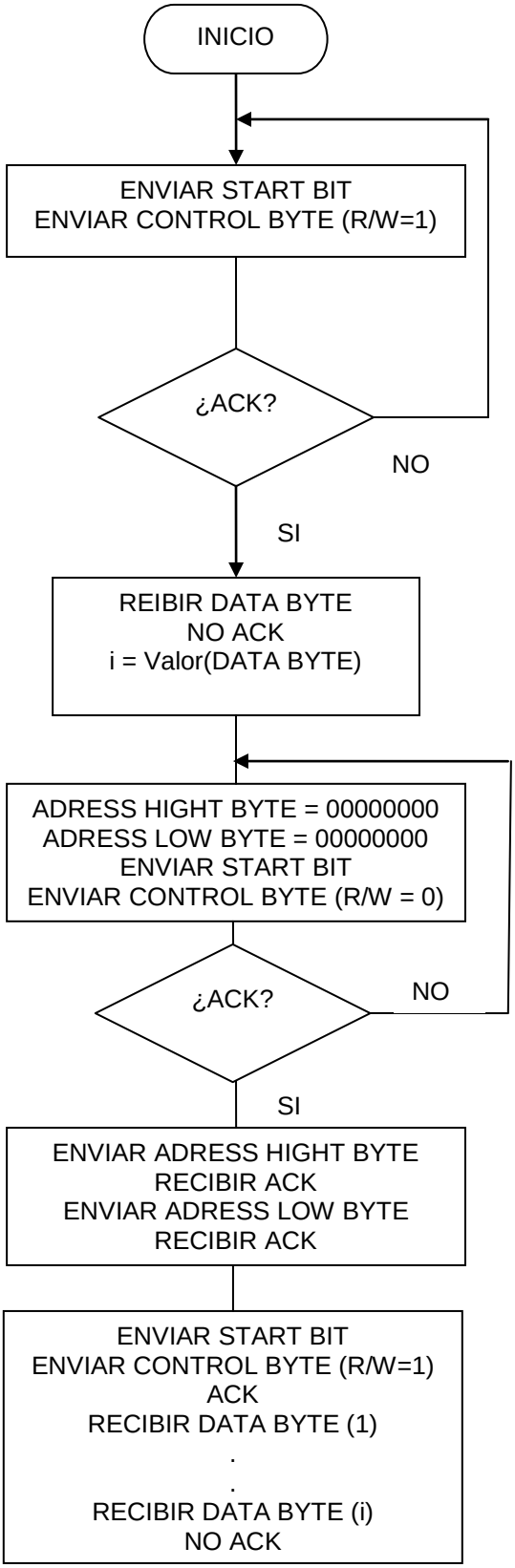


Fig. 4.10: *Cambio de niveles de tensión mediante un Max 232*

El programa de interfase se hizo en Visual C++ (ver Apéndice C). En la siguiente página se muestra el diagrama de flujo del programa:
En el siguiente capítulo se verán los resultados del *hardware* y *software* implementados del regulador y de la tarjeta de adquisición de datos.

DIAGRAMA DE BLOQUE DE LA COMUNICACIÓN ENTRE LA PC Y LA MEMORIA EEPROM



CAPÍTULO-V

PRUEBAS, RESULTADOS Y EVALUACIÓN DE DATOS ADQUIRIDOS

El presente capítulo tiene como objetivo:

Una muestra del funcionamiento de las tarjetas implementadas en base a la teoría antes expuesta.

Resultados para diferentes niveles de tensión de la batería simulada por una fuente de tensión variable, muestra de la regulación y de la comunicación.

Una evaluación de los datos almacenados.

El siguiente estudio se dividirá en 3 etapas:

5.1 Implementación del Regulador de tensión y del *Data Loger*.
5.2 Pruebas y Resultados.
5.3 Evaluación de los datos adquiridos.

5.1 Implementación del Regulador de tensión y del *Data Loger*: El *Hardware* mencionado en el capítulo 3, se plasmó en las tarjetas que vemos en la Fig. 5.1 y 5.2.

Fig. 5.1: *Regulador de Tensión internamente*

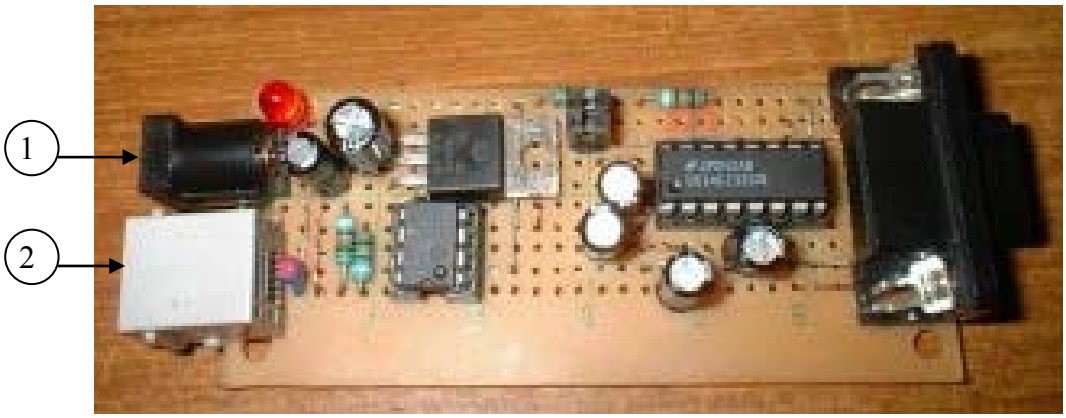


Fig. 5.2: *Tarjeta de adquisición datos (Data Loger)*

Como se puede apreciar en ambas figuras; las tarjetas tienen varias salidas entradas:

El regulador posee 6 pines, cada par de ellos corresponden al usuario (Fig. 5.1 (1)) salida, a la batería (Fig. 5.1 (2)) y al panel (Fig. 5.1 (3)) como salida.
También posee una salida de comunicación I2C (Fig. 5.1 (4)) y otra salida de alimentación para el *data loger* (Fig. 5.1 (5)) desde la batería.
Estas dos salidas del regulador se conectan una a una con las entradas de la tarjeta de adquisición de datos (Fig. 5.2 (1 y 2)) y la comunicación con la PC se realiza a través del conector DB 9 (Fig. 5.2 (3)).
La especificación para cada salida y entrada se indicará en el case de cada tarjeta.
Conexión entre ambas tarjetas:

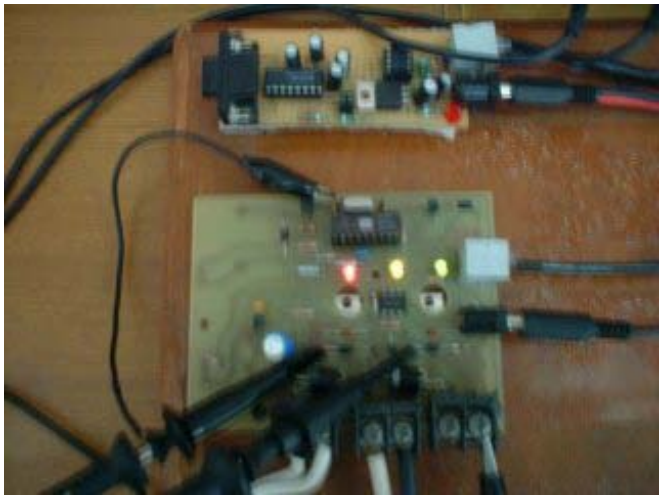


Fig. 5.3: *Conexión entre el regulador y la tarjeta de adquisición.*

El regulador enviará datos que se quieran almacenar como por ejemplo horas de descarga de la batería y se almacenarán en la EEPROM del *data loger*; estos datos serán descargados por el puerto serial a la computadora.

Fig. 5.4: *Conexión para descarga de datos de la tarjeta de adquisición con la PC.*

Para la realización de la descarga se necesita la tarjeta y el *software* Panel I2C, ya que la comunicación entre la PC y la tarjeta no es RS 232 sino es I2C, este *software* se hizo en Visual C++[16] y permite leer y escribir en la memoria. En el apéndice A esta el código fuente principal de dicho *software*; también se dará el programa y su compilación en el CD adjunto a esta tesis.

Para las respectivas pruebas, se simuló la batería como una fuente de tensión variable, esta se conecto a la tarjeta y se simuló el comportamiento de un día completo de trabajo del sistema:

Tabla 5.1: *Relación de tareas que el regulador ejecuta de acuerdo al estado de la batería*

Vbat (V)	**Indicador (Led)**	**Acción (Estado)**	**Panel - Bat**	**Bat - Usuario**
9 – 11V	Led: Verde	Sin Carga	Activado	Desactivado
11 – 11.45	Led: Ambar	Carga baja	Activado	Activado
11.6 – 13.45	Led: Ambar	Carga normal	Activado	Activado
13.45 – 13.55	Led: Ambar Rojo	Carga alta	Modulación	Activado
13.55 – 13.45	Led: Ambar	Carga normal	Modulación	Activado
13.45 – 11.5	Led: Ambar	Carga normal	Activado	Activado
11.0 – 10.0	Led: Ambar	Carga baja	Activado	Activado
10.0 – 9.9	Led: Verde.	Sin Carga	Activado	Desactivado

A continuación se presentan imágenes que reflejan el comportamiento del regulador, teniendo como única entrada, la tensión de la batería simulada por una Fuente de tensión.

5.2 Pruebas y Resultados.

Estado de Carga (9 -11V):

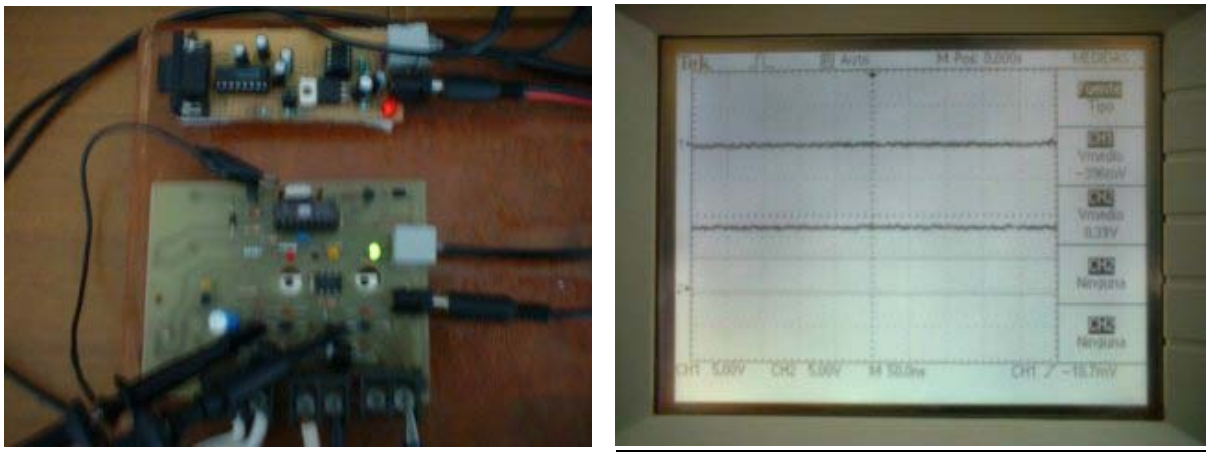


Fig. 5.5: *Carga baja en la batería; usuario desconectado.*

Si la tensión de la batería se encuentra en este intervalo, el usuario no tiene conexión con la batería, el regulador espera que la batería se cargue hasta que su tensión llegue a 11 voltios

[16] El código principal del programa en VC++ se encuentra en el Apéndice C.

y después conectará a la carga o usuario. En este estado, no hay transmisión de datos entre el regulador y la EEPROM ya que no es un estado de descarga. Este estado es indicado por el *led* verde.

Estado de Carga - Descarga (11V - 13.5):

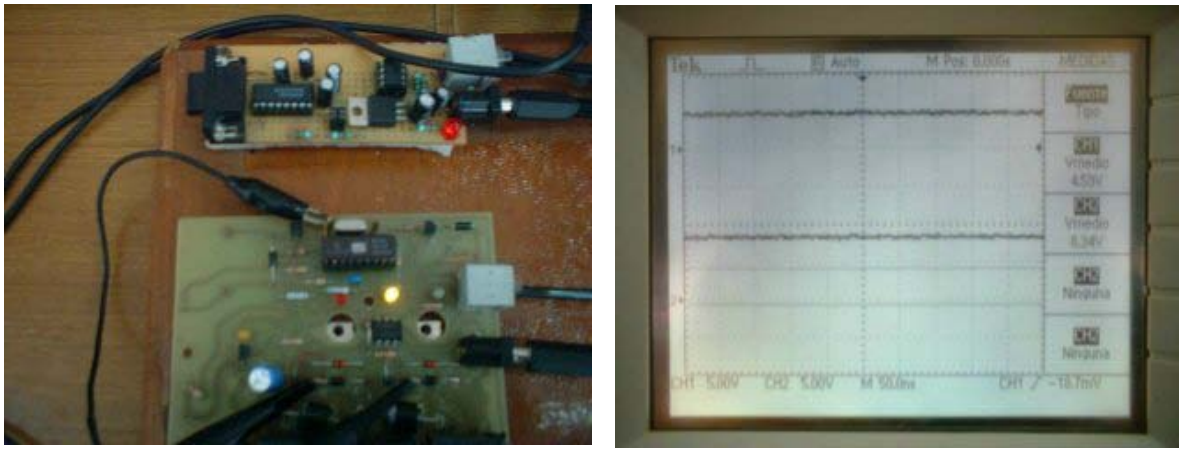


Fig. 5.6: *Estado de Flotación.*

Si llega la tensión a 11 voltios la conexión del usuario es inmediata, la transmisión de datos se activa y se entra en un intervalo de permanente estado llamado banda de flotación que esta entre 11 y 13.5 voltios.
A este intervalo se le llama banda de flotación porque tanto el panel como el usuario se encuentra conectados permanentemente a la batería (Ver Osciloscopio en Fig. 5.6).
Este estado es señalizado por el *led* ámbar.

Estado de Carga - Descarga y Envío de Datos a la EEPROM:

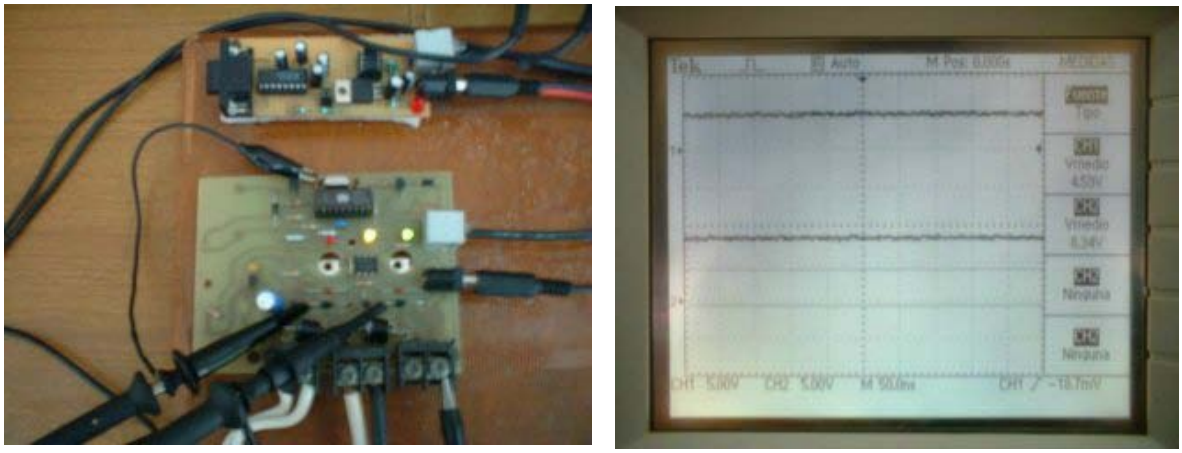


Fig. 5.7: *Estado de Flotación y Comunicación.*

El nivel de tensión se encuentra en el mismo intervalo anterior, sólo que ahora como la transmisión ha sido activada, la tarjeta esta transmitiendo datos al mismo tiempo. Es decir regula y transmite datos al mismo instante. El estado de conexión del Usuario y del Panel es el mismo (Ver Osciloscopio en Fig. 5.7).

Ya que en estado de baja carga (V < 11Voltios), no hay comunicación, la señalización de la comunicación se hace por la activación del led verde (Ver Tarjeta en Fig. 5.7).

Estado de Modulación (Vbat = 13.5 V):

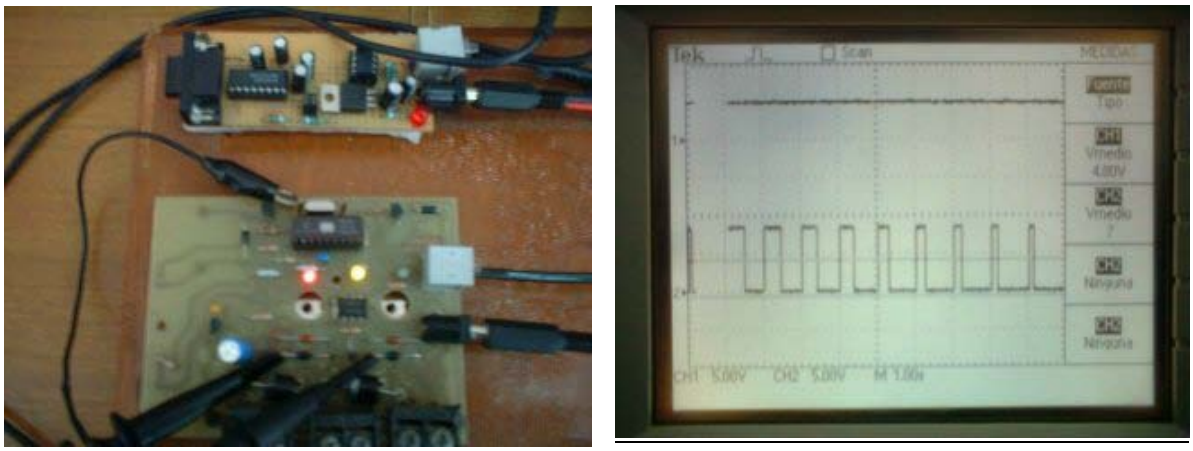


Fig. 58: *Modulación.*

Una vez superada la tensión 13.5 se entra en el estado de modulación en donde se hace la modulación del ancho del pulso de activación del MOSFET que permite de energía entre Panel y Batería. Como se puede apreciar en el osciloscopio de la figura 5.8 el ancho de activación va disminuyendo, bajando de esta manera el flujo de energía que va hacia la batería. Este estado es señalizado por el *led* rojo.
Cortando la onda directamente haría que desperdiciemos la energía durante el *duty cicle* en donde el pulso esta activo. Este el fundamento de lus o de PWM como principio de regulación.

Estado de Modulación y envío de datos:

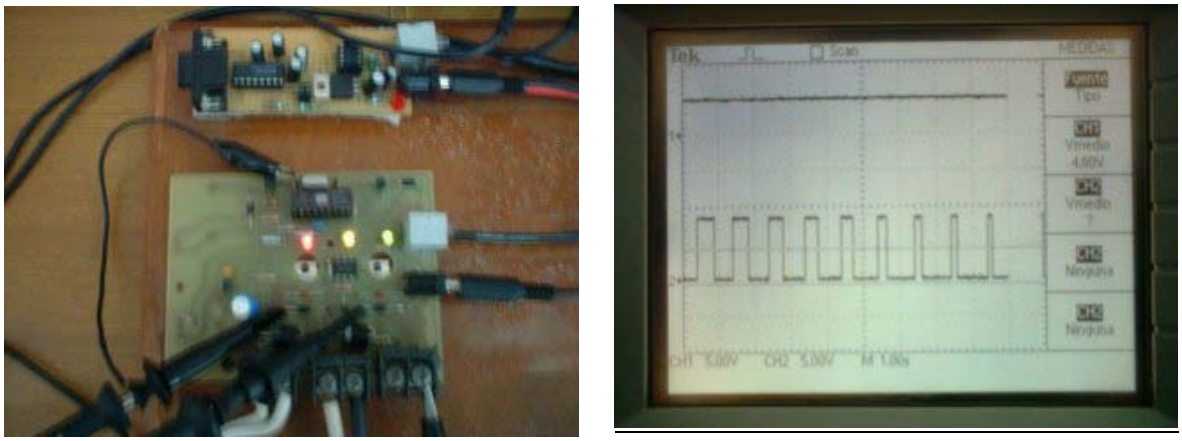


Fig. 5.9: Modulación y envío de datos.

La modulación continúa pero justo coincide con el tiempo del envío de los datos, es decir al mismo tiempo que modula, hace la transmisión. Esto es posible ya que el con el

microcontrolador PIC se pueden manejar tiempos de hasta 1us lo que no es perceptible por nosotros. Cuando se habla de ejecutar varias tareas a la vez "*Real Time Kernel*" (base de los sistemas operativos), lo que se esta haciendo es la planificación de tareas con interrupciones en la ejecución de cada una de ellas para proceder a ejecutar otra.
La base de esta implementación es programar tareas que se ejecuten en tiempos múltiplos de un tiempo mínimo de interrupción.

Estado de sólo descarga (Vbat se mantiene en 13.5 V):

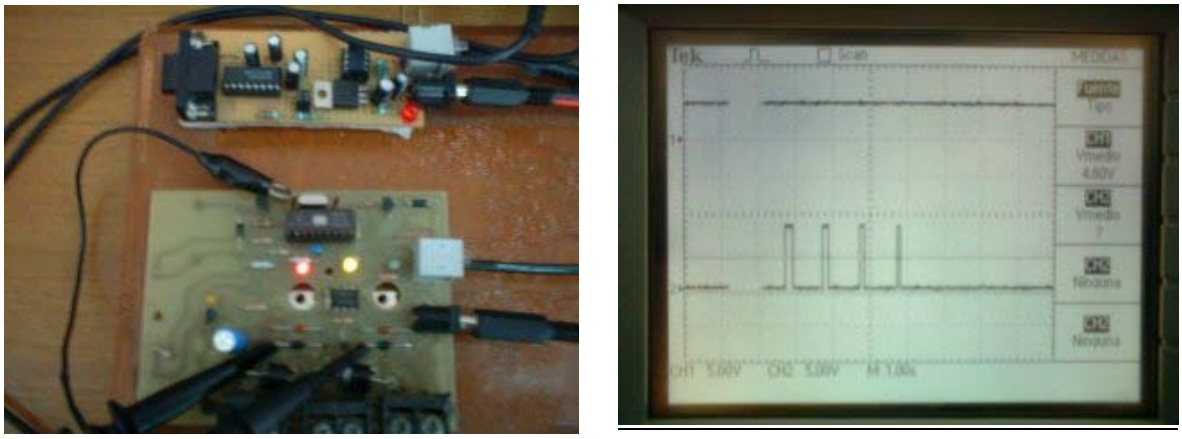


Fig. 5.10: *Estado de sólo descarga*

Si la tensión de la batería permanece mayor a 13.5 por más de 10 segundos la modulación acabará cortando definitivamente el flujo de energía entre la batería y el Panel. Esto se hace para evitar la sobrecarga de la batería (ver Osciloscopio en Fig. 5.10). Entonces sólo se quedará conectado el usuario permitiendo sólo la descarga.

De vuelta al estado de Carga - Descarga (13.3<V<13.5):

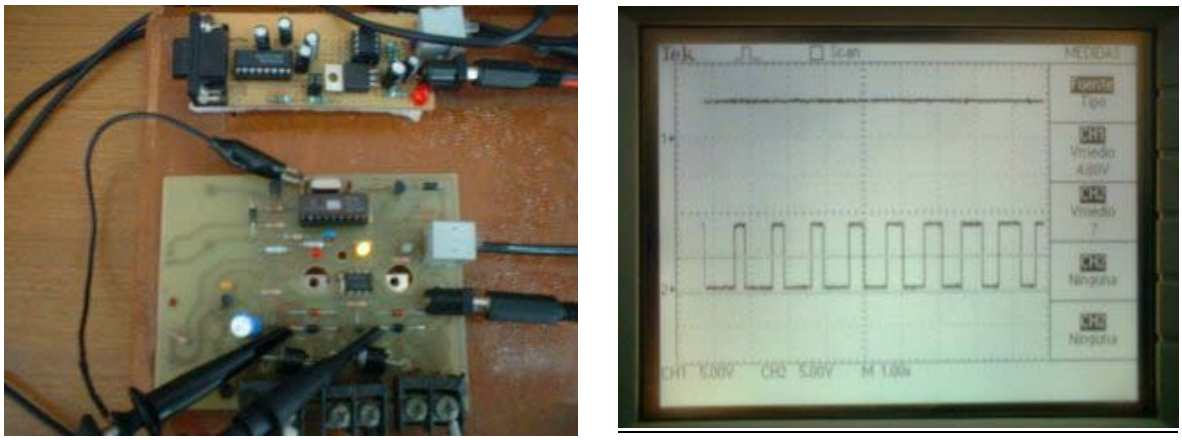


Fig. 5.11: *Reconexión en Alta.*

Si el nivel de tensión de la batería desciende y está entre 13.3 y 13.5, el regulador modula el ancho del pulso de conexión del Panel, haciéndolo cada vez más reciente (ver Osciloscopio en Fig. 5.10). No se conecta directamente el Panel ya que como se va estar

apenas por debajo del límite 13.5 puede ser que la tensión suba al conectar el panel, lo que hace este principio es aprovechar hasta la menor cantidad energía que pueda ser absorbida.

Tensión se mantiene menor a 13.5V

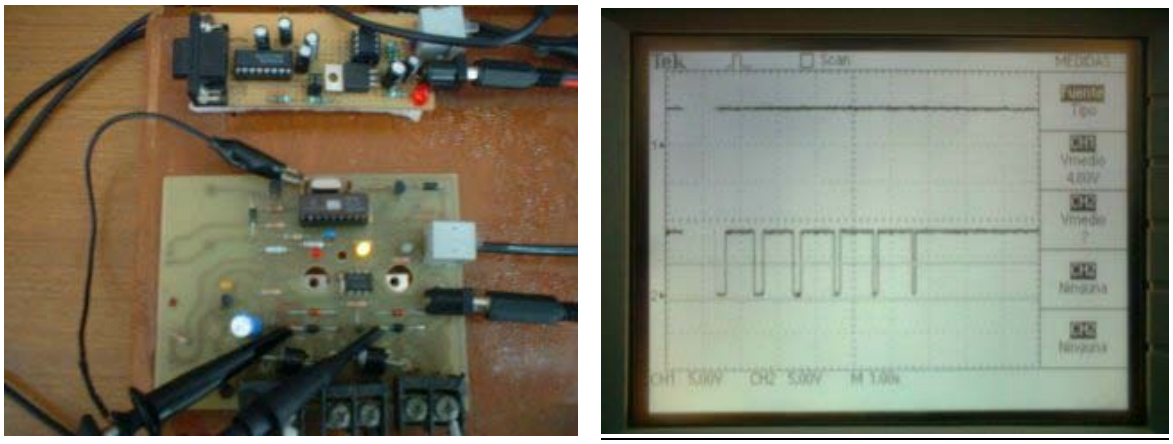


Fig. 5.12: *Estado de Carga -Descarga*

Sí la tensión de la batería permanece menor a 13,5 por más de 10 segundos, el estado de la batería volverá a Carga - Descarga, la modulación creciente activará permanentemente el flujo de carga de la Batería (Ver Osciloscopio en Fig. 5.12) y se entraría otra vez en la banda de flotación.

De vuelta a la región de flotación (11 < V < 13.5):

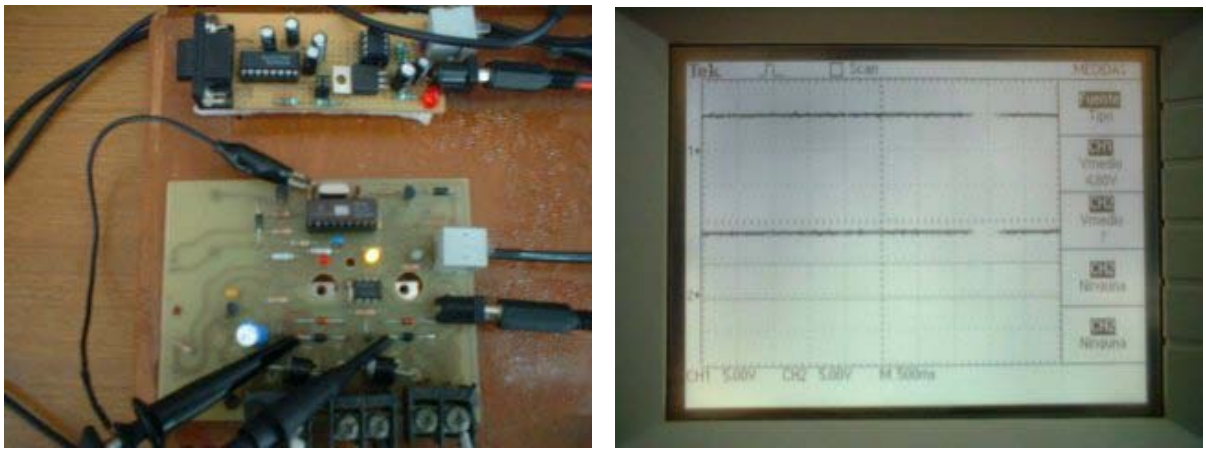


Fig. 5.14: *Banda de flotación*

La Fig. 5.14 muestra el estado en de la batería, en donde el Panel y el usuarios se encuentra conectados a la batería (Ver Osciloscopio en Fig. 5.14). Vale recordar que el regulador considera una histéresis en la conexión y desconexión de la carga (Usuario) de 0.5 Voltios y esto es modificable por *software*.

Lógicamente la transmisión de datos se ha venido dando en paralelo a lo ejecutado hasta este momento.

Desconexión de la carga (V < 10):

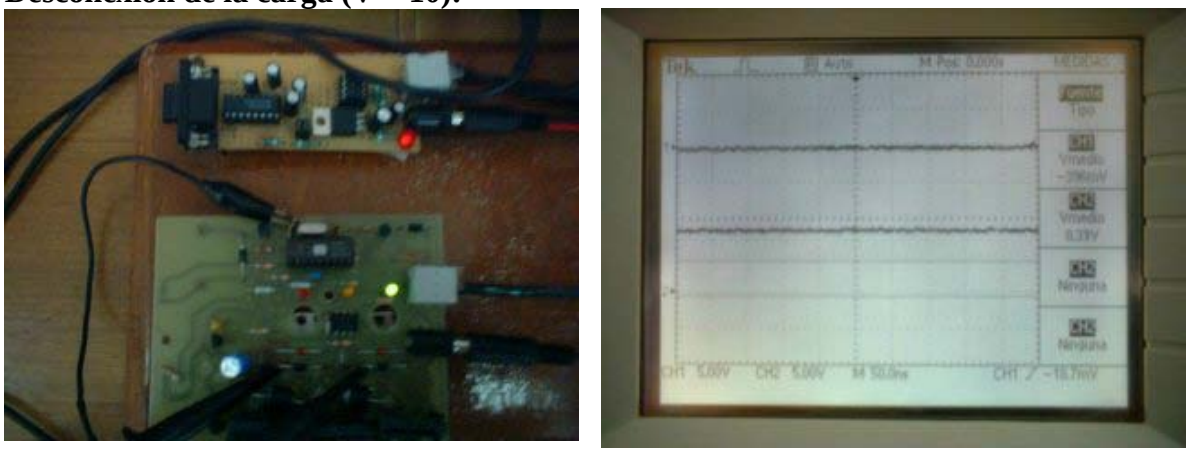


Fig. 5.15: *Desconexión del consumo*

Finalmente cuando la tensión de sciende bajo los 10 voltios la carga (el usuario) es desconectada y todo queda como al principio del análisis. En esta etapa ya no hay transmisión de datos, este momento es el final de la descarga.

Evaluación de los datos adquiridos.

La capacidad de una batería se mide en Ah (amperios-hora), a medida que la batería se carga o se descarga, va aumentado o disminuyendo a su valor máximo o mínimo según corresponda.
El tiempo de descarga depende de los amperios requeridos por el usuario, el cual puede fluctuar de una manera irregular y diferente instante a instante (figura 5.8)

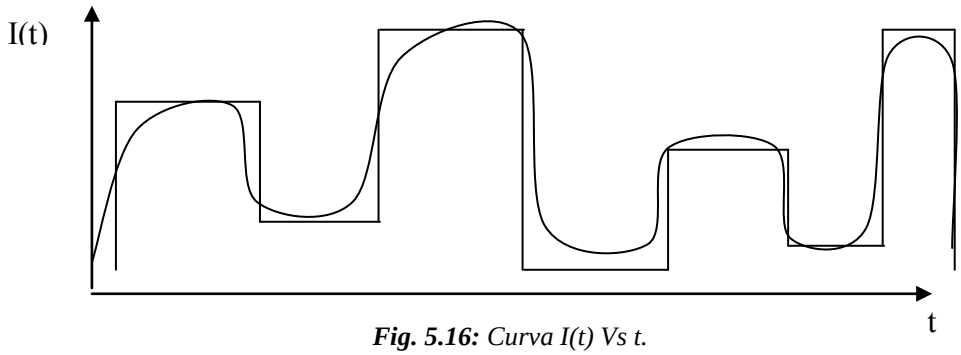


Fig. 5.16: *Curva I(t) Vs t.*

I(t) depende del usuario, del requerimiento de corriente y no es una función estimable. Para obtenerla tendríamos que adquirir datos de corriente cada cierto tiempo y de esta manera aproximar dicha curva. Como el objetivo es hallar la potencia consumida también se

debería adquirir datos de tensión de la batería. Si el tiempo de muestreo de datos es m, entonces los KWh consumidos:

$$\text{Ener} = (\text{V1 x I1} + \text{V2 x I2} + \text{V3 x I3} + \text{V4 x I4} + \text{V5 x I5} + \ldots\ldots\ldots \text{Vn x In}) \text{ x m(h)} \qquad 5.1$$

El área total bajo la curva en la gráfica I(t) vs t es igual a la capacidad de la batería consumida por el usuario.

$$\int_0^t I(t)xdt = Ah(Consumidas) \qquad 5.2$$

Lo que lleva a concluir en obtener un valor medio para la corriente que multiplicado por el tiempo total de descarga resultaría el mismo valor del área bajo la curva real de I(t) vs t.

$$\text{Im} \times T = \int_0^t I(t)xdt = Ah\langle Consumidas\rangle \ldots\ldots\ldots \qquad \ldots\ldots\ldots\ldots 5.3$$

El parámetro Ah es característica principal de toda batería, entonces la corriente media resultaría:

$$\text{Im} = \frac{Ah(Consumidas)}{t(h)} \ldots\ldots\ldots \qquad \ldots\ldots \qquad \ldots\ldots\ldots\ldots\ldots\ldots 5.4$$

La variación del voltaje en la batería es irregular esta constantemente cambiando. Gracias a la tarjeta de adquisición de datos se puede interpolar una curva V vs T minuto a minuto.

$$Ener = \text{Im} \times \int V(t) \times dt \qquad 5.5$$

La tarjeta reguladora envía cada dos minutos los datos de voltajes tomados cada minuto. Con esto obtendríamos una curva con un Rango de voltajes variables y un dominio de intervalos de tiempos definidos separados por 1 minuto.
Entonces:

$$\int V(t) \times dt = (V1 + V2 + \ldots\ldots + Vt) \times t \qquad 5.6$$

$$\int V(t) \times dt = Vm \times T \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots 5.7$$

Vt = Voltaje de la batería a los t minutos de descarga de la batería obtenido por el *data loger.*
t =1 minuto.
T = Tiempo total de descarga.
Vm = Valor medio de la Tensión.

Reemplazando la ecuación 5.7 en la 5.5 obtenemos:

$$Ener = \text{Im} \times Vm \times T \qquad 5.8$$

Reemplazando la ecuación 5.3 en la 5.8 obtenemos:

$$Ener = Vm \times Ah(Consumidas) \ldots\ldots\ldots \qquad \ldots\ldots\ldots \qquad \ldots\ldots\ldots 5.9$$

El *data loger* se utiliza para guardar los datos de tiempo descarga día a día y para luego hacer una estadística de la intensidad de la luz solar durante un periodo, aproximando el rendimiento del panel constante. De esta manera se puede destinar mejores zonas de aprovechamiento de la luminosidad solar.

$$EnerSolar = \frac{Ener}{\eta} \qquad 5.10$$

EnerSolar = Energía Solar en la zona donde esta ubicado el panel.
η = Rendimiento del Panel Solar.
Para poder comprobar el modelo expuesto se debe adquirir los datos de tensión que se han almacenado en la memoria EEPROM de la tarjeta de adquisición. Para poder hacer esto se hizo un ejecutable (Ver figura 5.17) cuyo diagrama de flujo se encuentra en el capítulo anterior.

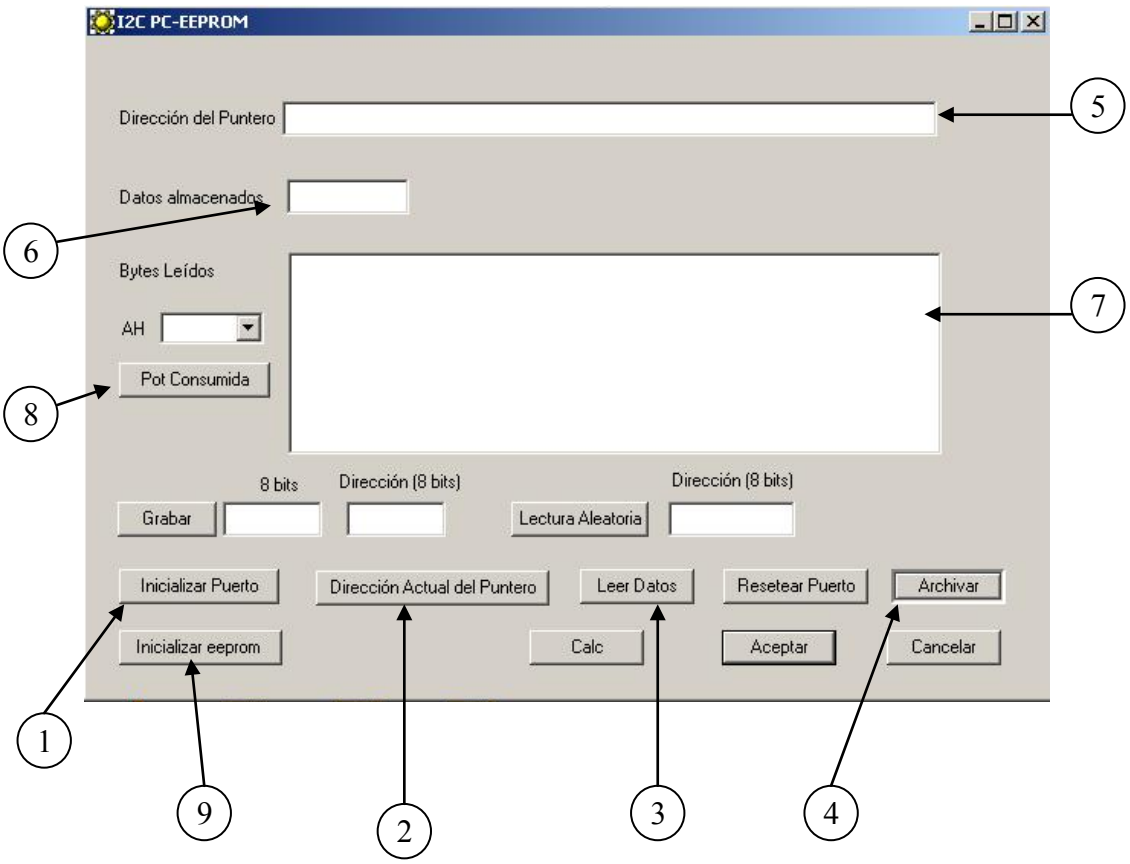


Fig. 5.17: Programa Solar I2C.

En donde:

1 (Inicializar Puerto): El primer botón que hay que accionar, pues este permite inicializar el puerto COM1 para su uso en la comunicación.

2 (Dirección del Puntero): Para poder leer los datos de la tarjeta de adquisición, primero se debe saber cuantos datos tiene, pulsando este botón.

3 (Leer Datos): Una vez leída la posición del puntero de la memoria EEPROM, con este botón procedemos a descargar los datos de voltajes que hay en esta memoria y los datos aparecerán en pantalla.

4 (Archivar): Este botón se debe pulsar en el caso de que se quiera archivar los datos para su posterior procesamiento; el tipo de extensión es I2C y se puede abrir con cualquier hoja de procesador de textos como *Excel*.

5 (Estado de la PC y EEPROM): En esta caja de testo se puede ver el estado de la PC, es decir una vez hecha cada tarea saldrá que la tarea ha sido ejecutada, esto sirve para ver si la comunicación se está ejecutando correctamente. En esta casilla se puede ver también la posición del puntero.

6 (Datos almacenados): Inmediatamente después de haber interrogado la dirección del puntero, esta casilla mostrará la cantidad de datos almacenados en la memoria EEPROM.

7 (Datos Leídos): Una vez terminado el proceso de lectura de datos, accionado por el botón 3, se mostrará en esta casilla los datos que se han descargado de la memoria EEPROM.

8 (Potencia): En esta caja editable, el usuario puede escoger la capacidad de la batería y así determinar la energía que se descargó de la batería.

9 (Inicializar EEPROM): Este botón sirve para borrar los datos de la memoria y dejarla en cero. Lo que internamente se hace es mover el puntero de la memoria a la dirección cero de ésta. Entonces para la próxima escritura se empezará a sobrescribir los datos anteriormente gravados.

Los demás botones sirven para el control total de la memoria, como grabar o leer aleatoriamente un dato.
Los datos guardados en la memoria son los voltajes de la fuente de tensión que se usaron para ver el comportamiento del regulador. Se hizo esto para ver la efectividad del sistema ya que con la fuente se puede variar la tensión manualmente en intervalos de tiempos más cortos que la de una descarga real de batería.

Procedemos a la descarga de la Memoria:

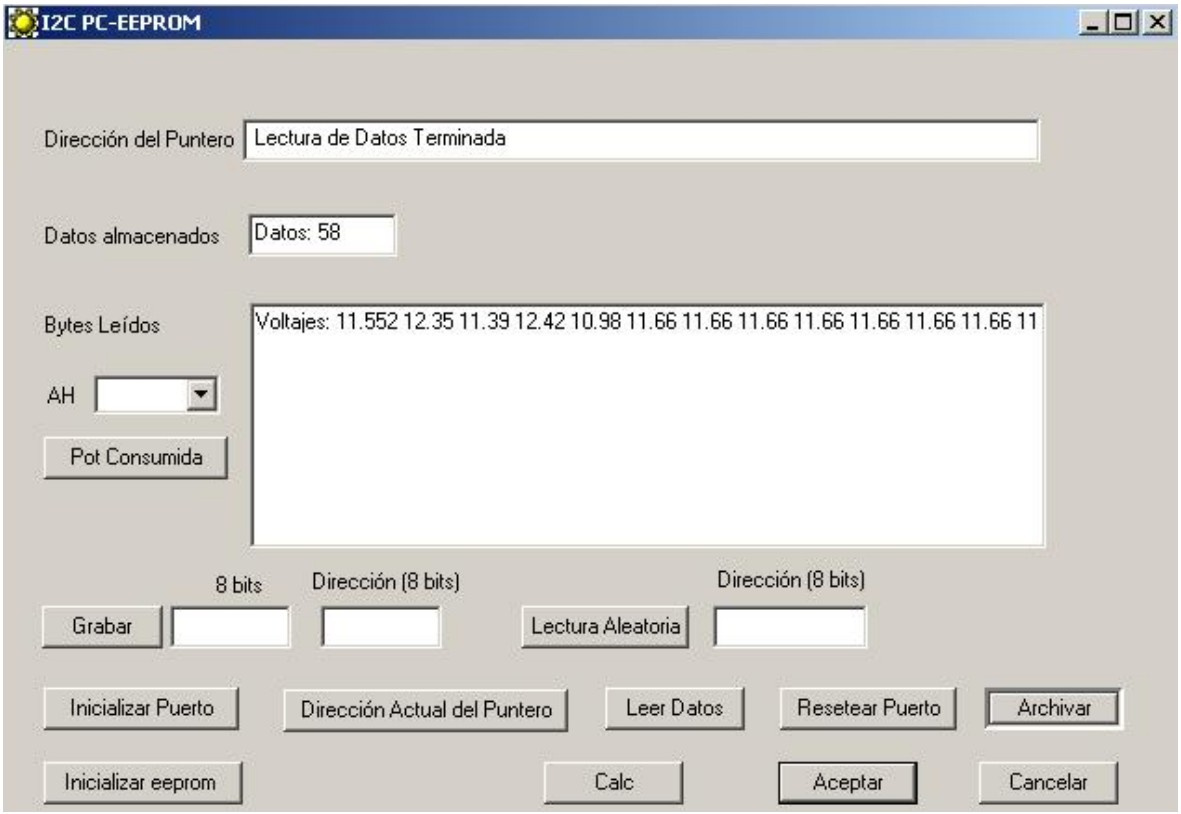


Fig. 5.18: *Descarga de Datos almacenados en la tarjeta de adquisición.*

Estos datos se encuentran en el archivo definido por programa VoltdeBat.I2C y se puede abrir desde *Excel*.
Este archivo se encuentra vinculado con otro archivo en donde se grafica la curva V vs T, la vinculación se hace para actualizar automáticamente la curva una al archivar los datos desde el ejecutable I2C.

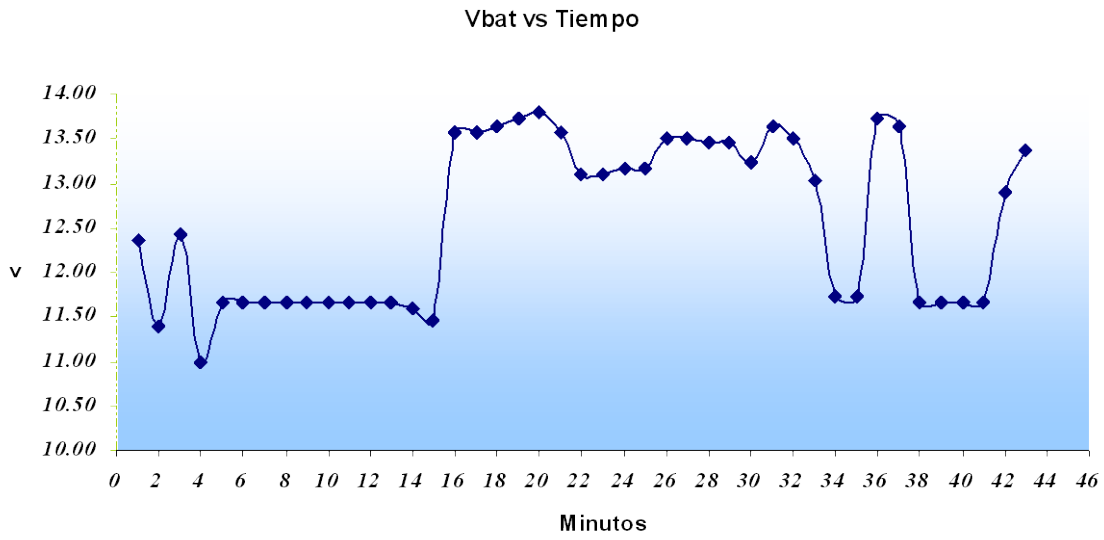


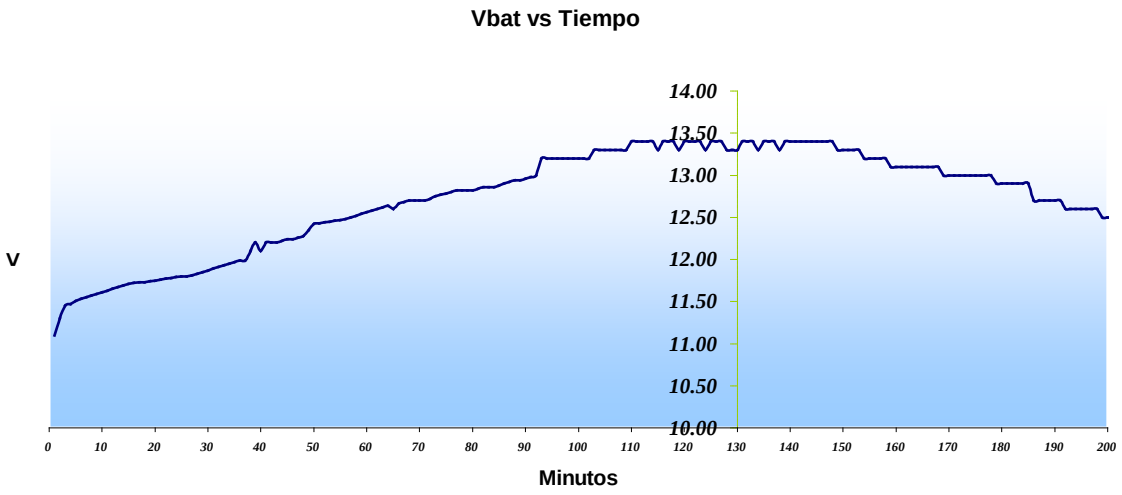
Fig. 5.19: *Diagrama V(t) vs T obtenido de la descarga de datos de un sistema simulado por una fuente de tensión.*

Lógicamente esta curva no es la de la batería sino de la fuente de tensión, cuya tensión fue variándose de manera puntual como se muestra en la curva.
Una curva real nos permite ver la profundidad de descarga de la batería, esta curva posee valores de carga y descarga de la misma. Desde un valor máximo de tensión hacia la izquierda es la carga y hacia la derecha la descarga. Con esto podemos hallar el tiempo de descarga de la batería, es decir podremos saber la profundidad de la descarga de la batería.
La potencia consumida se obtiene hallando el valor medio de todas las tensiones registradas y multiplicar este valor con los Ah descargados que para una tensión de 11 Voltios (tensión en donde se corta el consumo del usuario), este valor es el 77 - 78% de su capacidad normal. Es decir para una batería en buen estado de 150 A h que se carga al máximo (V = 13.5) y luego se descarga hasta que su tensión llegue a 11 voltios, los Ah descargados son 77.5*150/100 = 116.25 Ah.
Este valor multiplicado por el valor medio de la tensión que en nuestro caso viene a ser 12.61, da 1465.913 Wh.

En la Universidad se realizó la siguiente prueba con una instalación fotovoltaica de las siguientes características:

Panel: 50 *watts*, 18 voltios en vacío.
Batería: 12 voltios, 20Ah.
Carga: 20 *watts*, 12 voltios.

La prueba se realizó el día 23 de Diciembre de 2002 desde las 9 am, hasta las 6 pm de la tarde.
El regulador funcionó bien y de la tarjeta de adquisición se obtuvieron los siguientes datos de tensión de la batería cada minuto:



Esta curva real obtenida del *data loger*, refleja el comportamiento de la carga y descarga de la batería en función del tiempo. Este día fue muy radiante, como apreciamos la batería llega a una nivel de 13.5 voltios y luego se mantiene; en este tiempo el regulador esta modulando. Se aprovecha el máximo de energía ya que la tensión de desconexión y rearme del panel es la misma. Esto se logra debido a que la regulación no desactiva directamente al panel sino va limitando poco a poco, mediante un tren de pulsos de decreciente al mosfet, el flujo de energía del panel hacia la batería. Esto hace que la batería vea como valor de tensión del panel, una tensión variante en el tiempo, lo que se reduce a una tensión eficaz cada vez menor, según el ancho de pulso de la onda de activación.
La tensión media es 12.36 y se procede con el análisis anterior:
En la zona de carga de la batería, apreciamos que el panel esta alimentando a la batería y a la carga.

$$EnerPanel = Vmc \times Ah\ (Cargados) + 20 \times t_c . + 20 \times t_d - Vmd \times Ah\ (Descargados)$$

Por la conservación de la energía la energía cargado es la que se descargan, entonces:

$$EnerPanel = 20 \times (Tc + Td)$$

EnerPanel: Energía dada por el panel.
Vmc: Voltaje medio durante el tiempo en que se carga la batería.
t_c: Tiempo en el que se ha cargado la batería.
t_d: Tiempo de descarga de la batería.
De la gráfica obtenemos el tiempo de carga (t_c = 2.5 horas) y de descarga (t_d = 6.5 horas).

$$EnerPanel = 120\ wh.$$

En este caso podemos determinar el valor de la energía consumida porque conocemos el valor de la carga. En el caso de no saberlo sería imprescindible el uso de un sensor *hall*[17].
El resultado es lógico. La batería ha quedado en el mismo estado inicial y sólo ha servido para acumular la energía y repartirla en un mayor tiempo, a esto habría que aumentarle la energía que consumen las tarjetas y que consume la batería (resistencia interna), pero no influye ya que son de un valor despreciable con respecto al consumo del usuario.

[17] Sensor de corriente que se basa en el principio del efecto *hall*.

El sistema de regulación constituido se puede usar también para obtener la curva de la tensión del panel versus tiempo, desde el amanecer hasta la puesta del sol. De esta manera podríamos hacer una estadística del estado del clima en ciertas zonas claves y se podría escoger la mejor zona para poner los paneles solares. Realizar este estudio tomaría muchos días de prueba y no es objetivo de esta tesis realizarlo pero que da demostrado que el sistema implementado brinda la ayuda suficiente para poder hacerlo.
En la siguiente sección están las conclusiones de la presente tesis.

CONCLUSIONES

- Gracias a la buena regulación, el nivel de tensión de la batería siempre está entre los niveles mínimo y máximo programados por software que en este caso fueron 11V y 13.5V.

- El regulador es para baterías de diferentes tensiones nominales como las comunes de 12, 24 y 48 V oltios (el sistema es muy flexible). Para cambio de niveles de tensión más altos hay que seleccionar los MOSFETS oportunos.

- La banda de igualación de la batería es la variación de tensión (medida en voltios) del panel solar en 10 segundos, que para un estado óptimo del clima vendría a ser .05 voltios (esto por la respuesta lenta del panel ante las variaciones de radiación). Esta banda de igualación se puede cambiar por programa justando la modulación por ancho de pulso.

- La conexión y desconexión de la carga, no afecta la regulación, esto es debido a que en el diseño se ha considerado una histéresis de un voltio. Esta histéresis es modificable por *software*.

- La lectura de los niveles de tensión por una PC a través de la EEPROM cada minuto, permite obtener con precisión la curva de carga y descarga de la batería, esta curva es un indicador del estado de la profundidad de la descarga que se relaciona con la vida útil de la misma.

- Conectando sólo el panel en los bornes del regulador correspondientes a la batería, obtenemos la curva de tensión del panel solar vs. tiempo. Teniendo esta curva para diferentes días y posiciones, se puede obtener la radiación del día aproximada y una óptima posición para los paneles.

- La tarjeta de adquisición de datos implementada no sólo es útil para el regulador, sino es útil para la mayoría de procesos en donde se requiera el almacenamiento de datos.

- La memoria de la tarjeta de adquisición de datos es expandible. Gracias a su direccionamiento físico que consta de 3 *bits* se pueden colgar 8 memorias EEPROM de un mismo de la misma línea de datos y sólo bastaría una modificación en el *software* que se ejecuta desde la PC para descargar los 32768 datos.

- La conexión de un radio *modem* en la tarjeta permitiría el envío de datos para el constante monitoreo de la tensión en tiempo real.

- El uso de un sensor de corriente en la tarjeta (sensor *hall*) permitiría que esta envíe datos de corriente. Con esto se tendría en constante monitoreo los datos de tensión y corriente, lo que haría a la tarjeta además de un regulador, un totalizador.

- Otra ventaja del uso del sensor es la obtención de la capacidad de la batería. Esta variable es fundamental en la batería y va cambiando a medida que la batería se deteriora. Con la obtención de este valor se puede saber el estado de la batería y no sería necesario utilizar un densímetro.

REFERENCIAS

[1] HUARCAYA RENTERÍA, José Luis. Tesis Ing. "Sistema SCADA para una red de semáforos". Departamento de Electrónica y Automática. Universidad de Piura. Piura, Perú. 1999.

[2] BARRIENTOS SILVA, Carlos. Tesis Ing. "Identificación y control de un sistema multivariable de dos rotores". Departamento de Electrónica y Automática, Universidad de Piura. Piura, Perú. 1998.

[3] BIFFI CHIRA, Jenny Anthuanet. Tesis Ing. “Control Robusto aplicado a un Sistema Multivariable de 2 rotores: Twin Rotor Mimo System (TRMS)”. Departamento de Electrónica y Automática, Universidad de Piura. Piura, Perú. 1998.

[4] VÁSQUEZ DÍAZ, Edilberto. Tesis Mgter. "Identificación utilizando SubEspacios de Estado". Departamento de

Ingeniería de Sistemas y Automática. Universidad de Valladolid, España. 1999.

[5] DE KEYSER, Robin. Apuntes del curso “Adaptive and Predictive Control”. Maestría en Sistemas de Control Industrial. Departamento de Ingeniería de Sistemas y Automática. Universidad de Valladolid. Valladolid, España. 1998

[6] CAMACHO E.F. Y BORDONS. “Model Predictive Control in the Process Industry”.

[7] SKOGESTAD and POSTLETHWAITE. "Multivariable feedback control". John Willey & Sons Ltd. Baffins Lane, Chichester, West Sussex, PO19 IUD, England.

[8] CEBALLOS, Francisco. "Microsoft Visual C++. Aplicaciones para Win32". 2da. Edición, 1997.

[9] CEBALLOS, Francisco. "Microsoft Visual C++. Programación Avanzada en Win32". 1ra. Edición, 1999.

[10] LJUNG, Lennart. "System Identification. Theory for the User”, 1987.

[11] WILLIAMS, R.I. "Handbook of SCADA System for the Oil & Gas Industry”, 1st. Edition, 1992

[12] OZKUL, Tarik. "Data acquisition and process control using personal computers”, 1996

APÉNDICE A

CÓDIGO PRINCIPAL DE LOS PROGRAMAS EN LENGUAJE ENSAMBLADOR

RUTINAS PARA EL ACCESO A LA TARJETA DE ADQUSICIÓN DE DATOS

Archivo: Reg.asm

```
;       Regulación de Voltaje       20/08/02
        List    p=16C711
        #include "P16C711.INC"
;-------------------------------------
;Inicialización
;-------------------------------------
;-----------------------------------------------------------------
;Si el Factor de Divisor tensión es 5.1 ( Vmáx en RA0 es 25.5v)
;R5.1
;#define Max  .135                  ;Representa a 13.5
;#define Minr .115                  ;Representa a 11.5
;#define Min  .110                  ;Representa a 11
;-----------------------------------------------------------------
;Si el Factor de Divisor tensión es 4 ( Vmáx en RA0 es 20v)
;R4
;#define Max  .172                  ;Representa a 13.5
;#define Minr .146                  ;Representa a 11.5
;#define Min  .141                  ;Representa a 11
;-----------------------------------------------------------------
;Si el Factor de Divisor tensión es 3.5 ( Vmáx en RA0 es 17.5v)

;#define Max  .196                  ;Representa a 13.5
;#define Minr .167                  ;Representa a 11.5
;#define Min  .160                  ;Representa a 11
;#define Minfd       .145           ;Representa a 10
;-----------------------------------------------------------------
;Si el Factor de Divisor tensión es 3.5 ( Vmáx en RA0 es 17.5v)
#define Max   .196                  ;Representa a 13.5
#define Minr  .180                  ;Representa a
#define Min   .169                  ;Representa a 11.5
#define Minfd .160                  ;Representa a 11
;-----------------------------------------------------------------
;Si el Factor de Divisor tensión es 3 ( Vmáx en RA0 es 15v)
;R3
;#define Max  .229                  ;Representa a 13.5
;#define Minr .195                  ;Representa a 11.5
;#define Min  .187                  ;Representa a 11
;-----------------------------------------------------------------
#define RB0   PORTB,0               ;Rx
#define RB1   PORTB,1               ;Tx
#define RB2   PORTB,2               ;Alarma (Se activa con cero)
```

```
#define RB3   PORTB,3              ;Panel
#define RB4   PORTB,4              ;Bajo   (Se activa con cero)
#define RB5   PORTB,5              ;Salida
#define RB6   PORTB,6              ;Load
#define RB7   PORTB,7              ;Zumbador
;-------------------------------------
#define RA0   PORTA,0
#define RA1   PORTA,1
#define RA2   PORTA,2              ;Canal de Conversión A/D
#define RA3   PORTA,3
#define RA4   PORTA,4              ;Alto (Se activa con cero)
#define RA5   PORTA,5
#define RA6   PORTA,6
#define RA7   PORTA,7
;---Para EEPROM-----------------------
#define       SDA   PORTB,0
#define       SCL   PORTB,1
#define       a     .15
;-------------------------------------
        cblock 0x10                ;Creamos un espacio en la Memoria RAM
        Modulador
        Contador
        DirecLB
        DirecHB
        Horas
        Cobi
        switch
        Cor
        Segundazos
        Minutazos
        Tiempo
        Dia
        CByte
        Delaya
        Delayc
        DHB
        DLB
        i
        d
        Dato
        endc
#define AckBit        Cor,7
;-------------------------------------
```

```
        ORG     0
        goto    INIT
        nop
        nop
        bcf     INTCON,T0IF
        bsf     Cor,0
        call    INTERR
        movlw   .60
        movwf   TMR0
        bsf     STATUS,RP0
        movlw   b'00000111'
        movwf   OPTION_REG^0x80         ; Pre-escaler 1:256
        bcf     STATUS,RP0
        movlw   b'10100000'
        movwf   INTCON
        retfie
;=================================================================
INIT    bsf     STATUS,RP0
        movlw   b'00001111'
        movwf   TRISA^0x80              ;RA0-RA3, Entradas
        movlw   b'00000011'
        movwf   TRISB^0x80              ; RB7-RB2 Salidas
        clrf    ADCON1^0x80             ; C  anales A  nalógicos  RA0-
RA3
        bcf     STATUS,RP0
        clrf    ADRES
;
        movlw   .240
        movwf   Segundazos
        movlw   .5
        movwf   Minutazos
        movlw   .2
        movwf   d
        clrf    Tiempo
        clrf    Horas
        clrf    switch
        clrf    PORTA                   ;Limpiamos  Puerto A
        clrf    PORTB
        clrf    Dia
        clrf    Cor
        clrf    CByte

        bsf     STATUS,RP0
        movlw   b'00000111'
        movwf   PTION_REG^0x80          ;Pre-escaler 1:256
```

```
        bcf     STATUS,RP0
        movlw   .60
        movwf   TMR0
        movlw   b'10100000'     ;Con Interrupcionesn edl timer0
        movwf   INTCON
;-------------------------------------
;Cargado de Batería
;-------------------------------------
        movlw   .20
        movwf   Modulador
        bsf     RB3             ;Activamos la carga de la Batería
;-------------------------------------
TEST    bsf     RA4             ;Desactivamos nivel alto
        bcf     RB7             ;Desactivamos el Zumbador
        call    CONVER
        rrf     switch,0
        btfsc   STATUS,C
        goto    Fdia
        movf    ADRES,W
        sublw   Min
        btfsc   STATUS,C        ;Volt de Batería es < 11.0 V
        goto    Mosfet1         ;Si
;
ALAR    bsf     switch,0
        bcf     RB2             ;Activamos la alarma
        bsf     RB6             ;Activamos la Carga al Foco
        movf    ADRES,W                 ;No
        sublw   Minr
        btfsc   STATUS,C        ;Volt de Batería es < 11.5 V
        goto    Alarma          ;Si
        bsf     RB4             ;Desactivamos nivel bajo.
        bcf     RB7             ;Desactivamos el Zumbador
        movf    ADRES,W         ;No
        sublw   Max
        btfss   STATUS,C        ;Volt de Batería es > 13.5 V
        goto    PWM             ;Si
        goto    PWM1            ;No
;-------------------------------------
;------------------------------
;Rutina de espera de un segundo
;------------------------------
Espera
        bcf     Cor,0
;
```

```
Loop1
        btfss       Cor,0
        goto        Loop1
        bcf         Cor,0
        decfsz      Contador,F
        goto        Loop1
        Return
;-------------------------------------
;Rutina de Conversión
;-------------------------------------
CONVER
        bcf         Cor,0
        movlw       b'01010001'
        movwf       ADCON0          ;activamos el Convertidor A/D
        bsf         ADCON0,GO_DONE
        btfsc       Cor,0
        goto        CONVER
Loop
        btfsc       ADCON0,GO_DONE
        goto        Loop
        btfsc       Cor,0
        goto        CONVER
        return
;-------------------------------------
Mosfet1
        bcf         RB4             ;Activamos el Señailzor de nivel Bajo
        bsf         RB2             ;Desactivamos alarma
        bcf         RB7             ;desactivamos el Zumbador
        bcf         RB6             ;desactivamos la Carga
        goto        TEST
;-----------------------------------
Alarma      bsf     RB4             ;Desactivamos el  S  eñalizador de  l
nivel Bajo
        bsf     RB7             ;Activamos el Zumbador
        bsf     RB3             ;ACtivamos la carga panel batería.
        goto    TEST
;-----------------------------------
Fdia    movf    ADRES,W
        sublw   Minfd
        btfsc   STATUS,C        ;Volt de Batería es < Minfd
        goto    Mosfet1fd       ;Si
        goto    ALAR            ;no, seguir con el programa principal
;-------------------------------------
Mosfet1fd
        bcf     RB4             ;Activamos el Señailzor de nivel Bajo
```

```
        bcf     RB7                 ;desactivamos el Zumbador
        bcf     RB6                 ;desactivamos la Carga
        clrf    switch
        bsf     Cor,6
        goto    TEST
;-----------------------------
PWM  bcf    RA4             ;Activamos el Señalizador del nivel alto
        movf    Modulador,0
        btfss   STATUS,Z
        decf    Modulador,F ;Decrementar el ancho de nivel alto en %5
        movf    Modulador,0
        movwf Contador
        btfss   STATUS,Z
        bsf     RB3             ;Se activa la carga de la Batería
        movf    Modulador,0
        btfss   STATUS,Z
        call    Espera
;
        bcf     RB3             ;Se desactiva la carga de la Batería
        movf    Modulador,W
        sublw   .20
        movwf Contador      ;Incrementar el ancho de nivel bajo en %5
        btfss   STATUS,Z
        call    Espera
        goto    TEST
;-----------------------------
PWM1 movlw .20
        xorwf   Modulador,W
        btfss   STATUS,Z    ;El modulador es 20
        incf    Modulador,F ;No, entonces incrementar el ancho de nivel alto en
%5
        movf    Modulador,0 ;Si
        movwf Contador
        bsf     RB3             ;Se Activa la carga de la Batería
        call    Espera
;
        movlw .20
        xorwf   Modulador,W
        btfss   STATUS,Z
        bcf     RB3             ;Desactivamos la Batería en un determinado ancho
de pulso dado por el contador
        movf    Modulador,W
        sublw   .20
        movwf Contador      ;Decrementar el ancho de nivel bajo en %5
        btfss   STATUS,Z
```

```
        call    Espera
        goto    TEST
;=======================================
INTERR
        decfsz  Segundazos,1
        return
COSA    movlw .240
        movwf   Segundazos
        decfsz  Minutazos,1
        return
        movlw .5
        movwf   Minutazos
        movf    ADRES,W
        decfsz  d,1
        movwf   Tiempo
        movlw .0
        xorwf   d,0
        btfss   STATUS,Z
        return
        movlw .2
        movwf   d
        movf    ADRES,0
        movwf   Dia
        btfsc   RB6
        call    EEPROM
        return
;===============================================

#include "piceep.ASM"
;===============================================
        END
```

RUTINA PARA LA COMUNICACION

Archivo: Piceep.asm

```
;____igual_____________________________________________________
04/12/02
;------------EEPROM-----------------------
EEPROM          bcf     RB4
                movlw .15
                movwf   i
                bcf     Cor,6
                bsf     STATUS,RP0
                bsf     SDA
```

```
                bcf     STATUS,RP0
EEPROM1
                decf    i,1
                btfsc   STATUS,Z
                return
                btfss   SDA
                goto    EEPROM1
                bsf     STATUS,RP0
                bcf     TRISB^0x80,1
                bcf     PORTB,0
                bcf     STATUS,RP0
                movlw .248
                movwf   Cobi
                movlw .2
                movwf   i
;-------------------------------------------------------------------------------
MAIN            call    READ
                movf    DHB,0
                movwf   DirecHB
                movf    DLB,0
                movwf   DirecLB
MAIN1                   bcf     AckBit
                call    START
                movlw           b'10100000'             ;Operaciónd de Escritura
                movwf   CByte
                call    CB
                call    ACK
                btfsc   AckBit
                goto    MAIN1
                call    AHB
                call    ACK
                btfsc   AckBit
                goto    MAIN1
                call    ALB
                call    ACK
                btfsc   AckBit
                goto    MAIN1
                call    MENSAJE
                call    ACK
                btfsc   AckBit
                goto    MAIN1
                call    STOP
                decfsz  i,1
                goto    DIRECCION
                goto    REANUDA
```

```
                bsf     RB4
                return
;-------------------------------------------------------------------------------
START           bsf     SDA
                bcf     SCL             ;Activamos el reloj
                call    DELAYnus
                bsf     SCL
                call    DELAYnus
                Bcf     SDA             ;Originamos la condición de inicio
                call    DELAYnus        ;Esperamos 2499us
                Bcf     SCL             ;Ponemos a cero el reloj
                call    DELAYnus
                return
;-------------------------------------------------------------------------------
CB              rlf     CByte,1
                btfsc   STATUS,C
                bsf     SDA
                btfss   STATUS,C
                bcf     SDA
                call    DELAYnus
                bsf     SCL
                call    DELAY2nus       ;nuevo
                bcf     SCL
                call    DELAYnus
                incfsz  Cobi,1
                goto    CB
                bsf     STATUS,RP0
                bsf     PORTB,0         ;Para el bit de reconocimiento
                bcf     STATUS,RP0
                movlw .248
                movwfCobi
                return
;-------------------------------------------------------------------------
ACK             bsf     STATUS,RP0
                bsf     PORTB,0         ;Para el bit de reconocimiento
                bcf     STATUS,RP0
                call    DELAYnus
                bsf     SCL
                btfsc   SDA
                bsf     AckBit
                call    DELAY2nus
                btfsc   SDA
                bsf     AckBit
                bcf     SCL
                bsf     STATUS,RP0
```

```
                bcf     PORTB,0             ;Para el bit de reconocimiento
                bcf     STATUS,RP0
                call    DELAYnus
                return

;-------------------------------------------------------------------------------
ACKM            bsf     STATUS,RP0
                bcf     PORTB,0             ;Para el bit de reconocimiento
                bcf     STATUS,RP0
                bcf     SDA
                call    DELAYnus
                bsf     SCL
                call    DELAY2nus
                bcf     SCL
                call    DELAYnus
                return
;-------------------------------------------------------------------------------
RDA             bsf     STATUS,RP0
                bsf     PORTB,0
                bcf     STATUS,RP0
RDDHB           call    DELAYnus
                bsf     SCL
                call    DELAYnus
                rlf     Dato,1
                btfsc   SDA
                bsf     Dato,0
                btfss   SDA
                bcf     Dato,0
                call    DELAYnus
                bcf     SCL
                call    DELAYnus
                incfsz  Cobi,1
                goto    RDDHB
                movlw .248
                movwfCobi
                return
;-------------------------------------------------------------------------------
NOACK           bsf     STATUS,RP0
                bcf     PORTB,0                 ;Para el bit de reconocimiento
                bcf     STATUS,RP0
                bsf     SDA
                call    DELAYnus
                bsf     SCL
                call    DELAY2nus
                bcf     SCL
```

```
                call    DELAYnus
                return
;-------------------------------------------------------------------------------
;ADDRESS HIGHT BYTE

AHB             rlf     DHB,1           ;
                btfss   STATUS,C
                bcf     SDA
                btfsc   STATUS, C       ;
                bsf     SDA             ;
                call    DELAYnus
                bsf     SCL             ;
                call    DELAY2nus
                bcf     SCL
                call    DELAYnus        ;
                incfsz  Cobi,1
                goto    AHB
                movlw .248
                movwfCobi
                return

;-------------------------------------------------------------------------------
;ADDRESS LOW BYTE

ALB             rlf     DLB,1           ;
                btfss   STATUS,C
                bcf     SDA             ;
                btfsc   STATUS, C       ;
                bsf     SDA             ;
                call    DELAYnus
                bsf     SCL
                call    DELAY2nus
                bcf     SCL             ;
                call    DELAYnus
                incfsz  Cobi,1
                goto    ALB
                movlw .248
                movwfCobi
                return
;-------------------------------------------------------------------------------
MENSAJE
TIME            rlf     Tiempo, 1       ;
                btfss   STATUS,C
                bcf     SDA             ;
                btfsc   STATUS, C       ;
```

```
                bsf     SDA             ;
                call    DELAYnus
                bsf     SCL             ;
                call    DELAY2nus
                bcf     SCL
                call    DELAYnus        ;
                incfsz  Cobi,1
                goto    TIME
                movlw .248
                movwf Cobi
                call    ACK

DAY             rlf     Dia,1           ;
                btfss   STATUS,C
                bcf     SDA             ;
                btfsc   STATUS,C        ;
                bsf     SDA             ;
                call    DELAYnus
                bsf     SCL             ;
                call    DELAY2nus
                bcf     SCL
                call    DELAYnus        ;
                incfsz  Cobi,1
                goto    DAY
                movlw .248
                movwf Cobi
                return
;-------------------------------------------------------------------------
STOP            Bcf     SDA             ;Ponemos a cero la línea SDA
                call    DELAYnus
                Bsf     SCL             ;Ponemos a uno el reloj
                call    DELAYnus
                Bsf     SDA             ;Ponemos a uno SDA
                call    DELAYnus
                return                  ; Ponemos a cero el reloj
;-------------------------------------------------------------------------
DELAYnus
                movlw .255
                movwf Delaya
                movlw .255
                movwf Delayc
XDELAYnus       decfsz  Delaya,1
                goto    XDELAYnus
                movlw a
                movwf Delaya
```

```
                decfsz  Delayc,1
                goto    XDELAYnus
                return
;-------------------------------------------------------------------------
DELAY2nus       call    DELAYnus
                call    DELAYnus
                return
;-------------------------------------------------------------------------
DIRECCION
                incf    DirecLB,1
                btfsc   STATUS,2
                incf    DirecHB,1
                incf    DirecLB,1
                btfsc   STATUS,2
                incf    DirecHB,1
                movf    DirecHB,0
                movwf   Tiempo
                movf    DirecLB,0
                movwf   Dia
                call    DELAY2nus
                clrf    DHB
                clrf    DLB
                goto    MAIN1
;-------------------------------------------------------------------------
REANUDA         clrf    Tiempo
                clrf    Dia
                return
;=========================================================================
===============
#include       "read.asm"
                END
```

Archivo: RPointer.asm

```
;Programa para leer secuencialmente una memoria 24LC32A
READ            bcf     AckBit
                clrf    DHB
                clrf    DLB
                call    START
                movlw           b'10100000'             ;Operación de Escritura
                movwf   CByte
                call    CB
                call    ACK
                btfsc   AckBit
                goto    READ
```

```
call    AHB
call    ACK
btfsc   AckBit
goto    READ
call    ALB
call    ACK
btfsc   AckBit
goto    READ
call    START
movlw           b'10100001'             ;Operaciónd de Lectura
movwfCByte
call    CB
call    ACK
btfsc   AckBit
goto    READ
call    RDA
call    ACKM
movf    Dato,0
movwfDHB
call    RDA
call    NOACK
movf    Dato,0
movwfDLB
call    STOP
return
```

APÉNDICE B

CÓDIGO PRINCIPAL DE LOS PROGRAMAS EN LENGUAJE ENSAMBLADOR

RUTINAS PARA EL ACCESO A LA TARJETA DE ADQUSICIÓN DE DATOS

Archivo: Reg.asm

```
;       Regulación de Voltaje       20/08/02
        List    p=16C711
        #include "P16C711.INC"
;-----------------
;Inicialización
;--------------------------------------------------------------------------
;Si el Factor de Divisor tensión es 5.1 ( Vmáx en RA0 es 25.5v)
;R5.1
;#define Max  .135                  ;Representa a 13.5
;#define Minr .115                  ;Representa a 11.5
;#define Min  .110                  ;Representa a 11
;--------------------------------------------------------------------------
;Si el Factor de Divisor tensión es 4 ( Vmáx en RA0 es 20v)
;R4
;#define Max  .172                  ;Representa a 13.5
;#define Minr .146                  ;Representa a 11.5
;#define Min  .141                  ;Representa a 11
;--------------------------------------------------------------------------
;Si el Factor de Divisor tensión es 3.5 ( Vmáx en RA0 es 17.5v)

;#define Max  .196                  ;Representa a 13.5
;#define Minr .167                  ;Representa a 11.5
;#define Min  .160                  ;Representa a 11
;#define Minfd       .145           ;Representa a 10
;--------------------------------------------------------------------------
;Si el Factor de Divisor tensión es 3.5 ( Vmáx en RA0 es 17.5v)
#define Max  .196                   ;Representa a 13.5
#define Minr .180                   ;Representa a
#define Min  .169                   ;Representa a 11.5
#define Minfd .160                  ;Representa a 11
;--------------------------------------------------------------------------
;Si el Factor de Divisor tensión es 3 ( Vmáx en RA0 es 15v)
;R3
;#define Max  .229                  ;Representa a 13.5
;#define Minr .195                  ;Representa a 11.5
;#define Min  .187                  ;Representa a 11
;--------------------------------------------------------------------------
#define RB0   PORTB,0               ;Rx
#define RB1   PORTB,1               ;Tx
#define RB2   PORTB,2               ;Alarma (Se activa con cero)
#define RB3   PORTB,3               ;Panel
```

```
#define RB4   PORTB,4          ;Bajo   (Se activa con cero)
#define RB5   PORTB,5          ;Salida
#define RB6   PORTB,6          ;Load
#define RB7   PORTB,7          ;Zumbador
;-------------------------------------------------------------------------
#define RA0   PORTA,0
#define RA1   PORTA,1
#define RA2   PORTA,2          ;Canal de Conversión A/D
#define RA3   PORTA,3
#define RA4   PORTA,4          ;Alto (Se activa con cero)
#define RA5   PORTA,5
#define RA6   PORTA,6
#define RA7   PORTA,7
;---Para EEPROM-----------------------------------------------------
#define       SDA   PORTB,0
#define       SCL   PORTB,1
#define       a     .15
;-------------------------------------------------------------------------
        cblock 0x10             ;Creamos un espacio en la Memoria RAM
        Modulador
        Contador
        DirecLB
        DirecHB
        Horas
        Cobi
        switch
        Cor
        Segundazos
        Minutazos
        Tiempo
        Dia
        CByte
        Delaya
        Delayc
        DHB
        DLB
        i
        d
        Dato
        endc
#define AckBit        Cor,7
;-------------------------------------------------------------------------
        ORG           0
        goto          INIT
        nop
```

```
        nop
        bcf     INTCON,T0IF
        bsf     Cor,0
        call    INTERR
        movlw   .60
        movwf   TMR0
        bsf     STATUS,RP0
        movlw   b'00000111'
        movwf   OPTION_REG^0x80     ; Pre-escaler 1:256
        bcf     STATUS,RP0
        movlw   b'10100000'
        movwf   INTCON
        retfie
;=================================================================
INIT    bsf     STATUS,RP0
        movlw   b'00001111'
        movwf   TRISA^0x80          ;RA0-RA3, Entradas
        movlw   b'00000011'
        movwf   TRISB^0x80          ; RB7-RB2 Salidas
        clrf    ADCON1^0x80         ; C  anales A  nalógicos  RA0-
RA3
        bcf     STATUS,RP0
        clrf    ADRES
;
        movlw   .240
        movwf   Segundazos
        movlw   .5
        movwf   Minutazos
        movlw   .2
        movwf   d
        clrf    Tiempo
        clrf    Horas
        clrf    switch
        clrf    PORTA               ;Limpiamos  Puerto A
        clrf    PORTB
        clrf    Dia
        clrf    Cor
        clrf    CByte

        bsf     STATUS,RP0
        movlw   b'00000111'
        movwf   PTION_REG^0x80      ;Pre-escaler 1:256
        bcf     STATUS,RP0
        movlw   .60
        movwf   TMR0
```

```
        movlw   b'10100000'     ;Con Interrupcionesn edl timer0
        movwf   INTCON
;-------------------------------------------------------------------------------
;Cargado de Batería
;-------------------------------------------------------------------------------
        movlw   .20
        movwf   Modulador
        bsf     RB3             ;Activamos la carga de la Batería
;-------------------------------------------------------------------------------
TEST    bsf     RA4             ;Desactivamos nivel alto
        bcf     RB7             ;Desactivamos el Zumbador
        call    CONVER
        rrf     switch,0
        btfsc   STATUS,C
        goto    Fdia
        movf    ADRES,W
        sublw   Min
        btfsc   STATUS,C        ;Volt de Batería es < 11.0 V
        goto    Mosfet1         ;Si
;
ALAR    bsf     switch,0
        bcf     RB2             ;Activamos la alarma
        bsf     RB6             ;Activamos la Carga al Foco
        movf    ADRES,W                 ;No
        sublw   Minr
        btfsc   STATUS,C        ;Volt de Batería es < 11.5 V
        goto    Alarma          ;Si
        bsf     RB4             ;Desactivamos nivel bajo.
        bcf     RB7             ;Desactivamos el Zumbador
        movf    ADRES,W         ;No
        sublw   Max
        btfss   STATUS,C        ;Volt de Batería es > 13.5 V
        goto    PWM             ;Si
        goto    PWM1            ;No
;-------------------------------------------------------------------------------
;Rutina de espera de un segundo
;-------------------------------------------------------------------------------
Espera
        bcf     Cor,0
;

Loop1
        btfss   Cor,0
        goto    Loop1
        bcf     Cor,0
```

```
        decfsz      Contador,F
        goto        Loop1
        Return
;-------------------------------------------------------------------------------
;Rutina de Conversión
;-------------------------------------------------------------------------------
CONVER
        bcf         Cor,0
        movlw       b'01010001'
        movwf       ADCON0          ;activamos el Convertidor A/D
        bsf         ADCON0,GO_DONE
        btfsc       Cor,0
        goto        CONVER
Loop
        btfsc       ADCON0,GO_DONE
        goto        Loop
        btfsc       Cor,0
        goto        CONVER
        return
;-------------------------------------------------------------------------------
Mosfet1
        bcf         RB4             ;Activamos el Señailzor de nivel Bajo
        bsf         RB2             ;Desactivamos alarma
        bcf         RB7             ;desactivamos el Zumbador
        bcf         RB6             ;desactivamos la Carga
        goto        TEST
;-------------------------------------------------------------------------------
Alarma
        bsf         RB4             ;Desactivamos el  S eñalizador de  l
                                    nivel Bajo
        bsf         RB7             ;Activamos el Zumbador
        bsf         RB3             ;ACtivamos la carga panel batería.
        goto        TEST
;-------------------------------------------------------------------------------
Fdia
        movf        ADRES,W
        sublw       Minfd
        btfsc       STATUS,C        ;Volt de Batería es < Minfd
        goto        Mosfet1fd       ;Si
        goto        ALAR            ;no, seguir con el programa principal
;-------------------------------------------------------------------------------
Mosfet1fd
        bcf         RB4             ;Activamos el Señailzor de nivel Bajo
        bcf         RB7             ;desactivamos el Zumbador
        bcf         RB6             ;desactivamos la Carga
```

```
        clrf    switch
        bsf     Cor,6
        goto    TEST
;-------------------------------------------------------------------------------

PWM
        Bcf     RA4             ;Activamos e l S eñalizador de l ni vel
                                alto
        movf    Modulador,0
        btfss   STATUS,Z
        decf    Modulador,F     ;Decrementar  el an cho  de ni vel al to
                                en %5
        movf    Modulador,0
        movwf   Contador
        btfss   STATUS,Z
        bsf     RB3             ;Se activa la carga de la Batería
        movf    Modulador,0
        btfss   STATUS,Z
        call    Espera
;
        bcf     RB3             ;Se desactiva la carga de la Batería

        movf    Modulador,W
        sublw   .20
        movwf    Contador       ;Incrementar e l a ncho de  ni vel ba jo
                                en %5
        btfss   STATUS,Z
        call    Espera
        goto    TEST
;-------------------------------------------------------------------------------
PWM1

        movlw   .20
        xorwf   Modulador,W
        btfss   STATUS,Z        ;El modulador es 20
        incf    Modulador,F     ;No,  entonces i ncrementar el  an cho
                                de nivel alto en %5
        movf    Modulador,0     ;Si
        movwf   Contador
        bsf     RB3             ;Se Activa la carga de la Batería
        call    Espera
;       movlw . 20
        xorwf   Modulador,W
        btfss   STATUS,Z
        bcf     RB3             ;Desactivamos la Batería
```

```
        movf    Modulador,W
        sublw   .20
        movwf   Contador        ;Decrementar el  an cho  de ni vel ba jo
                                en %5
        btfss   STATUS,Z
        call    Espera
        goto    TEST
;--------------------------------------------------------------------------------
INTERR
        decfsz  Segundazos,1
        return
ITE
        movlw   .240
        movwf   Segundazos
        decfsz  Minutazos,1
        return
        movlw   .5
        movwf   Minutazos
        movf    ADRES,W
        decfsz  d,1
        movwf   Tiempo
        movlw   .0
        xorwf   d,0
        btfss   STATUS,Z
        return
        movlw   .2
        movwf   d
        movf    ADRES,0
        movwf   Dia
        btfsc   RB6
        call    EEPROM
        return
;--------------------------------------------------------------------------------

#include "piceep.ASM"
;--------------------------------------------------------------------------------
        END
```

RUTINA PARA LA COMUNICACION

Archivo: Piceep.asm

```
;------------EEPROM--------------
EEPROM
        Bcf             RB4
        movlw           .15
        movwf           i
        bcf             Cor,6
        bsf             STATUS,RP0
        bsf             SDA
        bcf             STATUS,RP0

EEPROM1
        Decf            i,1
        Btfsc           STATUS,Z
        return
        btfss           SDA
        goto            EEPROM1
        bsf             STATUS,RP0
        bcf             TRISB^0x80,1
        bcf             PORTB,0
        bcf             STATUS,RP0
        movlw           .248
        movwf           Cobi
        movlw           .2
        movwf           i
;--------------------------------------------------------------------------------

MAIN
        call            READ
        movf            DHB,0
        movwf           DirecHB
        movf            DLB,0
        movwf           DirecLB

MAIN1
        Bcf             AckBit
        Call            START
        movlw           b'10100000'             ;Operaciónd de Escritura
        movwf           CByte
        call            CB
        call            ACK
        btfsc           AckBit
```

```
        goto    MAIN1
        call    AHB
        call    ACK
        btfsc   AckBit
        goto    MAIN1
        call    ALB
        call    ACK
        btfsc   AckBit
        goto    MAIN1
        call    MENSAJE
        call    ACK
        btfsc   AckBit
        goto    MAIN1
        call    STOP
        decfsz  i,1
        goto    DIRECCION
        goto    REANUDA
        bsf     RB4
        return
;--------------------------------------------------------------------------

START
        Bsf     SDA
        bcf     SCL             ;Activamos el reloj
        call    DELAYnus
        bsf     SCL
        call    DELAYnus
        Bcf     SDA             ;Originamos la condición de inicio
        call    DELAYnus        ;Esperamos 2499us
        Bcf     SCL             ;Ponemos a cero el reloj
        Call    DELAYnus
        return
;--------------------------------------------------------------------------
CB
        Rlf     CByte,1
        Btfsc   STATUS,C
        bsf     SDA
        btfss   STATUS,C
        bcf     SDA
        call    DELAYnus
        bsf     SCL
        call    DELAY2nus       ;nuevo
        bcf     SCL
        call    DELAYnus
        incfsz  Cobi,1
```

```
        goto    CB
        bsf     STATUS,RP0
        bsf     PORTB,0         ;Para el bit de reconocimiento
        bcf     STATUS,RP0
        movlw   .248
        movwf   Cobi
        return
;-------------------------------------------------------------------------
ACK
        bsf     STATUS,RP0
        bsf     PORTB,0         ;Para el bit de reconocimiento
        bcf     STATUS,RP0
        call    DELAYnus
        bsf     SCL
        btfsc   SDA
        bsf     AckBit
        call    DELAY2nus
        btfsc   SDA
        bsf     AckBit
        bcf     SCL
        bsf     STATUS,RP0
        bcf     PORTB,0         ;Para el bit de reconocimiento
        bcf     STATUS,RP0
        call    DELAYnus
        return
;-------------------------------------------------------------------------
ACKM
        bsf     STATUS,RP0
        bcf     PORTB,0         ;Para el bit de reconocimiento
        bcf     STATUS,RP0
        bcf     SDA
        call    DELAYnus
        bsf     SCL
        call    DELAY2nus
        bcf     SCL
        call    DELAYnus
        return
;-------------------------------------------------------------------------
RDA
        bsf     STATUS,RP0
        bsf     PORTB,0
        bcf     STATUS,RP0

RDDHB
        call    DELAYnus
```

```
        bsf     SCL
        call    DELAYnus
        rlf     Dato,1
        btfsc   SDA
        bsf     Dato,0
        btfss   SDA
        bcf     Dato,0
        call    DELAYnus
        bcf     SCL
        call    DELAYnus
        incfsz  Cobi,1
        goto    RDDHB
        movlw   .248
        movwf   Cobi
        return
;-------------------------------------------------------------------------------
NOACK
        bsf     STATUS,RP0
        bcf     PORTB,0                 ;Para el bit de reconocimiento
        bcf     STATUS,RP0
        bsf     SDA
        call    DELAYnus
        bsf     SCL
        call    DELAY2nus
        bcf     SCL
        call    DELAYnus
        return
;-------------------------------------------------------------------------------

;ADDRESS HIGHT BYTE

AHB
        rlf     DHB,1           ;
        btfss   STATUS,C
        bcf     SDA
        btfsc   STATUS, C       ;
        bsf     SDA             ;
        call    DELAYnus
        bsf     SCL             ;
        call    DELAY2nus
        bcf     SCL
        call    DELAYnus        ;
        incfsz  Cobi,1
        goto    AHB
        movlw   .248
```

```
        movwf   Cobi
        return

;-------------------------------------------------------------------------------
;ADDRESS LOW BYTE

ALB
        rlf     DLB,1           ;
        btfss   STATUS,C
        bcf     SDA             ;
        btfsc   STATUS, C       ;
        bsf     SDA             ;
        call    DELAYnus
        bsf     SCL
        call    DELAY2nus
        bcf     SCL             ;
        call    DELAYnus
        incfsz  Cobi,1
        goto    ALB
        movlw   .248
        movwf   Cobi
        return
;-------------------------------------------------------------------------------
MENSAJE
TIME
        rlf     Tiempo, 1       ;
        btfss   STATUS,C
        bcf     SDA             ;
        btfsc   STATUS, C       ;
        bsf     SDA             ;
        call    DELAYnus
        bsf     SCL             ;
        call    DELAY2nus
        bcf     SCL
        call    DELAYnus        ;
        incfsz  Cobi,1
        goto    TIME
        movlw   .248
        movwf   Cobi
        call    ACK



DAY
        rlf     Dia,1           ;
```

```
        btfss       STATUS,C
        bcf         SDA             ;
        btfsc       STATUS,C        ;
        bsf         SDA             ;
        call        DELAYnus
        bsf         SCL             ;
        call        DELAY2nus
        bcf         SCL
        call        DELAYnus        ;
        incfsz      Cobi,1
        goto        DAY
        movlw       .248
        movwf       Cobi
        return
;-------------------------------------------------------------------------
STOP
        Bcf         SDA             ;Ponemos a cero la línea SDA
        call        DELAYnus
        Bsf         SCL             ;Ponemos a uno el reloj
        call        DELAYnus
        Bsf         SDA             ;Ponemos a uno SDA
        call        DELAYnus
        return                      ; Ponemos a cero el reloj
;-------------------------------------------------------------------------
DELAYnus
        movlw       .255
        movwf       Delaya
        movlw       .255
        movwf       Delayc

XDELAYnus
        decfsz      Delaya,1
        goto        XDELAYnus
        movlw       a
        movwf       Delaya
        decfsz      Delayc,1
        goto        XDELAYnus
        return
;-------------------------------------------------------------------------
DELAY2nus
        call        DELAYnus
        call        DELAYnus
        return
;-------------------------------------------------------------------------
DIRECCION
```

```
        incf        DirecLB,1
        btfsc       STATUS,2
        incf        DirecHB,1
        incf        DirecLB,1
        btfsc       STATUS,2
        incf        DirecHB,1
        movf        DirecHB,0
        movwf       Tiempo
        movf        DirecLB,0
        movwf       Dia
        call        DELAY2nus
        clrf        DHB
        clrf        DLB
        goto        MAIN1
;--------------------------------------------------------------------------
REANUDA
        clrf        Tiempo
        clrf        Dia
        return
;--------------------------------------------------------------------------
#include    "read.asm"
            END
```

Archivo: RPointer.asm

```
;Programa para leer secuencialmente una memoria 24LC32A
READ
        bcf         AckBit
        clrf        DHB
        clrf        DLB
        call        START
        movlw       b'10100000'         ;Operación de Escritura
        movwf       CByte
        call        CB
        call        ACK
        btfsc       AckBit
        goto        READ
        call        AHB
        call        ACK
        btfsc       AckBit
        goto        READ
        call        ALB
        call        ACK
        btfsc       AckBit
        goto        READ
```

```
call        START
movlw       b'10100001'         ;Operaciónd de Lectura
movwf       CByte
call        CB
call        ACK
btfsc       AckBit
goto        READ
call        RDA
call        ACKM
movf        Dato,0
movwf       DHB
call        RDA
call        NOACK
movf        Dato,0
movwf       DLB
call        STOP
return
```

APÉNDICE C

CÓDIGO PRINCIPAL DE LOS PROGRAMAS EN MICROSOFT VISUAL C++

RUTINAS PARA EL ACCESO A LA TARJETA DE ADQUSICIÓN DE DATOS

Archivo: I2CFDLG.h

```
#include "wsc.h"
#define		HSDA SioDTR(COM1,'C')
#define		LSDA  SioDTR(COM1,'S')
#define		HSCL  SioRTS(COM1,'C')
#define		LSCL  SioRTS(COM1,'S')
#define		READ SioCTS(COM1)
#define		Tiempo() GetCurrentTime()
#define		Reset   SioReset(COM1,1024,1024)


class CSolar
{
private:
	int code;
	void ms(int);

public:

	char Dato[1];
	int n, w;
	float r;
	CSolar::CSolar(int frec= 1);
	void Inicio(void);
	void Parada(void);
	void Enviar(CString buffer);
	int Ack(void);
	void Ackm(void);
	void NoAck(void);
	CString Recibir(void);
}
```

Archivo: I2CFDLG.cpp

```
#include "stdafx.h"
#include "I2C.h"
#include "wsc.h"
#include "math.h"
#include "I2CFDlg.h"
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
```

```
static char THIS_FILE[] = __FILE__;
#endif

CSolar::CSolar(int frec)
{
	n=frec;
}
void CSolar::Inicio(void)
{
	HSDA;
	LSCL;
	HSCL;
	ms(3*n);
	LSDA;
	ms(2*n);
	LSCL;
	ms(3*n);
}
void CSolar::Enviar(CString buffer)
{
	int t=0;
	CString s="0";
	CString s1="1";
	CString s2="0";
	while(t<8)
		{
			s.SetAt(0,buffer[t]);
			if (s==s1 )
			{
				HSDA;
			}
			if (s==s2 )
			{
				LSDA;
			}
				ms(2*n);
				HSCL;
				ms(5*n);
				LSCL;
				ms(3*n);
				t=t+1;

		}
}
```

```
CString CSolar::Recibir(void)
{
	int t=0;
	CString Dato("00000000");
	r=0;
	while(t<8)

		{
			LSDA;
			ms(2*n);
			HSCL;
			ms(3*n);

		code = READ;
		if (code>0 )
		{
			Dato.SetAt(t,'0');;

		}
		if (code==0 )
		{
			Dato.SetAt(t,'1');
			r=r+(float)pow(2,7-t);
		}
			ms(2*n);
			LSCL;
			ms(3*n);
			t=t+1;
		}
return Dato;
}

int CSolar::Ack(void)
{
	LSDA;
	ms(2*n);
	HSCL;
	ms(3*n);
	code = READ;
	if (code==0)
	{
		w=1;
	}
	if (code>0)
	{
```

```
                w=0;
        }
        ms(2*n);
        LSCL;
        ms(3*n);
        return w;
}

void CSolar::Ackm(void)
{
        LSDA;
        ms(2*n);
        HSCL;
        ms(5*n);
        LSCL;
        ms(3*n);
}

void CSolar::NoAck(void)
{
        HSDA;
        ms(2*n);
        HSCL;
        ms(5*n);
        LSCL;
        ms(3*n);
}

void CSolar::Parada(void)
{
        LSDA;
        ms(2*n);
        HSCL;
        ms(3*n);
        HSDA;
        ms(2*n);
}

void CSolar::ms(int temp)
{
long wait;
wait = (long)temp + (long)Tiempo();
while (wait > (long)Tiempo());
}
```

RUTINA PARA LA COMUNICACION

Archivo: I2CDLG.cpp

```
// I2CDlg.cpp : implementation file
//

#include "stdafx.h"
#include "I2C.h"
#include "I2CDlg.h"
#include "I2CFDlg.h"
#include "DOS.h"
#include <stdio.h>
#include <process.h>
#ifdef _DEBUG
#define new DEBUG_NEW
#undef THIS_FILE
static char THIS_FILE[] = __FILE__;
#endif

/////////////////////////////////////////////////////////////////////////////
// CAboutDlg dialog used for App About

FILE *stream;
CString buffer1;
CString buffer2;
float value;
int sw=0;

class CAboutDlg : public CDialog
{
public:
        CAboutDlg();

// Dialog Data
        //{{AFX_DATA(CAboutDlg)
        enum { IDD = IDD_ABOUTBOX };
        //}}AFX_DATA

        // ClassWizard generated virtual function overrides
        //{{AFX_VIRTUAL(CAboutDlg)
        protected:
        virtual voi d D oDataExchange(CDataExchange*  pDX);     / / D DX/DDV
support
        //}}AFX_VIRTUAL
```

```
// Implementation
protected:
	//{{AFX_MSG(CAboutDlg)
	//}}AFX_MSG
	DECLARE_MESSAGE_MAP()
};

CAboutDlg::CAboutDlg() : CDialog(CAboutDlg::IDD)
{
	//{{AFX_DATA_INIT(CAboutDlg)
	//}}AFX_DATA_INIT
}

void CAboutDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CAboutDlg)
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CAboutDlg, CDialog)
	//{{AFX_MSG_MAP(CAboutDlg)
		// No message handlers
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CI2CDlg dialog

CI2CDlg::CI2CDlg(CWnd* pParent /*=NULL*/)
	: CDialog(CI2CDlg::IDD, pParent)
{
	//{{AFX_DATA_INIT(CI2CDlg)
	m_mensaje = _T("");
	m_Bytes = _T("");
	m_NDatos = _T("");
	m_ah = _T("");
	m_bufferbits = _T("");
	m_bufferDirecG = _T("");
	m_bufferDirecLA = _T("");
	//}}AFX_DATA_INIT
	// Note that LoadIcon does not require a subsequent DestroyIcon in Win32
	m_hIcon = AfxGetApp()->LoadIcon(IDR_MAINFRAME);
}
```

```
void CI2CDlg::DoDataExchange(CDataExchange* pDX)
{
	CDialog::DoDataExchange(pDX);
	//{{AFX_DATA_MAP(CI2CDlg)
	DDX_Text(pDX, IDC_EDIT1, m_mensaje);
	DDX_Text(pDX, IDC_Bytes, m_Bytes);
	DDX_Text(pDX, IDC_EDIT2, m_NDatos);
	DDX_CBString(pDX, IDC_AH, m_ah);
	DDV_MaxChars(pDX, m_ah, 8);
	DDX_Text(pDX, IDC_DAES, m_bufferbits);
	DDX_Text(pDX, IDC_DAES2, m_bufferDirecG);
	DDV_MaxChars(pDX, m_bufferDirecG, 8);
	DDX_Text(pDX, IDC_DAES3, m_bufferDirecLA);
	DDV_MaxChars(pDX, m_bufferDirecLA, 8);
	//}}AFX_DATA_MAP
}

BEGIN_MESSAGE_MAP(CI2CDlg, CDialog)
	//{{AFX_MSG_MAP(CI2CDlg)
	ON_WM_SYSCOMMAND()
	ON_WM_PAINT()
	ON_WM_QUERYDRAGICON()
	ON_BN_CLICKED(IDC_Calc, OnCalc)
	ON_BN_CLICKED(IDC_LEER, OnLeer)
	ON_BN_CLICKED(IDC_Res, OnRes)
	ON_BN_CLICKED(IDC_Res2, OnRes2)
	ON_BN_CLICKED(IDC_LEERDATOS, OnLeerdatos)
	ON_BN_CLICKED(IDC_EepromInit, OnEepromInit)
	ON_BN_CLICKED(IDC_Pot, OnPot)
	ON_BN_CLICKED(IDC_LEERRANDOM, OnLeerrandom)
	ON_BN_CLICKED(IDC_GRABAR, OnGrabar)
	ON_BN_CLICKED(IDC_ARCHIVAR, OnArchivar)
	//}}AFX_MSG_MAP
END_MESSAGE_MAP()

/////////////////////////////////////////////////////////////////////////////
// CI2CDlg message handlers
```

```
BOOL CI2CDlg::OnInitDialog()
{
	CDialog::OnInitDialog();

	// Add "About..." menu item to system menu.

	// IDM_ABOUTBOX must be in the system command range.
	ASSERT((IDM_ABOUTBOX & 0xFFF0) == IDM_ABOUTBOX);
	ASSERT(IDM_ABOUTBOX < 0xF000);

	CMenu* pSysMenu = GetSystemMenu(FALSE);
	if (pSysMenu != NULL)
	{
		CString strAboutMenu;
		strAboutMenu.LoadString(IDS_ABOUTBOX);
		if (!strAboutMenu.IsEmpty())
		{
			pSysMenu->AppendMenu(MF_SEPARATOR);
			pSysMenu->AppendMenu(MF_STRING,
IDM_ABOUTBOX, strAboutMenu);
		}
	}

	// Set the icon for this dialog.  The framework does this automatically
	//  when the application's main window is not a dialog
	SetIcon(m_hIcon, TRUE);			// Set big icon
	SetIcon(m_hIcon, FALSE);		// Set small icon

	// TODO: Add extra initialization here

	return TRUE;  // return TRUE  unless you set the focus to a control
}

void CI2CDlg::OnSysCommand(UINT nID, LPARAM lParam)
{
	if ((nID & 0xFFF0) == IDM_ABOUTBOX)
	{
		CAboutDlg dlgAbout;
		dlgAbout.DoModal();
	}
	else
	{
		CDialog::OnSysCommand(nID, lParam);
	}
}
```

```
// If you add a minimize button to your dialog, you will need the code below
//  to draw the icon.  For MFC applications using the document/view model,
//  this is automatically done for you by the framework.

void CI2CDlg::OnPaint()
{
        if (IsIconic())
        {
                CPaintDC dc(this); // device context for painting

                SendMessage(WM_ICONERASEBKGND, (              WPARAM)
dc.GetSafeHdc(), 0);

                // Center icon in client rectangle
                int cxIcon = GetSystemMetrics(SM_CXICON);
                int cyIcon = GetSystemMetrics(SM_CYICON);
                CRect rect;
                GetClientRect(&rect);
                int x = (rect.Width() - cxIcon + 1) / 2;
                int y = (rect.Height() - cyIcon + 1) / 2;

                // Draw the icon
                dc.DrawIcon(x, y, m_hIcon);
        }
        else
        {
                CDialog::OnPaint();
        }
}

// The system calls this to obtain the cursor to display while the user drags
//  the minimized window.
HCURSOR CI2CDlg::OnQueryDragIcon()
{
        return (HCURSOR) m_hIcon;
}

void CI2CDlg::OnCalc()
{
                WinExec("calc.exe",SW_SHOW);
}
```

```
void CI2CDlg::OnLeer()
{
	if (sw==0)
	{
	MessageBox("Primero debe inicializar el Puerto");
	}

	if (sw==1)
	{
	CSolar eeprom(1);
	char buf[200];

	if (eeprom.Ack() ==0)
	{
	MessageBox("Conecte el Data Loger al Puerto Serial");
	}

	if (eeprom.Ack() ==1)
	{

	LECTURA:

		eeprom.Inicio();
		buffer1="10100000";
		eeprom.Enviar(buffer1);
		if (eeprom.Ack() ==1)
		{
		goto LECTURA;
		}

		buffer1="00000000";
		eeprom.Enviar(buffer1);
		if (eeprom.Ack() ==1)
		{
		goto LECTURA;
		}

		buffer1="00000000";
		eeprom.Enviar(buffer1);
		if (eeprom.Ack() ==1)
		{
		goto LECTURA;
		}
```

```
                eeprom.Inicio();
                buffer1="10100001";
                eeprom.Enviar(buffer1);
                if (eeprom.Ack() ==1)
                {
                goto LECTURA;
                }
                buffer2 = eeprom.Recibir();
                eeprom.Ackm();
                buffer1 = eeprom.Recibir();
                eeprom.NoAck();
                eeprom.Parada();
                value= eeprom.r;

                sprintf( buf,"Datos: %d", int(value-2) );
                m_NDatos=buf;
                m_mensaje = " El puntero esta en la dirección: " + buffer2 + " " +
                buffer1;
                UpdateData(FALSE);
                }
        }
}


void CI2CDlg::OnRes()
{

        if (sw==1)
        {
        MessageBox("El Puerto ya fue inicializado, para reinicializar pulse Reset y
        luego Inicialice Puerto");
        }

        if(sw==0)
        {
        m_mensaje ="";
        m_Bytes ="";
        m_NDatos ="";
        m_ah = "";
        m_bufferbits = "";
        m_bufferDirecG = "";
        m_bufferDirecLA = "";
        sw=1;
```

```
	int code;
	code = SioReset(COM1,1024,1024);
		if(code<0)
		{
		char Temp[80];
		SioWinError((LPSTR)Temp,80);
		m_mensaje =Temp;
		UpdateData(FALSE);
		SioDone(COM1);
		exit(1);
		}
		m_mensaje ="Puerto inicializado";
		UpdateData(FALSE);
	}
}

void CI2CDlg::OnRes2()
{
	sw=0;
	SioDone(COM1);
	m_mensaje ="Puerto Reseteado";
	UpdateData(FALSE);
}

void CI2CDlg::OnLeerdatos()
{
	if (sw==0)
	{
	MessageBox("Primero debe inicializar el Puerto");
	}

if(sw==1)
	{
	CSolar eeprom(1);
	CString buffer3;
	CString buffer4="Voltajes: ";
	int i=int(value)-2;
	char buf[200];

		if (eeprom.Ack() ==0)
		{
		MessageBox("Conecte el Data Loger al Puerto Serial");
		}
```

```
            if (eeprom.Ack() ==1)
            {
        LECTURA1:
            eeprom.Inicio();
            buffer1="10100000";
            eeprom.Enviar(buffer1);
            if (eeprom.Ack() ==1)
            {
            goto LECTURA1;}
            buffer1="00000000";
            eeprom.Enviar(buffer1);
            if (eeprom.Ack() ==1)
            {
            goto LECTURA1;}
            buffer1="00000010";
            eeprom.Enviar(buffer1);
            if (eeprom.Ack() ==1)
            {
            goto LECTURA1;}
            eeprom.Inicio();
            buffer1="10100001";
            eeprom.Enviar(buffer1);
            if (eeprom.Ack() ==1)
            {
            goto LECTURA1;}
            while(i>0)
            {
            i=i-1;
            buffer3 = eeprom.Recibir();
            eeprom.Ackm();
            value = float((eeprom.r)*17.5/255);
            sprintf( buf, "%f", value );
            buffer3= buf;
            buffer4 = buffer4  + buffer3.Left(5)+ " ";
            }
            m_Bytes= buffer4;
            eeprom.NoAck();
            eeprom.Parada();
            m_mensaje = " Lectura de Datos Terminada ";
            buffer1 = buffer4.Right(buffer4.GetLength()-9);
            UpdateData(FALSE);
            }
        }
}
```

```
void CI2CDlg::OnEepromInit()
{
	if (sw==0)
	{
	MessageBox("Primero debe inicializar el Puerto");
	}

	if(sw==1)
	{
	CSolar eeprom(1);
	CString buffer3;
	CString	buffer4;

		if (eeprom.Ack() ==0)
		{
			MessageBox("Conecte el Data Loger al Puerto Serial");
		}


		if (eeprom.Ack() ==1)
		{
	ESCRITURA:
		eeprom.Inicio();
		buffer3="10100000";
		eeprom.Enviar(buffer3);
		if (eeprom.Ack() ==1)
		{
		goto ESCRITURA;
		}

		buffer4="00000000";
		eeprom.Enviar(buffer4);
		if (eeprom.Ack() ==1)
		{
		goto ESCRITURA;
		}

		buffer3="00000000";
		eeprom.Enviar(buffer3);
		if (eeprom.Ack() ==1)
		{
		goto ESCRITURA;
		}
```

```
                buffer3="00000000";
                eeprom.Enviar(buffer3);
                if (eeprom.Ack() ==1)
                {
                goto ESCRITURA;
                }
                buffer4="00000010";
                eeprom.Enviar(buffer4);
                if (eeprom.Ack() ==1)
                {
                goto ESCRITURA;
                }
                eeprom.Parada();
                m_mensaje = "Memoria Inicializada" ;
                UpdateData(FALSE);

                }
        }
}


void CI2CDlg::OnPot()
{
        if (sw==0)
        {
        MessageBox("Primero debe inicializar el Puerto");
        }

        if (sw==1)
        {

        UpdateData(TRUE);
        CString AH1="200";
        CString AH2="150";
        CString AH3="100";
        CString AH4="50";
        CString AH5="25";
        float Pot;
        char buf[200];

                if(m_ah==AH1)
                {
                Pot = float(12.5  * (200*(10/value)));
                sprintf( buf,     "Potencia: %f", Pot );
                m_mensaje=buf;
```

```
                MessageBox("Guardando en Base de Datos");
                }

                if(m_ah== AH2)
                {
                Pot = float(12.5  * (150*(10/value)));
                sprintf( buf,     "Potencia: %f", Pot );
                m_mensaje=buf;
                MessageBox("Guardando en Base de datos");
                }

                if(m_ah== AH3)
                {
                Pot = float(12.5  * (100*(10/value)));
                sprintf( buf,     "Potencia: %f", Pot );
                m_mensaje=buf;
                MessageBox("Guardando en Base de datos");
                }
                if(m_ah==AH4)
                {
                Pot = float(12.5  * (100*(10/value)));
                sprintf( buf,     "Potencia: %f", Pot );
                 m_mensaje=buf;
                MessageBox("Guardando en Base de datos");
                }

                if(m_ah==AH5)
                {
                Pot = float(12.5  * (25*(10/value)));
                sprintf( buf,     "Potencia: %f", Pot );
                m_mensaje=buf;
                MessageBox("Guardando en Base de datos");
                }
                UpdateData(FALSE);
        }
}

void CI2CDlg::OnLeerrandom()
{
                if (sw==0)
                {
                MessageBox("Primero debe inicializar el Puerto");
                }
```

```
        if(sw==1)
        {
        CSolar eeprom(1);
        CString buffer3;
        UpdateData(TRUE);
        if (eeprom.Ack() ==0)
        {
        MessageBox("Conecte el Data Loger al Puerto Serial");
        }

        if (eeprom.Ack() ==1)
        {
LECTURA1:

        eeprom.Inicio();
        buffer1="10100000";
        eeprom.Enviar(buffer1);
        if (eeprom.Ack() ==1)
        {
        goto LECTURA1;
        }
        buffer1="00000000";
        eeprom.Enviar(buffer1);
        if (eeprom.Ack() ==1)
        {
        goto LECTURA1;
        }
        eeprom.Enviar(m_bufferDirecLA);
        if (eeprom.Ack() ==1)
        {
        goto LECTURA1;
        }
        eeprom.Inicio();
        buffer1="10100001";
        eeprom.Enviar(buffer1);
        if (eeprom.Ack() ==1)
        {
        goto LECTURA1;
        }
        buffer3 = eeprom.Recibir();
        eeprom.NoAck();
        eeprom.Parada();
        m_Bytes=buffer3;
        m_mensaje = " Lectura Aleatoria Terminada ";
        UpdateData(FALSE);
```

```
                }
        }
}

void CI2CDlg::OnGrabar()
{
        if (sw==0)
        {
        MessageBox("Primero debe inicializar el Puerto");
        }

        if(sw==1)
        {
        CSolar eeprom(1);
        CString buffer3;
        CString         buffer4;
        UpdateData(TRUE);

                if (eeprom.Ack() ==0)
                {
                MessageBox("Conecte el Data Loger al Puerto Serial");
                }

                if (eeprom.Ack() ==1)
                {
                ESCRITURA:
                eeprom.Inicio();
                buffer3="10100000";
                eeprom.Enviar(buffer3);
                if (eeprom.Ack() ==1)
                {
                goto ESCRITURA;
                }
                buffer4="00000000";
                eeprom.Enviar(buffer4);
                if (eeprom.Ack() ==1)
                {
                goto ESCRITURA;
                }
                eeprom.Enviar(m_bufferDirecG);
                if (eeprom.Ack() ==1)
                {
                goto ESCRITURA;
                }
```

```
                eeprom.Enviar(m_bufferbits);
                if (eeprom.Ack() ==1)
                {
                goto ESCRITURA;
                }
                eeprom.Parada();
                m_mensaje = "Dato Grabado en la Dirección dada" ;
                UpdateData(FALSE);
                }
        }
}

void CI2CDlg::OnArchivar()
{
         int i =buffer1.GetLength()/6;
        char   s[] = "Voltajes de Batería cada minuto de descarga";
        char   c = '\n';
        stream = fopen( "VoltdeBat.I2C", "w" );
        fprintf( stream, "%s%c", s, c );
        while(i >0)
        {
        fprintf( stream, "%s\n", buffer1.Left(6));
        buffer1= buffer1.Right(buffer1.GetLength()-6);
        i = i-1;
        }
        fclose( stream );
        system( "type VoltdeBat.I2C" );

}
```