ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

CONSTRUCCIÓN DE UN PROTOTIPO DE CONTROL Y REGISTRO DE PERSONAL BASADO EN IDENTIFICACIÓN POR RADIOFRECUENCIA.

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE TECNÓLOGO EN ELECTRÓNICA Y TELECOMUNICACIONES

DANIELA DEL PILAR CHANGO ZURITA

daniela.chango74 @gmail.com

DIRECTOR: ING ALCÍVAR COSTALES

alcivar.costales @epn.edu.ec

QUITO, ENERO 2013

DECLARACIÓN

Yo, Daniela del Pilar Chango Zurita, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentado para ningún grado o calificación profesional y que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mi derecho de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su reglamento y por la normatividad institucional vigente.

_____

Daniela del Pilar Chango Zurita

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por la Señorita Daniela del Pilar Chango Zurita, bajo mi supervisión.

_____

Ing. Alcívar Costales

DIRECTOR DE PROYECTO

AGRADECIMIENTO

En primer lugar gradezco a Dios quien guio y colmo de bendiciones mi camino durante el tiempo que realice mis estudios, que a pesar de las adversidades que se presentaron durante el mismo supo darme fuerzas para culminar una de las metas trazadas en mi vida.

A mis padres, quienes incentivaron en mí el sentido de superación y responsabilidad, el apoyo incondicional en los malos y buenos momentos que me ayudaron a tomar decisiones importantes y siempre están ahí cuando más los necesito.

A dos personas especiales que llegaron a mi vida y me apoyaranon en todo momento cada uno a su manera, son dos hombres excepcionales a los cuales les tengo mucha admiración y respeto por su capacidad y deseos de superación.

DEDICATORIA

Dedico este trabajo a Dios por haberme dado la vida, el cual me hizo ver al mundo desde una perspectiva diferente, además de hacerme comprender que existe un único sentimiento puro y verdadero el cual es el amor de Dios, Señor eres mi fortaleza en los momentos de angustia, tristeza y debilidad tú siempre estás conmigo a pesar de todo pendiente de mis acciones, cuidándome y protegiéndome, para ti va dedicado este logro.

A mi padre Pedro Chango ya que sin su apoyo no lo hubiese logrado, gracias por nunca perder la fe en mí y haberme tenido tanta pasiencia, siempre lo llevo en mi corazón como lo más importante en mi vida.

Al hombre que llego a mi vida y la cambio por completo, quien me ha enseñdo tantas cosas que valoro, quien me apoyado en todo momento de tantas maneras a pesarde todas las dificultades este alegría por culminar esta meta se la dedico.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ÍNDICE DE ANEXOS

CAPÍTULO I

FUNDAMENTO TEÓRICO

1.1 IDENTIFICACIÓN POR RADIOFRECUENCIA "RFID"[1]

1.1.1 INTRODUCCIÓN

En la actualidad, la tecnología más extendida para la identificación de objetos es la de los códigos de barras. Sin embargo, éstos presentan algunas desventajas, como la escasa cantidad de datos que pueden almacenar y la imposibilidad de ser reprogramados. La mejora ideada constituyó el origen de la tecnología RFID; consistía en usar chips de silicio que pudieran transferir los datos que almacenaban al lector sin contacto físico, de forma equivalente a los lectores de infrarrojos utilizados para leer los códigos de barras.

La RFID puede utilizarse para cubrir diferentes necesidades. Es ideal para sectores internos de una compañía y áreas logísticas. Las mayores dificultades que afrontan los estándares RFID actualmente son: las medidas anti-choque para evitar que varias etiquetas se lean en forma simultánea, la lectura de las etiquetas a través de líquidos, la aprobación lenta de los estándares, la re-evaluación de antiguos procedimientos, los problemas éticos y de seguridad.

1.1.2 RESEÑA HISTÓRICA

Se ha sugerido que el primer dispositivo conocido similar a RFID pudo haber sido una herramienta de espionaje inventada por Léon Theremin para el gobierno soviético en 1945. El dispositivo de Theremin era un dispositivo de escucha secreto pasivo, no una etiqueta de identificación, por lo que esta aplicación es dudosa. Según algunas fuentes, la tecnología usada en RFID habría existido desde comienzos de los años 1920, desarrollados por el MIT y usados extensivamente por los británicos en la Segunda Guerra Mundial (fuente que establece que los *sistemas* RFID han existido desde finales de los años 1960 y

---

[1] http://es.wikipedia.org/wiki/RFID

que sólo recientemente se había popularizado gracias a las reducciones de costos).

Otro trabajo temprano que trata el RFID es el artículo de 1948 de Harry Stockman, titulado “Comunicación por medio de la energía reflejada” (Actas del IRE, pp. 1196-1204, octubre de 1948). Stockman predijo que “... el trabajo considerable de investigación y de desarrollo tiene que ser realizado antes de que los problemas básicos restantes en la comunicación de la energía reflejada se solucionen, y antes de que el campo de aplicaciones útiles se explore.” Hicieron falta treinta años de avances en multitud de campos diversos antes de que RFID se convirtiera en una realidad.

1.1.3 DEFINICIÓN DE RFID

RFID (siglas de *Radio Frequency IDentification*, en español identificación por radiofrecuencia) es un sistema de almacenamiento y recuperación de datos remoto que usa dispositivos denominados etiquetas, tarjetas, transpondedores o tags RFID. El propósito fundamental de la tecnología RFID es transmitir la identidad de un objeto (similar a un número de serie único) mediante ondas de radio. Las tecnologías RFID se agrupan dentro de las denominadas Auto ID (*automatic identification*, o identificación automática).

Esta tecnología se utiliza para identificar un elemento, seguir su ruta de movimiento y calcular distancias gracias a una etiqueta especial que emite ondas de radio, la cual se adjunta o se encuentra incorporada al objeto. La tecnología RFID permite la lectura de etiquetas incluso cuando éstas no se encuentran en una línea visual directa y puede además penetrar finas capas de materiales (pintura, nieve, etc.).

1.1.4 CARACTERÍSTICAS DE UN SISTEMA RFID

Un sistema RFID puede estar caracterizado de tres formas basado en los siguientes atributos:

- Frecuencia Operacional

- Rango de Lectura
- Método de Conexión Físico

1.1.4.1 Caracterización Basada en la Frecuencia Operacional

La frecuencia de operación es el atributo más importante de un sistema RFID. Es la frecuencia a la cual los lectores transmiten su señal. Está cercanamente asociado con el típico atributo de la distancia de lectura. En la mayoría de los casos, la frecuencia de un sistema RFID está determinada por su típico requerimiento de la distancia de lectura.

Los diferentes tipos de frecuencias utilizadas en RFID son las siguientes:

- Baja Frecuencia (LF)
- Alta Frecuencia (HF)
- Ultra Alta Frecuencia (UHF)
- Microondas

1.1.4.2 Caracterización Basada en el Rango de Lectura

El rango de lectura de un sistema RFID está definido como la distancia de lectura entre la etiqueta y el lector. Usando este criterio, un sistema RFID puede estar divido entre los siguientes tres tipos:

1.1.4.2.1 Rango Cercano

El rango de lectura es menor que 1cm. Operan a frecuencias de LF y HF.

1.1.4.2.2 Rango Remoto

El rango de lectura es de 1cm a 100cm. Operan a frecuencias de LF y HF.

1.1.4.2.3 Rango Largo

El rango de lectura es mayor a 100cm. Operan a frecuencias de UHF y microondas.

1.1.4.3 Caracterización Basada en el Método de Conexión Físico

La conexión física se refiere al método usado para enganchar la etiqueta con la antena (eso es, el mecanismo por el cual la energía es transferida a la etiqueta desde la antena). Basados en este criterio, son posibles tres tipos de sistemas RFID:

- Magnético
- Eléctrico
- Electromagnético

1.1.5 MODO DE FUNCIONAMIENTO DE LA TECNOLOGÍA RFID

Para la creación de un sistema RFID hay que tener en cuenta diversos factores de diseño como el rango de alcance donde se puede mantener la comunicación, la velocidad de flujo de datos que podemos obtener entre lector y etiqueta, el tamaño físico de la etiqueta, la habilidad del lector para mantener la comunicación con varias etiquetas a la vez o la robustez que ofrece la comunicación a posibles interferencias de materiales entre lector y etiqueta. Se debe tener en cuenta también el nivel de emisión para no sobrepasar las regulaciones impuestas en cada país, si existe una batería suplementaria para realizar la comunicación entre etiqueta y lector o la frecuencia portadora RF usada en la comunicación entre lector y transponder.

El funcionamiento del sistema, es bastante sencillo, como podemos observar en la figura 1.1, el lector envía una serie de ondas de radiofrecuencia al tag, que son captadas por la microantena de éste. Dichas ondas activan el microchip, el cual, a través de la microantena y mediante ondas de radiofrecuencia, transmite al lector la información que tengan en su memoria. Finalmente, el lector recibe la información que tiene el tag y lo envía a una base de datos en la que previamente se han registrado las características del producto o puede procesarlo según convenga a cada aplicación. La comunicación entre el lector y la etiqueta se realiza mediante señales de radiofrecuencia a una determinada frecuencia que generan las antenas de lector y etiqueta, estas frecuencias pueden ser iguales o pueden ser armónicos. La comunicación entre ellas tiene unas determinadas

características de alcance, velocidad y seguridad según el rango de frecuencia, el tipo de antenas utilizadas, el tipo de etiquetas y demás parámetros que se pueden configurar para una aplicación u otra.

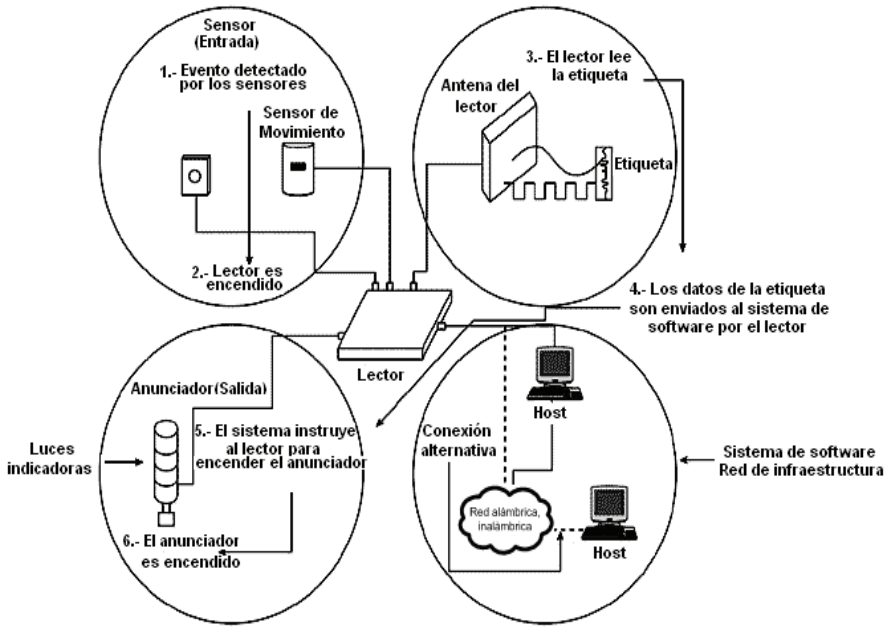


Figura 1.1: Modo de Funcionamiento de la Tecnología RFID

1.1.6 COMPONENTES BÁSICOS DE LA TECNOLOGÍA RFID

El sistema RFID está formado de los siguientes componentes como se podrán ver en la figura 1.2[2]:

1.1.6.1 Etiqueta

Éste es un componente obligatorio de cualquier sistema RFID.

1.1.6.2 Lector

Este es también un componente obligatorio.

1.1.6.3 Antena del Lector

Éste es otro componente obligatorio. Actualmente algunos lectores tienen las antenas incorporadas.

---

[2] http://www.libera.net/uploads/documents/whitepaper_rfid.pdf

1.1.6.4 Sensor, actor y anunciador

Estos componentes opcionales se necesitan para la entrada y salida externa del sistema.

1.1.6.5 Host y Sistema de Software

Teóricamente, un sistema RFID puede funcionar independientemente sin este componente. Prácticamente, un sistema RFID pierde su valor sin este componente.

1.1.6.6 Infraestructura de Comunicación

Componentes obligatorios, es la asociación de una red alámbica e inalámbrica y una infraestructura de comunicación serial la cual se usa para conectar los componentes previamente listados para tener una comunicación efectiva entre sí.

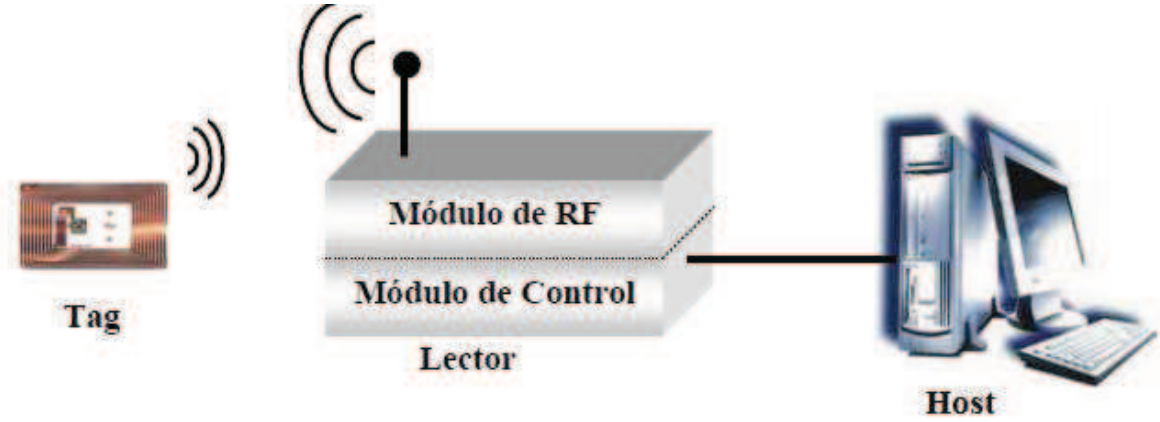


Figura 1.2: Componentes de un Sistema RFID

1.1.7 ACOPLAMIENTO DE UN SISTEMA RFID

1.1.7.1 Acoplamiento Inductivo

El acoplamiento inductivo se basa en el mismo funcionamiento de los transformadores. En la figura 1.3 podemos observar un esquema del acoplamiento inductivo. En estas frecuencias el campo creado por la antena del interrogador es la energía que aprovecha el transponder para su comunicación. Este campo está cerca de la antena del interrogador, lo que permite alcanzar unas distancias cercanas al diámetro de la antena. A distancias mayores la

potencia necesaria es muy elevada. La bobina del lector genera un fuerte campo electromagnético, que penetra en la sección de la antena del transponder y en su zona cercana.

Las antenas de estos sistemas son bobinas, tanto del lector como del transponder, de gran tamaño.

La eficiencia de la energía transmitida entre las antenas del lector y del transponder es proporcional a la frecuencia de operación.

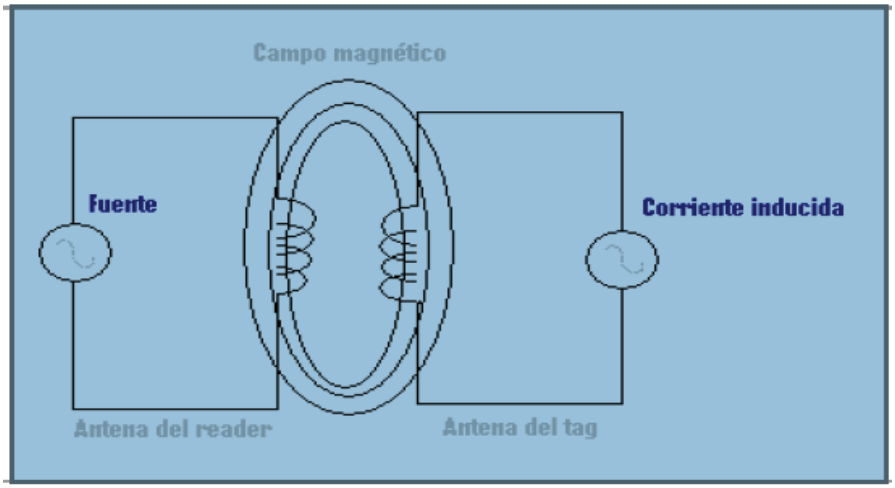


Figura 1.3: Esquema del acoplamiento inductivo entre lector y transponder.

1.1.8 BANDAS DE FRECUENCIAS DE UN SISTEMA RFID [3]

Los sistemas RFID se clasifican dependiendo del rango de frecuencias que usan. Existen cuatro tipos de sistemas detallados a continuación:

1.1.8.1 Frecuencia Baja (125 KHz - 134 KHz)

Su principal ventaja es su aceptación en todo el mundo, funciona cerca de los metales y está ampliamente difundida. La distancia de lectura es inferior a 1,5 metros, por lo que las aplicaciones más habituales son la identificación de animales, barriles de cerveza, auto key and lock o bibliotecas.

---

[3] http://www.novenca.com/site/index.php?option=com_content&view=article&id=166&Itemid=123

1.1.8.2 Frecuencia Alta (13,56 MHz)

Esta frecuencia también está muy difundida, pero a diferencia de la frecuencia baja, la alta no funciona cerca de los metales. Normalmente se utiliza en aplicaciones tales como la trazabilidad de los productos, movimientos de equipajes de avión o acceso a edificios.

1.1.8.3 Frecuencia Ultra-Alta (860MHz-960 MHz)

Los equipos que operan a estas frecuencias UHF (*Ultra High Frequency*) no pueden ser utilizados de forma global porque no existen regulaciones globales para su uso y su aplicación depende de la legalidad del país. Este tipo de frecuencia se usa para aplicaciones de trazabilidad con tags activos.

1.1.8.4 Frecuencia de Microondas (2,45 GHz y 5,8 GHz)

Estas frecuencias son las más habituales para los tags activos, pero no tienen el problema de la falta de regulaciones globales, además ofrecen largas distancias de lectura y altas velocidades de transmisión. Los tags activos que operan en el rango de las microondas son muy usados para seguimiento y trazabilidad de personas u objetos.

1.1.9 REGULACIÓN Y ESTANDARIZACIÓN [4]

1.1.9.1 Regulación

No hay ninguna corporación pública global que gobierne las frecuencias usadas para RFID. En principio, cada país puede fijar sus propias reglas.

Las principales corporaciones que gobiernan la asignación de las frecuencias para RFID son:

- EE.UU.: FCC (Federal Communications Commission)
- Canadá: DOC (Departamento de la Comunicación)

---

[4] http://www.subtel.gob.cl/prontus_subtel/site/edic/base/port/inicio.html

- Europa: ERO, CEPT, ETSI y administraciones nacionales. Obsérvese que las administraciones nacionales tienen que ratificar el uso de una frecuencia específica antes de que pueda ser utilizada en ese país.
- Japón: MPHPT (Ministry of Public Management, Home Affairs, Post and Telecommunication).
- China: Ministerio de la Industria de Información.
- Australia: Autoridad Australiana de la Comunicación (Australian Communication Authority).
- Nueva Zelanda: Ministerio de desarrollo económico de Nueva Zelanda (New Zealand Ministry of Economic Development).
- Argentina: CNC (Comisión Nacional de Comunicaciones).
- Chile: SUBTEL
- Las etiquetas RFID de baja frecuencia (LF: 125KHz - 134 KHz y 140KHz - 148.5 kHz) y de alta frecuencia (HF: 13.56 MHz) se pueden utilizar de forma global sin necesidad de licencia. La frecuencia ultraalta (UHF: 868MHz - 928 MHz) no puede ser utilizada de forma global, ya que no hay un único estándar global. En Norteamérica, la frecuencia ultraelevada se puede utilizar sin licencia para frecuencias (908MHz - 928 MHz), pero hay restricciones en la energía de transmisión. En Europa la frecuencia ultraelevada está bajo consideración para 865.6MHz - 867.6 MHz. Su uso es sin licencia sólo para el rango de 869.40MHz - 869.65 MHz, pero existen restricciones en la energía de transmisión. El estándar UHF norteamericano (908MHz-928 MHz) no es aceptado en Francia e Italia ya que interfiere con sus bandas militares. En China y Japón no hay regulación para el uso de la frecuencia ultraelevada. Cada aplicación de frecuencia ultraelevada en estos países necesita de una licencia, que debe ser solicitada a las autoridades locales, y puede ser revocada. En Australia y Nueva Zelanda, el rango es de 918MHz - 926 MHz para uso sin licencia, pero hay restricciones en la energía de transmisión.

Existen regulaciones adicionales relacionadas con la salud y condiciones ambientales. Por ejemplo, en Europa, la regulación Waste Electrical and Electronic Equipment (“Equipos eléctricos y electrónicos inútiles”), no permite que

se desechen las etiquetas RFID. Esto significa que las etiquetas RFID que estén en cajas de cartón deben ser quitadas antes de deshacerse de ellas.

1.1.9.2 Estandarización

Los estándares de RFID abordan cuatro áreas fundamentales:

- **Protocolo en la interfaz aérea:** especifica el modo en el que etiquetas RFID y lectores se comunican mediante radiofrecuencia.
- **Contenido de los datos:** especifica el formato y semántica de los datos que se comunican entre etiquetas y lectores.
- **Certificación:** pruebas que los productos deben cumplir para garantizar que cumplen los estándares y pueden interoperar con otros dispositivos de distintos fabricantes.
- **Aplicaciones:** usos de los sistemas RFID.

Como en otras áreas tecnológicas, la estandarización en el campo de RFID se caracteriza por la existencia de varios grupos de especificaciones competidoras. Por una parte está ISO, y por otra Auto-ID Centre (conocida desde octubre de 2003 como EPCglobal, de EPC, *Electronic Product Code*). Ambas comparten el objetivo de conseguir etiquetas de bajo coste que operen en UHF.

Por su parte, ISO ha desarrollado estándares de RFID para la identificación automática y la gestión de objetos. Existen varios estándares relacionados, como ISO 10536**,** ISO 14443 **e** ISO 15693.

Pero la serie de estándares estrictamente relacionada con las RFID y las frecuencias empleadas en dichos sistemas es la serie ***18000*** como se ve en la tabla 1.1.

Tabla 1.1: Estándar ISO – 18000 para la definición de la interfaz aire RFID

Estándar ISO - Serie 18000 para la definición del interfaz aire RFID	
18000-1	Parámetros genéricos para la interfaz aire en todas las frecuencias.
18000-2	Parámetros genéricos para la interfaz aire para comunicaciones por debajo de 135KHz.
18000-3	Parámetros genéricos para la interfaz aire para comunicaciones a 13.56MHz.
18000-4	Parámetros genéricos para la interfaz aire para comunicaciones a 2.45GHz.
18000-5	Parámetros genéricos para la interfaz aire para comunicaciones a 5.8GHz.
18000-6	Parámetros genéricos para la interfaz aire para comunicaciones desde 860 a 960MHz.
18000-7	Parámetros genéricos para la interfaz aire para comunicaciones a 433MKHz.

- Los estándares **EPC** para etiquetas son de dos clases:

Clase 1: etiqueta simple, pasiva, de sólo lectura con una memoria no volátil programable una sola vez.

Clase 2: etiqueta de sólo lectura que se programa en el momento de fabricación del chip (no reprogramable posteriormente).

EPC Radio-Frequency Identify Protocols zClass-1 Generation-2 UHF RFID: creado por EPC Global, entre EAN (European Article Numbering) y UCC (Uniform Code Council), y tecnología desarrollada por Auto – ID Center, en este documento se desarrolla el estándar para el protocolo de interfaz aérea de comunicación entre el tag y el lector.

1.1.10 APLICACIÓN DE LA TECNOLOGÍA RFID

La RFID puede utilizarse para cubrir diferentes necesidades. Es ideal para sectores internos de una compañía y áreas logísticas.

Existen aplicaciones que utilizan esta tecnología para:

- Identificar animales y mascotas, mediante la inserción de un pequeño transponder en su interior.
- Control de flotillas de vehículos
- Control de llantas

- Autenticidad de productos
- Seguimiento de productos
- Control de acceso
- Inmovilizador de Vehículos
- Peaje
- Manejo de envíos
- Pasaportes
- Carnet de conducir
- Seguimiento de cilindros de gas (Air Liquide, AGA)
- Seguimiento de paquetes (Wall Mart)
- Seguimiento de vestimenta industrial alquilada (Elis)
- Administración de los libros de una biblioteca
- Identificación de camiones y vagonetas (SNFF).
- Contenedores de automóviles
- Contenedores metálicos de gran tamaño
- Jaulas rodantes para envíos postales
- Tambores metálicos
- Seguimiento de barricas de cerveza

1.2 LECTOR RFID

Un lector RFID, también conocido como un integrador, es un dispositivo que puede leer y escribir datos en las etiquetas RFID que sean compatibles. El tiempo durante el cual el lector puede emitir energía de RF para leer las etiquetas se denomina ciclo de lectura del lector.

El lector emite ondas electromagnéticas a la antena de la etiqueta pasiva, la energía de esas ondas activa el circuito integrado de la etiqueta que contiene la información que se desea leer y ésta es enviada de regreso al lector a través de la antena de la etiqueta.

Básicamente tienen la habilidad de localizar, identificar o rastrear objetos. Los lectores no tienen las restricciones de “línea de lectura” que tienen los lectores de

código de barras. Pueden leer simultáneamente varias etiquetas (hasta 200 por segundo) o focalizar la lectura en una sola etiqueta en particular.

La máxima distancia a la que puede establecerse la comunicación entre el lector y la etiqueta depende de la potencia del lector y de la frecuencia que se utiliza para la comunicación entre el lector y la etiqueta.

Figura 1.4: Ejemplo de un lector RFID ID12[5]

1.2.1 COMPONENTES DEL LECTOR RFID

Un lector tiene principalmente los siguientes elementos que se podrán ver en la figura 1.5:

- Transmisor
- Receptor
- Microprocesador
- Memoria
- Canales de Input/Output para sensores externos, actuadores y anunciadores (aunque, hablando estrictamente, éstos son componentes optativos, que se proporciona casi siempre en un lector comercial).
- Controladora (La cual puede residir como un componente externo)
- Interfaz de comunicación
- Fuente de poder

[5] http://www.electronicamagnabit.com/tienda/kits-rf/155-lector-rfid-id-12.html

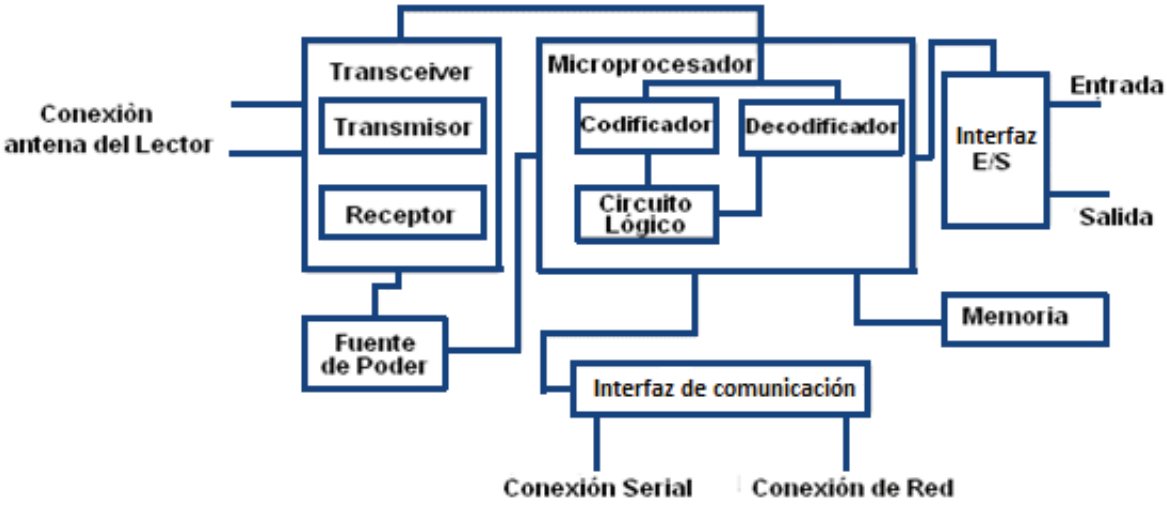


Figura 1.5: Componentes de un lector RFID

1.3 TARJETA RFID O TAG

Una etiqueta RFID es un dispositivo que puede almacenar y transmitir datos a un lector por medio de ondas de radio.

Figura 1.6: Ejemplo de un Tag Pasivo 125Khz[6]

1.3.1 CLASIFICACIÓN DE LAS ETIQUETAS RFID

Las etiquetas RFID se las clasifica[7] de la siguiente manera:

- Pasivas
- Activas
- Semi-activas (También conocidas como semi-pasivas)

---

[6] http://www.electronicamagnabit.com/tienda/kits-rf/155-lector-rfid-id-12.html

[7] http://es.kioskea.net/contents/rfid/rfid-intro.php3

1.3.1.1 Etiquetas Pasivas

Los tags pasivos no poseen alimentación eléctrica. La señal que les llega de los lectores induce una corriente eléctrica pequeña y suficiente para operar el circuito integrado CMOS del tag, de forma que puede generar y transmitir una respuesta, esta respuesta puede ser cualquier tipo de información, no sólo un código identificador. Un tag puede incluir memoria no volátil, posiblemente escribible (por ejemplo EEPROM).

1.3.1.2 Etiquetas Activas

Las etiquetas activas tienen incluida una fuente de poder por ejemplo una batería. Una etiqueta activa usa su fuente de poder para transmitir sus datos al lector y no necesita que el lector emita una señal de poder para la transmisión de datos.

1.3.1.3 Etiquetas Semi-Activas (semi-pasivas)

Las etiquetas Semi-activas tienen una fuente de poder y la electrónica respectiva para realizar tareas especializadas. La fuente de alimentación provee de energía a la etiqueta para su funcionamiento. Sin embargo, para transmitir sus datos, una etiqueta semi-activa usa la energía transmitida por el lector. En la comunicación del lector con la etiqueta, el lector siempre se comunica primero, seguido de la etiqueta.

1.3.2 TIPOS DE ETIQUETAS [8]

1.3.2.1 Etiquetas de Baja Frecuencia LF

Los tags de baja frecuencia LF (del inglés low frequency) normalmente se sirven de la inducción electromagnética.

Las etiquetas RFID de baja frecuencia (LF) se utilizan comúnmente para la identificación de animales, seguimiento de barricas de cerveza, y como llave de automóviles con sistema antirrobo.

---

[8] http://es.wikipedia.org/wiki/RFID

1.3.2.2 Etiquetas de Alta Frecuencia HF

En alta frecuencia (HF 13,56 MHz) se utiliza una espiral plana con 5-7 vueltas y un factor de forma parecido al de una tarjeta de crédito para lograr distancias de decenas de centímetros.

Las etiquetas RFID de alta frecuencia (HF) se utilizan en bibliotecas y seguimiento de libros, control de acceso en edificios, seguimiento de equipaje en aerolíneas, seguimiento de artículos de ropa.

1.3.2.3 Etiquetas de Ultra Alta Frecuencia UHF

Los tags pasivos en frecuencias ultraalta (UHF) y de microondas suelen acoplarse por radio a la antena del lector y utilizar antenas clásicas de dipolo.

Las etiquetas RFID de UHF se utilizan comúnmente de forma comercial en seguimiento de envases, y seguimiento de camiones y remolques en envíos.

1.3.3 ELEMENTOS DE UN TAG RFID

La etiqueta de radiofrecuencia (traspondedor, etiqueta RFID) está formada por un chip conectado a una antena, ambos contenidos en un rótulo (etiqueta RFID o rótulo RFID). Un dispositivo lo lee, captura y trasmite la información. En el caso de los tags activos y semi-activos se adiciona una batería para su alimentación. En la Figura 1.7 se pueden apreciar los principales componentes de un tag.

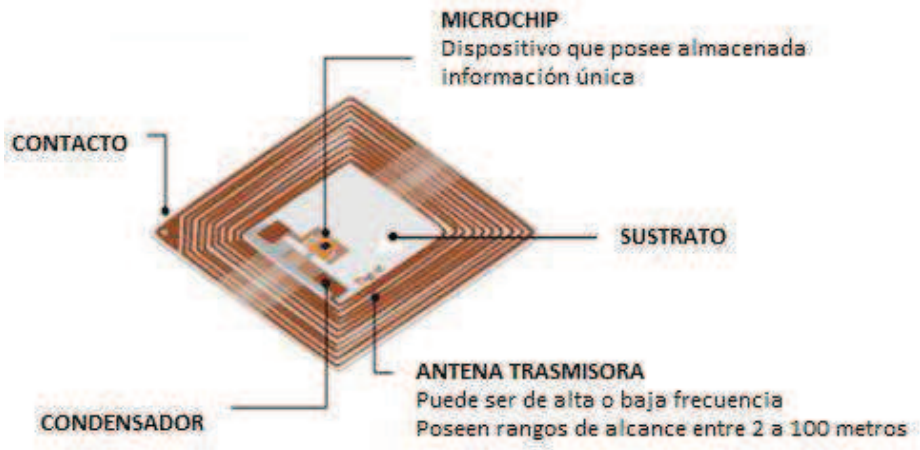


Figura 1.7: Estructura Interna de un Tag

1.3.3.1 Chip o Circuito Integrado

El microchip almacena un número de identificación, una especie de matrícula única de dicho producto. Hay varios tipos de esquemas propuestos para estos números, como por ejemplo el Electronic Product Code (EPC), diseñado por Auto-ID Center. Podemos decir, que cada objeto tendrá un código único que lo diferenciará e identificará no sólo de otros tipos de productos, sino de productos iguales.

1.3.3.2 Antena

Son los dispositivos que permiten radiar las señales emitidas de los lectores y leer las respuestas de los *tag*s.

Las antenas están presentes tanto en los lectores como en los tags, y de su frecuencia de operación y potencia de transmisión depende en gran parte el rango de cobertura en el que se puede dar la comunicación.

1.3.3.3 Sustrato

Simplemente es el material que mantiene el chip y la antena juntos y protegidos. Por lo general es un film plástico. Tanto el chip como la antena quedan adjuntados a él.

1.3.3.4 Condensador

Este dispositivo permite concentrar o almacenar momentáneamente la energía recibida durante el proceso de excitación del lector, es esencial en los tags pasivos.

1.3.3.5 Contacto

Esta parte del tag es utilizado para poder sujetarlo a algún dispositivo al cual se esté asociando dicha tarjeta.

1.4 MICROCONTROLADOR

1.4.1 DEFINICIÓN[9]

Es un circuito integrado programable, capaz de ejecutar las órdenes grabadas en su memoria. Está compuesto de varios bloques funcionales, los cuales cumplen una tarea específica. En la figura 1.8 se pueden apreciar algunas de sus partes.

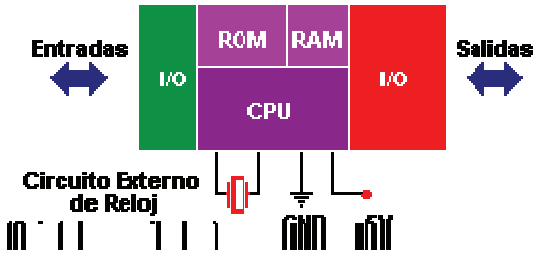


Figura 1.8: Bloques funcionales de un microprocesador.

1.4.2 HISTORIA

Inicialmente, cuando no existían los microprocesadores, las personas se ingeniaban en diseñar sus circuitos electrónicos y los resultados estaban expresados en diseños que implicaban muchos componentes electrónicos y cálculos matemáticos. Un circuito lógico básico requería de muchos elementos electrónicos basados en transistores, resistencias, etc., Lo cual desembocaba en circuitos con muchos ajustes y fallos; pero en el año 1971 apareció el primer microprocesador el cual originó un cambio decisivo en las técnicas de diseño de la mayoría de los equipos. Al principio se creía que el manejo de un microprocesador era para personas con un coeficiente intelectual muy alto; por lo contrario con la aparición de este circuito integrado todo sería mucho más fácil de entender y los diseños electrónicos serían mucho más pequeños y simplificados.

Entre los microprocesadores más conocidos tenemos el popular AVR que fue concebida por dos estudiantes en Instituto de la tecnología noruego (NTH) Alf-Egil

---

[9] http://r-luis.xbot.es/pic1/pic01.html

Bogen y Vegard Wollan. Los AVR son una familia de microcontroladores RISC de Atmel siendo el primer AVR de la línea el AT90S8515.

1.4.3 TIPOS DE ARQUITECTURAS DE MICROCONTROLADORES

1.4.3.1 Arquitectura Von Neumann

Aunque inicialmente todos los microcontroladores adoptaron la arquitectura clásica de Von Neumann, en el momento presente se impone la arquitectura Harvard. La arquitectura de von Neumann se caracteriza por disponer de una sola memoria principal donde se almacenan datos e instrucciones de forma indistinta. A dicha memoria se accede a través de un sistema de buses único (direcciones, datos y control) como se ve la figura 1.9.

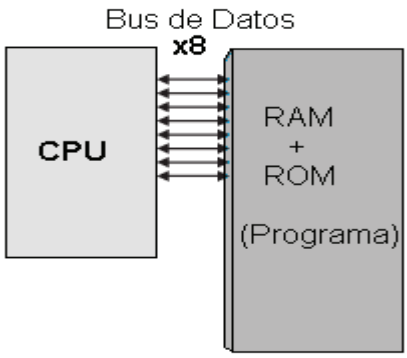


Figura 1.9: Arquitectura Von Neumann[10]

1.4.3.2 Arquitectura Harvard

La arquitectura Harvard[11] dispone de dos memorias independientes como se puede ver en la figura 1.10, que contiene sólo instrucciones y otra, sólo datos. Ambas disponen de sus respectivos sistemas de buses de acceso y es posible realizar operaciones de acceso (lectura o escritura) simultáneamente en ambas memorias.

---

[10] http://www.mikroe.com/eng/chapters/view

[11] http://perso.wanadoo.es/pictob/micropic.htm

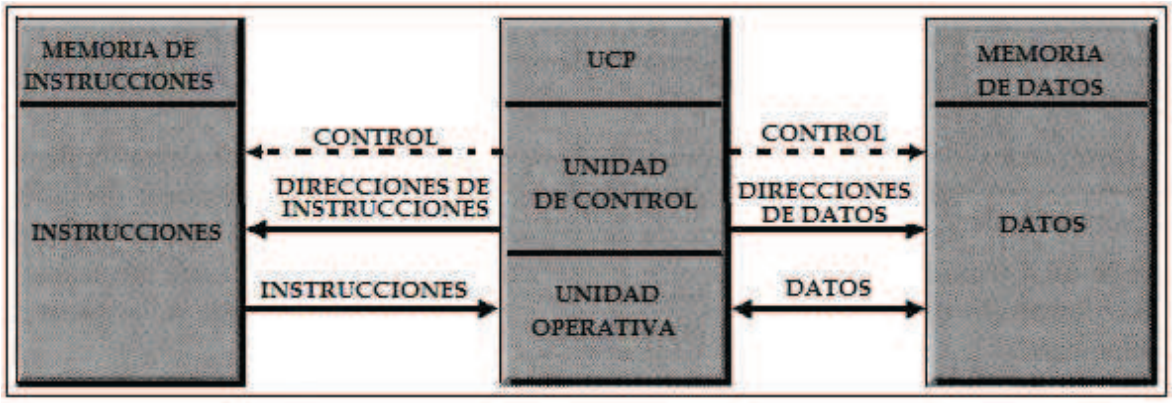


Figura 1.10: Arquitectura Harvard

Los microcontroladores PIC responden a la arquitectura Harvard.

1.4.4 ESTRUCTURA DE LOS MICROCONTROLADORES

1.4.4.1 El Procesador

Es el elemento más importante del microcontrolador y determina sus principales características, tanto a nivel hardware como software.

Se encarga de direccionar la memoria de instrucciones, recibir el código OP de la instrucción en curso, su decodificación y la ejecución de la operación que implica la instrucción, así como la búsqueda de los operandos y el almacenamiento del resultado.

Existen tres orientaciones en cuanto a la arquitectura y funcionalidad de los procesadores actuales:

1.4.4.1.1 CISC

Un gran número de procesadores usados en los microcontroladores están basados en la filosofía CISC (Computadores de Juego de Instrucciones Complejo).

Disponen de más de 80 instrucciones máquina en su repertorio, algunas de las cuales son muy sofisticadas y potentes, requiriendo muchos ciclos para su ejecución.

Una ventaja de los procesadores CISC es que ofrecen al programador instrucciones complejas que actúan como macros.

1.4.4.1.2 RISC

Tanto la industria de los computadores comerciales como la de los microcontroladores están decantándose hacia la filosofía RISC (Computadores de Juego de Instrucciones Reducido). En estos procesadores el repertorio de instrucciones máquina es muy reducido y las instrucciones son simples y, generalmente, se ejecutan en un ciclo.

La sencillez y rapidez de las instrucciones permiten optimizar el hardware y el software del procesador.

1.4.4.1.3 SISC

En los microcontroladores destinados a aplicaciones muy concretas, el juego de instrucciones, además de ser reducido, es “específico”, o sea, las instrucciones se adaptan a las necesidades de la aplicación prevista. Esta filosofía se ha bautizado con el nombre de SISC (Computadores de Juego de Instrucciones Específico).

1.4.4.2 Unidades de Memoria

En los microcontroladores la memoria de instrucciones y datos está integrada en el propio chip. Una parte debe ser no volátil, tipo ROM, y se destina a contener el programa de instrucciones que gobierna la aplicación. Otra parte de memoria será tipo RAM, volátil, y se destina a guardar las variables y los datos.

1.4.4.3 Memoria ROM (Read Only Memory)

Es una memoria no volátil de sólo lectura cuyo contenido se graba durante la fabricación del chip. El elevado costo del diseño de la máscara sólo hace aconsejable el empleo de los microcontroladores con este tipo de memoria cuando se precisan cantidades superiores a varios miles de unidades.

1.4.4.4 Memoria Flash

Se trata de una memoria no volátil, de bajo consumo, que se puede escribir y borrar, es programable en el circuito es decir, sin tener que sacar el circuito integrado de la tarjeta. Funciona como una ROM y una RAM, es más veloz y tolera más ciclos de escritura/borrado.

1.4.4.5 Memoria de Datos

Es una memoria RAM (Random Access Memory) que típicamente puede ser de 1, 2, 4, 8, 16, 32 kilobytes.

Muchos microcontroladores han incorporado este tipo de memoria como un periférico más, para el almacenamiento de datos de configuración o de los procesos que se controlan. Esta memoria es independiente de la memoria de datos tipo ROM o la memoria de programas, en la que se almacena el código del programa a ejecutar por el procesador del microcontrolador.

1.4.5 PUERTOS DE ENTRADA Y SALIDA[12]

La principal utilidad de los pines que posee la cápsula que contiene un microcontrolador es soportar las líneas de E/S que comunican al computador interno con los periféricos exteriores.

Los puertos de E/S, generalmente agrupadas en puertos de 8 bits de longitud, permiten leer datos del exterior o escribir en ellos desde el interior del microcontrolador, el destino habitual es el trabajo con dispositivos simples como relés, LED, o cualquier otra cosa que se le ocurra al programador.

Algunos puertos de E/S tienen características especiales que le permiten manejar salidas con determinados requerimientos de corriente, o incorporan mecanismos especiales de interrupción para el procesador.

---

[12] Ver Anexo 1

Típicamente cualquier pin de E/S puede ser considerada E/S de propósito general, pero como los microcontroladores no pueden tener infinitos pines, ni siquiera todos los pines que queramos, las E/S de propósito general comparten los pines con otros periféricos. Para usar un pin con cualquiera de las características a él asignadas debemos configurarlo mediante los registros destinados a ellos.

Según los controladores de periféricos que posea cada modelo de microcontrolador, las líneas de E/S se destinan a proporcionar el soporte a las señales de entrada, salida y control.

1.4.6 RELOJ PRINCIPAL

Todos los microcontroladores disponen de un circuito oscilador que genera una onda cuadrada de alta frecuencia, que configura los impulsos de reloj usados en la sincronización de todas las operaciones del sistema, generalmente, el circuito de reloj está incorporado en el microcontrolador y sólo se necesitan unos pocos componentes exteriores para seleccionar y estabilizar la frecuencia de trabajo. Dichos componentes suelen consistir en un cristal de cuarzo que genera una señal oscilatoria de entre 1 a 40 MHz.

1.4.7 RECURSOS ESPECIALES[13]

Cada fabricante oferta numerosas versiones de una arquitectura básica de microcontrolador.

1.4.7.1 Temporizadores o “Timers”.

Se emplean para controlar periodos de tiempo (temporizadores) y para llevar la cuenta de acontecimientos que suceden en el exterior (contadores), para la medida de tiempos se carga un registro con el valor adecuado y a continuación dicho valor se va incrementando o decrementando al ritmo de los impulsos de

---

[13] http://perso.wanadoo.es/pictob/microcr.htm

reloj o algún múltiplo hasta que se desborde y llegue a 0, momento en el que se produce un aviso.

1.4.7.2 Perro guardián o "Watchdog".

El Perro guardián consiste en un temporizador que, cuando se desborda y pasa por 0, provoca un reset automáticamente en el sistema.

Se debe diseñar el programa de trabajo que controla la tarea de forma que refresque o inicialice al Perro guardián antes de que provoque el reset. Si falla el programa o se bloquea, no se refrescará al Perro guardián y, al completar su temporización, "ladrará y ladrará" hasta provocar el reset.

1.4.7.3 Protección ante fallo de alimentación o "Brownout".

Se trata de un circuito que resetea al microcontrolador cuando el voltaje de alimentación (VDD) es inferior a un voltaje mínimo ("brownout"). Mientras el voltaje de alimentación sea inferior al de brownout el dispositivo se mantiene reseteado, comenzando a funcionar normalmente cuando sobrepasa dicho valor.

1.4.7.4 Estado de reposo o de bajo consumo.

Para ahorrar energía, (factor clave en los aparatos portátiles), los microcontroladores disponen de una instrucción especial (SLEEP en los PIC), que les pasa al estado de reposo o de bajo consumo, en el cual los requerimientos de potencia son mínimos. En dicho estado se detiene el reloj principal y se "congelan" sus circuitos asociados, quedando sumido en un profundo "sueño" el microcontrolador. Al activarse una interrupción ocasionada por el acontecimiento esperado, el microcontrolador se despierta y reanuda su trabajo.

1.4.7.5 Conversor A/D.

Los microcontroladores que incorporan un Conversor A/D (Analógico/Digital) pueden procesar señales analógicas, tan abundantes en las aplicaciones. Suelen disponer de un multiplexor que permite aplicar a la entrada del CAD diversas señales analógicas desde las patitas del circuito integrado.

1.4.7.6 Conversor D/A.

Transforma los datos digitales obtenidos del procesamiento del computador en su correspondiente señal analógica que saca al exterior por una de las patitas de la cápsula.

1.4.7.7 Comparador Analógico.

Algunos modelos de microcontroladores disponen internamente de un Amplificador Operacional que actúa como comparador entre una señal fija de referencia y otra variable que se aplica por uno de los pines de la cápsula. La salida del comparador proporciona un nivel lógico 1 ó 0 según una señal sea mayor o menor que la otra.

1.4.7.8 Modulador de Anchura de Impulsos o PWM.

Son circuitos que proporcionan en su salida impulsos de anchura variable. Los PWM (Pulse Width Modulator) son periféricos muy útiles sobre todo para el control de motores, inversión DC/AC para UPS, conversión digital analógica D/A, control regulado de luz (dimming) entre otras.

1.4.7.9 Puertas de E/S digitales.

Todos los microcontroladores destinan algunas de sus pines a soportar líneas de E/S digitales. Por lo general, estas líneas se agrupan de ocho en ocho formando Puertas, que pueden configurarse como Entrada o como Salida cargando un 1 ó un 0 en el bit correspondiente de un registro destinado a su configuración.

1.4.7.10 Puertas de comunicación

Las puertas de comunicación permiten dotar al microcontrolador de la posibilidad de comunicarse con otros dispositivos externos, otros buses de microprocesadores, buses de sistemas o buses de redes y poder adaptarlos con otros elementos bajo otras normas y protocolos.

Algunos modelos de microcontrolaodres disponen de recursos que permiten directamente esta tarea, entre los que destacan:

1.4.7.10.1 UART

Adaptador de comunicación serie asíncrona.

1.4.7.10.2 USART

Adaptador de comunicación serie síncrona y asíncrona.

1.4.7.10.3 Puerta paralela esclava

Para poder conectarse con los buses de otros microprocesadores.

1.4.7.10.4 USB (Universal Serial Bus)

Que es un moderno bus serie para los PC.

1.4.7.10.5 Bus I2C

Que es un interfaz serie de dos hilos desarrollado por Philips.

1.4.7.10.6 CAN (Controller Area Network)

Para permitir la adaptación con redes de conexionado multiplexado desarrollado conjuntamente por Bosch e Intel para el cableado de dispositivos en automóviles.

1.5 RELOJ DE TIEMPO REAL DS1307[14]

El semiconductor Maxim/Dallas DS1307 es un reloj / calendario de tiempo real exacto de baja potencia, el cual automáticamente, mantiene el tiempo y la fecha actual, provee información de segundos, minutos, horas, día, mes y año. La fecha al final del mes durante los meses con menos de 31 días, se ajusta automáticamente e incluye las correcciones para el año bisiesto. El reloj funciona en cualquiera de los formatos de hora 24 o 12 horas con indicador AM/PM, su

[14] http://picaxe.electronicasimple.com/2009/03/reloj-tiempo-real-ds1307.html

funcionamiento se basa en un oscilador externo de cristal de cuarzo de alta precisión, con una frecuencia de 32,768 KHz para proveer tiempo base exacto.

El DS1307 tiene un circuito integrado en el sensor de energía que detecta los fallos de alimentación y cambia automáticamente a la fuente de respaldo que es una pila de litio como se puede ver en la figura 1.11.

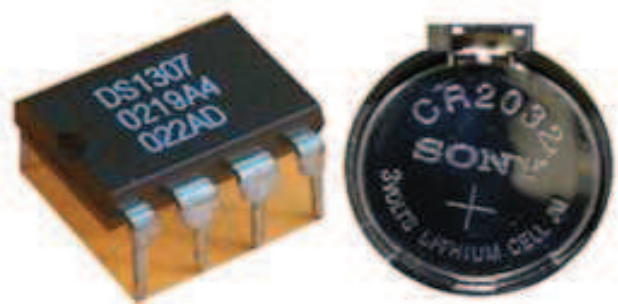


Figura 1.11: Integrado DS1307

1.5.1 DESCRIPCIÓN DE PINES

La figura 1.12, muestra la asignación de pines del DS1307

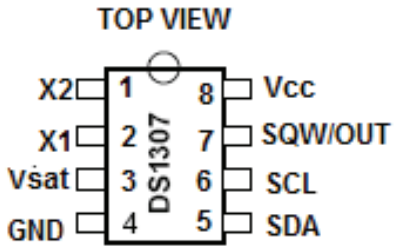


Figura 1.12: Distribución de Pines del Integrado DS1307

La figura 1.13 indica la forma de conexiones del integrado DS1307 y la respectiva descripción de sus pines son presentados en la tabla 1.2.

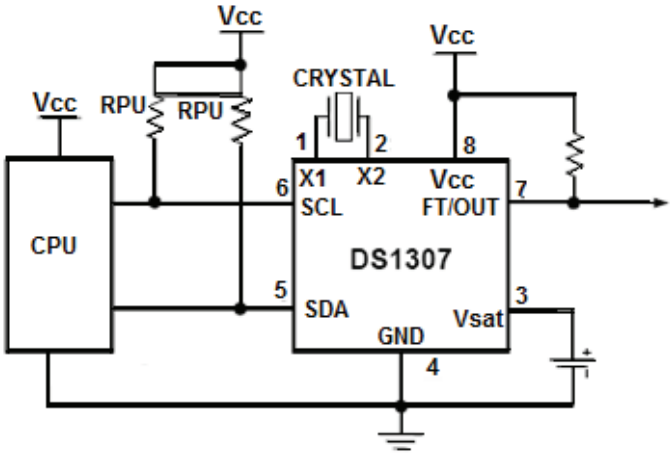


Figura 1.13: Diagrama de Conexiones del Integrado Ds1307

Tabla 1.2: Descripción de Pines del Integrado DS1307

PIN	DESCRIPCIÓN
VCC (pin8) GND (pin4)	La alimentación DC es proporcionada a este circuito a través de estos pines. VCC es la entrada de +5VDC, mientras que GND es la referencia, el voltaje mínimo al que trabaja es 4.5v y el máximo 5.5v.
VBAT (pin3)	Entrada de alimentación de una pila estándar de litio de 3 Voltios. El voltaje debe estar entre 2.5 y 3.5 voltios para una operación apropiada.
SCL (pin 6)	Entrada de reloj para sincronizar la transferencia de datos en la interfaz serial.
SDA (pin 5)	Entrada/salida de datos para la interfaz I2C. Este pin es de drenaje abierto, por lo que requiere de una resistencia pull-up externa.
X1 (pin1) X2 (pin2)	Conexiones para un cristal de cuarzo estándar de 32.768 Hz.
SQW/OUT (pin7)	Es una salida de colector abierto, que puede ser programada para hacer "flash" cada 1Hz. Esto permite la colocación de un led como indicador de segundos en aplicaciones de reloj.

Todos los datos de tiempo/fecha están en formato BCD, lo cual hace muy fácil su lectura y escritura usando notación hexadecimal. Por ejemplo 11:35 a.m. va a contener $11 en el registro de horas y $35 en el registro de minutos. Tenga en cuenta que el chip no va a operar hasta que sea puesto el tiempo y fecha actual.

La dirección de escritura del integrado es 1101000x, donde x es el bit que específica la acción a realizar: 0 Escritura, 1 Lectura. Las direcciones para

guardar los datos de hora y fecha se observan en la siguiente figura 1.14[15] con sus respectivos registros.

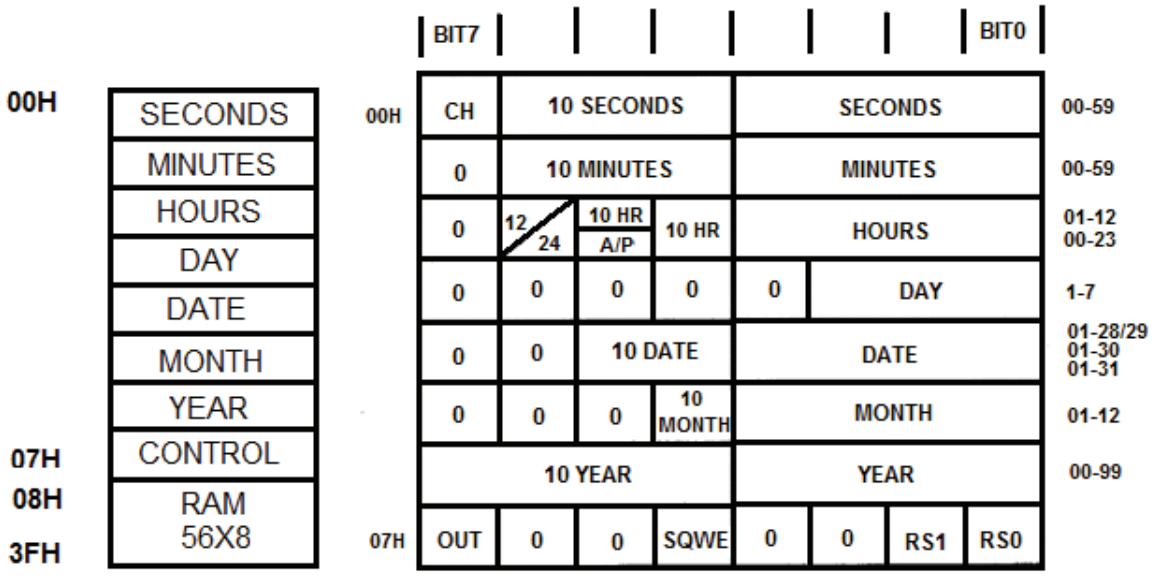


Figura 1.14: Direcciones de Horario

El registro de control presentado en la tabla 1.3, es usado para el control de operación del pin SQW/OUT (pin 7).

Tabla 1.3: Registro de Control.

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
OUT	0	0	SQWE	0	0	RS1	RS0

- **OUT (Output Control):** Este bit controla el nivel de salida del pin 7, cuando SQWE esta deshabilitado. Si SQWE es cero el nivel lógico de salida es 1.
- **SQWE (Square Wave Enable):** Cuando este bit es 1 lógico habilita la oscilación de salida. La frecuencia de oscilación depende del valor de los registros RS1 y RS0.
- **RS (Rate Select):** Estos bit controlan la frecuencia de SQWE, en la tabla 1.4 se indica las frecuencias a ser seleccionadas.

[15] Ver Anexo 2

Tabla 1.4: Rate Select

RS1	RS0	SQW OUTPUT FREQUENCY
0	0	1Hz
0	1	4.096KHz
1	0	8.192KHz
1	1	32.768KHz

1.5.2 CARACTERÍSTICAS GENERALES DEL DS1307[16]

- Datos de tiempo/fecha están en formato BCD
- Reloj de tiempo real que cuenta los segundos, los minutos, las horas, el año, mes y día.
- Formato de 12 Horas con indicador AM/PM ó de 24 horas.
- Protocolo de comunicación I2C.
- 56 bytes de RAM no volátil, para almacenamiento de datos.
- Señal de onda cuadrada programable.

1.6 MEMORIA 24LS256

La memoria 24LS256 mostrado en la figura 1.15, es un dispositivo que permite almacenar 256Kbits de información, ha sido desarrollada por el avance de las aplicaciones, tales como las comunicaciones personales o la adquisición de datos, que requieren trabajar a baja potencia.

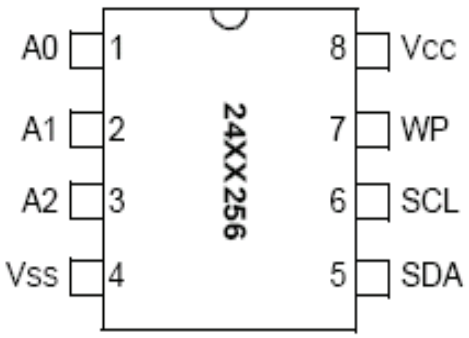


Figura 1.15: Memoria 24LS256

[16] http://electronicapractica.crearblog.com/2010/11/reloj-con-el-ds1307/#more-306

Esta memoria permite leer la información almacenada de forma secuencial o de forma aleatoria.

Las especificaciones de la memoria son las siguientes:

- Interfaz serial de comunicación de dos hilos (I^2C)
- 1000000 de ciclos de grabación y borrado
- Permite la conexión de dispositivos similares en cascada
- Tiempo de retención de la información, 200 años
- Opera en un amplio rango de voltaje (1.8 V a 5.5 V)

La figura 1.16 indica la forma de conexión entre la memoria y el microcontrolador.

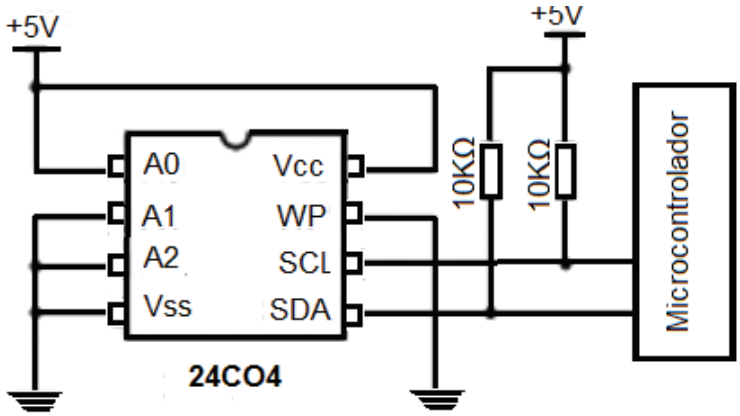


Figura 1.16: Diagrama de Conexión de la Memoria 24LS256

En la tabla 1.5 se ve la respectiva descripción de sus pines.

Tabla 1.5: Descripción de Pines de la Memoria 24LS256

DESCRIPCIÓN DE PINES	
A0,A1,A2 (pin1,2,3)	Estas entradas determinan la dirección que tendrá la memoria, son utilizadas para trabajar con varias memorias en cascada, pudiéndose conectar como máximo 8 memorias en cascada.
Vss(pin 4)	Gnd
SDA(pin 5)	Pin bi-direccional usado para transferir direcciones y datos.
SCL(pin 6)	Esta entrada es usada para sincronizar el dato transferido desde y hasta la memoria.
WP(pin 7)	(Write Project) Habilita la operación normal de la memoria, permitiendo leer y escribir en todo su rango (0000-7FFF).
Vcc(pin 8)	Vcc

La selección de la dirección de inicio en la memoria a trabajar se indica en la figura 1.17[17], la cual es controlada mediante el siguiente formato de byte.

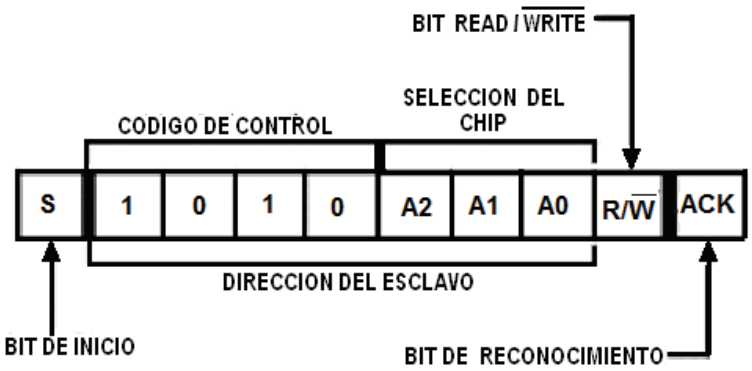


Figura 1.17: Dirección de Inicio

- El formato del byte de control consiste en iniciar enviando una condición de inicio, para luego enviar el código de control 1010 para el 24LC256.
- Los siguientes 3 bits (A2, A1, A0), seleccionan la memoria en la cual se trabajará en caso de haber más en cascada en el mismo bus
- El bit R/W (Read/Write) es seleccionado dependiendo de la aplicación a realizar, seguido de su bit de reconocimiento ACK.

Una vez seleccionada la memoria a trabajar, se elige la dirección en donde se debe leer o escribir un dato. La dirección es seleccionada utilizando dos bytes.

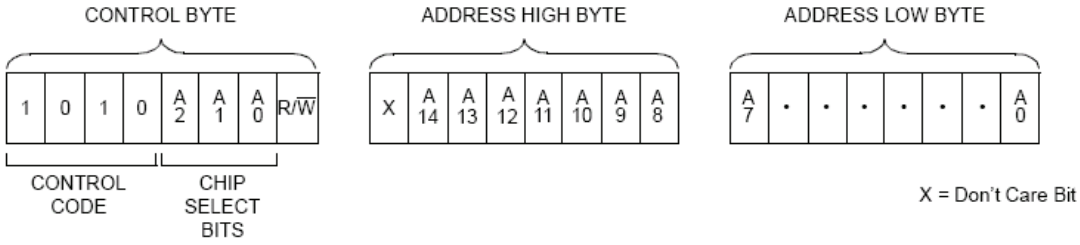


Figura 1.18: Selección de Localidad en la Memoria

La memoria tiene una capacidad de 256Kbits y para escribir en una localidad de esta su dirección puede variar entre el rango de (0000-7FFF), por lo que se

[17] Ver Anexo 3

necesita 2 bytes de direccionamiento, la manera de seleccionar una localidad es enviar primero el byte más significativo de la dirección seguido del byte menos significativo, para luego leer o escribir el dato en la localidad seleccionada.

1.7 MAX232, ADAPTADOR DE NIVELES TTL A RS-232[18]

El MAX232 es un circuito integrado que convierte los niveles de las líneas de un puerto serie RS232 a niveles TTL y viceversa, soluciona la conexión necesaria para lograr comunicación entre el puerto serie de una PC y cualquier otro circuito con funcionamiento en base a señales de nivel TTL/CMOS.

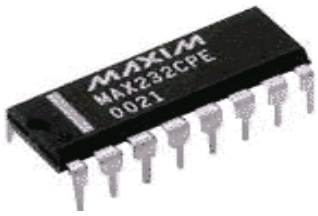


Figura 1.19: MAX232

El circuito integrado lleva internamente 2 conversores de nivel de TTL a RS232 y otros 2 de RS232 a TTL con los que podremos manejar las 4 señales más utilizadas del puerto serie del PC que son TX, RX, RTS y CTS.

- **TX** es la señal de transmisión de datos,
- **RX** es la de recepción
- **RTS y CTS** se utilizan para establecer el protocolo para el envío y recepción de los datos.

1.8 LENGUAJE DE PROGRAMACIÓN BASCOM AVR

La herramienta BASCOM AVR desarrollada por la empresa MCS Electronics, sirve para realizar programas de alto nivel para microcontroladores AVR. Ofrece una completa solución para editar, compilar, simular y programar. Posee un

---

[18] http://robots-argentina.com.ar/Comunicacion_max232.htm

compilador y un ensamblador que traduce las instrucciones estructuradas en lenguaje de máquina.[19]

Para iniciar programando en BASCOM AVR es posible descargar su versión demo, en la página principal de MCS Electronics.

Una vez instalado BASCOM dar doble click sobre el icono que permite abrir el programa. Dentro de ella podemos ver claramente la barra de herramientas, el menú y el área de trabajo (figura 1.20).

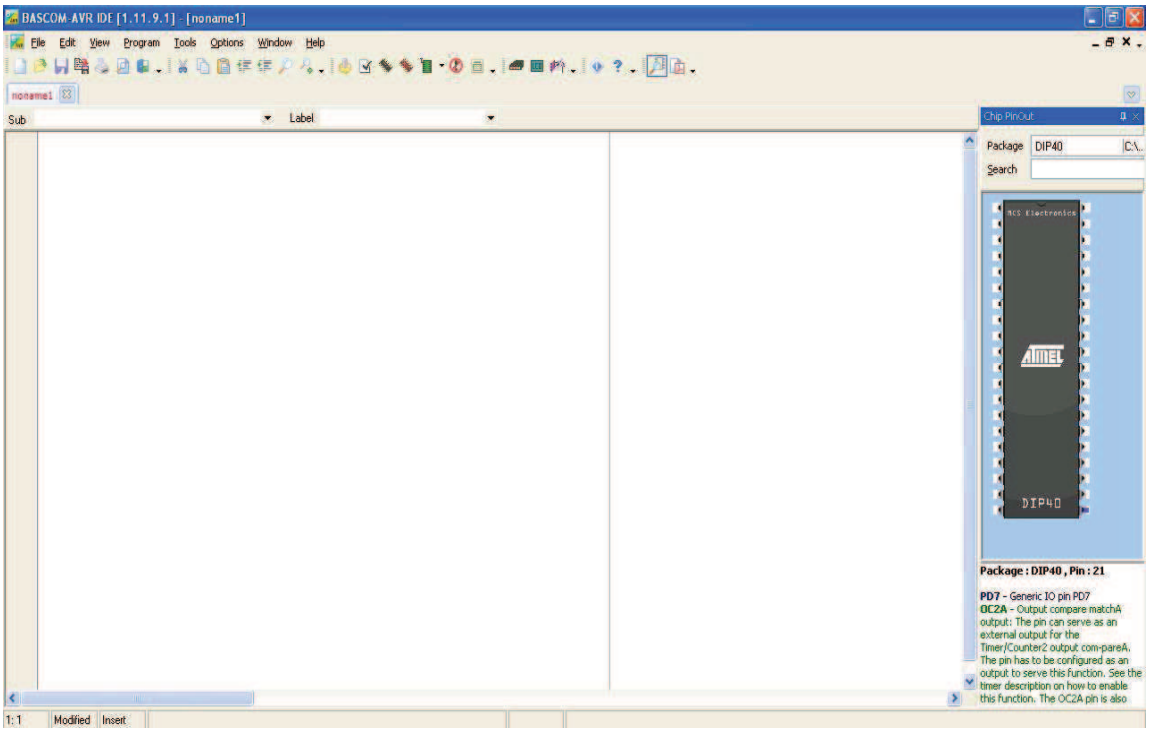


Figura 1.20: Ambiente del BASCOM AVR

1.8.1 ÍCONOS MÁS IMPORTANTES PARA MANEJAR BASCOM AVR.

ÍCONO	DESCRIPCIÓN
	**New:** Crea un nuevo archivo.
	**Find Text:** Busca un determinado texto en el programa.

[19] Valencia R. "Aplicaciones Electrónicas con Microcontroladores"

Compile Program: Compila el proyecto creado para obtener el archivo .hex que será grabado en el microcontrolador.

Al compilar un programa presionando el icono de la barra de herramientas o F7, si no existe ningún error se observa el cuadro de confirmación mostrado como se ve en la Figura 1.21.

Figura 1.21: Cuadro de Confirmación de Compilación

En el cual es posible visualizar el porcentaje de memoria que se encuentra utilizado del microcontrolador.

Pin Layout: Este icono nos muestra el integrado utilizado y también las características de cada pin cuando este es seleccionado con el mouse.

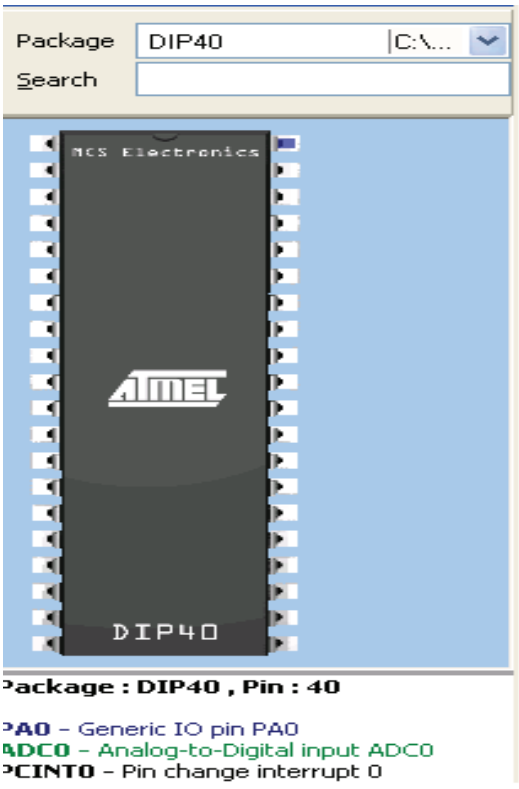


Figura 1.22: Integrado Utilizado

1.8.2 DIRECTIVAS DEL COMPILADOR

1.8.2.1 Principales Sentencias de BASCOM AVR

Son las instrucciones iniciales que el programa requiere para determinar las características del integrado, estas son las siguientes:

1.8.2.1.1 $regfile

Esta instrucción siempre va al inicio del proyecto a realizar, determina el microcontrolador que será utilizado.

Por ejemplo si vamos a utilizar:

- ATMEGA 48 → **$regfile**=”m48def.dat”
- ATMEGA 16 → **$regfile**=”m16def.dat”
- ATMEGA 8 → **$regfile**=”m8def.dat”

1.8.2.1.2 $crystal

Esta instrucción permite determinar la frecuencia de oscilación con la que va a funcionar el microcontrolador.

Ejemplo:

$crystal=1000000 para 1Mhz
$crystal=8000000 para 8Mhz

1.8.2.2 Configuraciones Iniciales

Inicializan un pin o grupo de pines para que realicen una tarea específica.

1.8.2.2.1 Config

Esta instrucción específica la configuración de un pin, un puerto o un dispositivo, ya que pueden ser configurados como entradas o salida de datos.

Ejemplo:

Config portb = output Declara todo el puerto B como salida.
Config pina.0 = input Pin A.0 como entrada.

1.8.2.2.2 DDRx, PORTx, PINx

DDR, PORT Y PIN son registros que nos permiten utilizar el puerto como entrada o salida de datos.

- **DDR:** Configura el pin como entrada o salida de datos.
- **PORT:** Es el registro de salida de datos.
- **PIN:** Es el registro de entrada de datos.

La Tabla 1.6, muestra las combinaciones para cada uno de sus pines para que funcionen en configuración especial.

Tabla 1.6: Configuración Especial de Pines.

DDRx	PORTx	I/O	Pull up	Comentario
0	0	Entrada	No	Tercer estado (Alta impedancia)
0	1	Entrada	Si	
1	0	Salida	No	Salida Push-Pull en Cero
1	1	Salida	No	Salida Push-Pull en Uno

Ejemplo:

Ddrb.0 = 1: Portb.0 = 0: Salida_1 Alias Portb.0 Puerto definido como salida.
Ddrd.7 = 0: Portd.7 = 1: Entrada_1 Alias Pind.7 Puerto definido como entrada.

Es importante recalcar que cuando se configura un puerto como salida, se debe ocupar la palabra PORT y si se lo configura como entrada se usa la palabra PIN.

1.8.2.2.3 Alias

Sirve para dar un nombre a un pin o puerto dentro de un proyecto, facilitando el uso del pin ya que es más fácil recordar la función que realiza el puerto.

Ejemplo:

Ddrb.0=1: Portb.0=0: Foco **Alias** Portb.0 Puerto definido como salida
Foco=1 Lleva a 1 lógico el pin declarado como Foco

1.8.2.3 Tipos de Datos

Los datos en un programa deben ser declarados según el tipo de variable a utilizar, con un criterio lógico para poder igualar o realizar cálculos con dichas variables, evitando que se presenten errores de dimensionamiento.

1.8.2.3.1 Dim

Dimensiona el tipo de variable que se va a utilizar, la tabla 1.7 indica los tipos de variables que puede utilizar BASCOM AVR.

Tabla 1.7: Tipos de Variables

TIPO	DIMENSIÓN
Bit	0 - 1
Byte	0 a 255
Word	0 a 65535
Long	-2147483648 a 2147483647
Integer	-32768 a 32767
Single	1.5 x 10-45 x 3.4 x 1038
String	Cadena de caracteres máximo 254
Double	5.0 x 10324 a 1.7 x 10308

Ejemplo:

Dim Dato_1 **As Byte** Declara Dato_1 como byte.

Dim Dato_2 (10) **As Byte** Declara Dato_2 como una matriz de bytes de 10 elementos.

Dim Dato_3 **As String *10** Declara Dato_3 como una String de 10 elementos.

Dim Dato_4 **As Word At $100** Declara Dato_4 como una Word ubicados en la dirección 100 de la memoria.

Dim Dato_5 (2) **As Byte At $100 Overlay** Declara Dato_5 en la misma dirección de la memoria de Dato_4.

Cuando se declara variables en una dirección específica de la memoria se tiene control sobre esta, para poder dividir los datos en una cadena de caracteres o grupo de bytes declarando otra variable en la misma dirección, esto se ilustra en la tabla 1.8, asumiendo que **Dato_4** tiene cargado el valor hexadecimal 27B5.

Tabla 1.8: Variables Declaradas en una Misma Dirección.

Dato_4		Dato_4 tiene el valor **27B5**
27	B5	
Dato_5		El byte menos significativo de Dato**_5 es B5**
27	B5	El byte más significativo de Dato**_5 es 27**

1.8.2.4 Manipulación de Bits

1.8.2.4.1 Reset

Con este comando se lleva un pin del microcontrolador al estado de 0 lógico.

Ejemplo:

```
Ddrb.0 = 1: Portb.0 = 0: Foco   Alias   Portb.0
Reset Foco
```

1.8.2.4.2 Toggle

Este comando sirve para complementar el estado anterior de alguna variable o pin de algún puerto.

Ejemplo:

```
Ddrb.0 = 1: Portb.0 = 0: Foco   Alias   Portb.0
Foco    =1
Toggle Foco                 Complementa Foco -> Foco=0
Toggle Foco                 Complementa Foco -> Foco=1
```

1.8.2.5 Manipulación de Strings

1.8.2.5.1 INSTR

Retorna la posición de una substring en una string

var = **INSTR**(Start, string, substr)
var = **INSTR**(string, substr)

- **Var:** Variable numérica donde será asignada la posición de la substring en la string. Retorna cero cuando no es encontrada.
- **Start:** Parámetro opcional que permite asignar la posición de inicio donde se desea que busque la substring. Por default cuando no es usada empieza la búsqueda desde la primera posición.
- **String:** String principal.
- **Substr**: String a encontrar.

Ejemplo:

Dim bit_posicion as Byte
Dim s_principal as string x 30
Dim s_buscar as string x 2
s_principal = “This is a test”: s_buscar = “a”
bit_posicion = **Instr** (1, s_buscar _principal, z) -> bit_posicion = 9

1.8.2.5.2 MID

La función MID retorna parte de una string a una sub string.
var = **MID(**var1, st [, l] **).**

La función MID remplaza parte de una variable de una string por otra string.
MID (var, st [, l] **)** = var1

- **var**: La string que es asignada.
- **var1:** La string fuente.
- **st**: Es la posición de inicio.
- **l**: Define el número de caracteres a conseguir o modificar

Ejemplo:

Dim **S_pr As** String x 15, **Z** As String * 15

S_pr = “ABCDEFG”	Z=”12345”
z = **Mid**(S_pr,2,3) → z=BCD	**Mid**(S_pr,2,2) = z → S_pr =A12DEFG
Z se carga con 3 caracteres de **S_pr** seleccionados desde la segunda posición	A **S_pr** se adicionan los primeros 2 caracteres de **Z** desde la segunda posición

Además se debe tener en cuenta que la posición inicial de una string siempre empieza en 1.

1.8.2.6 Instrucciones de Uso General

1.8.2.6.1 Wait:

Esta instrucción permite realizar una pausa, ya sea en segundos, milisegundos y microsegundos respectivamente.

Ejemplo:

Wait 3	Espera 3 segundos
Waitms 700	Espera 700 milisegundos
Waitus 500	Espera 500 microsegundos

1.8.2.6.2 Incr:

Incrementa el valor de una variable

Ejemplo:

Dim **A** As byte
Incr A

1.8.2.6.3 Decr

Decrementa el valor de una variable

Ejemplo:

Dim **A** As byte
Decr A

1.8.3 SÍMBOLOS OPERADORES

Dentro de los operadores, pueden utilizarse los matemáticos, de relación y lógicos.

Además se debe tomar en cuenta que BASCOM permite realizar operaciones únicamente con dos variables a la vez. A continuación podremos observar los operadores más comunes.

Operadores Matemáticos

Suma: a=b+c Resta: a=b-c

Multiplicación: a=b*c División: a MOD b

Operadores de relación

=	Igual	X=Y	<>	No es igual	X<>Y
<	Menor que	X<y	>	Mayor que	X>Y
<=	Menor Igual	X<=Y	>=	Mayor Igual	X>=Y

Operadores lógicos

NOT	Complemento (Negación)	AND	Conjunción (Y)
OR	Disyunción(O)	XOR	Or Exclusiva

1.8.3.1 Representación de Lógica Digital

Para la representación de un número binario o hexadecimal, dentro de BASCOM AVR, es necesario anteponer el símbolo “&”. En el caso de números decimales, no es necesario anteponer ningún símbolo.

Ejemplo:

Porta= &HC4	Número Hexadecimal
Porta= &b10000011	Número binario
Porta= 396	Número decimal

1.8.4 DECISIÓN Y ESTRUCTURAS[20]

1.8.4.1 Do – Loop

Esta sentencia crea un lazo cerrado, en el cual se ejecuta un conjunto de instrucciones de forma indefinida.

1.8.4.2 If – Then – Else

Son sentencias condicionales, que permiten condicionar la ejecución de instrucciones (Figura 1.23), basados en la evaluación entre dos o más variables usando los operadores lógicos.

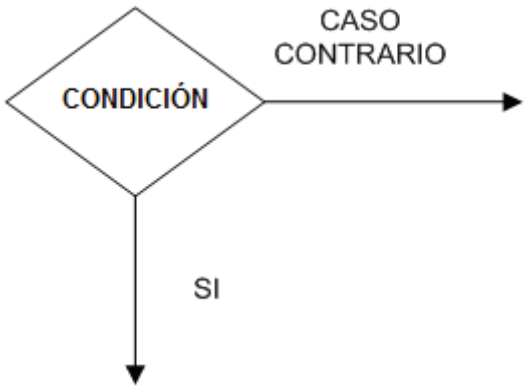


Figura 1.23: Condición Lógica IF-ELSE

[20] Valencia R. “Aplicaciones Electrónicas con Microcontroladores”

1.8.4.3 For – Next

Es una sentencia de repetición, dentro de esta sentencia se ejecutan un grupo de instrucciones hasta que se cumpla la condición que finaliza el lazo (Figura 1.24), esta condición de fin está dada por una variable que se decrementa o incrementa en pasos previamente establecidos.

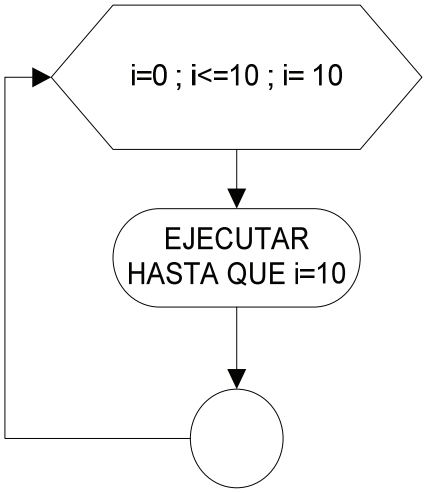


Figura 1.24: Condición de repetición FOR – NEXT

1.8.4.4 Select – Case

Son sentencias que se pueden ejecutar, dependiendo del valor de una variable de selección. Dentro de este esquema se puede tener un conjunto de casos que pueden ser ejecutados (Figura 1.25), dependiendo del valor de la variable que selecciona el caso.

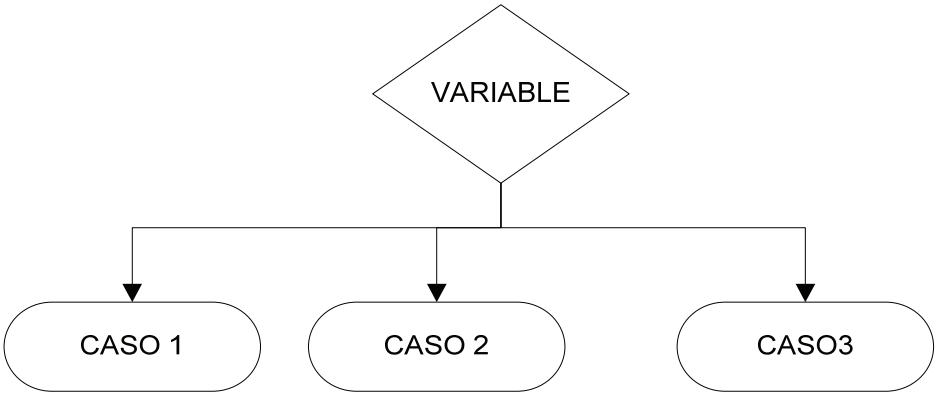


Figura 1.25: Condición de selección SELECT – CASE

1.8.4.5 Gosub

Esta sentencia obliga al programa a saltar a una subrutina, en donde ejecuta las instrucciones definidas para luego regresar y continuar con el programa.

Ejemplo:

```
Do
    Gosub Incrementar
Loop
        Incrementar:
            A=A+1
            If A > 50 And A < 60 Then Goto No_ejecutar
            B=B+5
            No_ejecutar
        Return:   Esta sentencia determina el fin de la subrutina
```

1.8.5 ESTRUCTURA DE UN PROGRAMA[21]

Cuando se estructura un programa en alto nivel, es necesario llevar un orden y vinculación de las instrucciones que se realizan.

Es primordial que se tengan estructuradas 4 partes dentro de un programa en lenguaje de alto nivel.

1. Configuraciones y Dimensionamientos de variables y subrutinas
2. Programa principal
3. Subrutinas
4. Tablas de datos

---

[21] Valencia R. “Aplicaciones Electrónicas con Microcontroladores”

El siguiente es un ejemplo de cómo se puede estructurar un programa en alto nivel, con tipos de instrucciones que se pueden realizar en su respectivo orden.

Configuraciones y Dimensionamientos	$regfile="m16def.dat" $crystal=8000000 Config Portb. Output Declare Sub Espera ()
Programa Principal	Do Instrucciones Principales en Ciclo repetitivo Loop
Subrutinas	Espera: Instrucciones de subrutina Return
Tablas de Datos	Tabla1: Data &HC0, &HF,…..

1.8.6 PROGISP 172

Al compilar un programa en BASCOM AVR se crea el archivo hexadecimal .hex el cual posee todas instrucciones que el microcontrolador necesita para funcionar, este archivo es guardado en el microcontrolador utilizando el grabador Progisp.

El grabador posee las siguientes características:

1. Se comunica con el computador mediante un puerto USB
2. Posee un jumper el cual permite alimentar al microcontrolador con el voltaje del computador o con una fuente externa.
3. Posee un jumper que selecciona la velocidad de grabación.

4. Para guardar el archivo hexadecimal el grabador posee 6 pines de conexión.

En la figura 1.26 se puede observar los pines de conexión entre grabador y el microcontrolador.

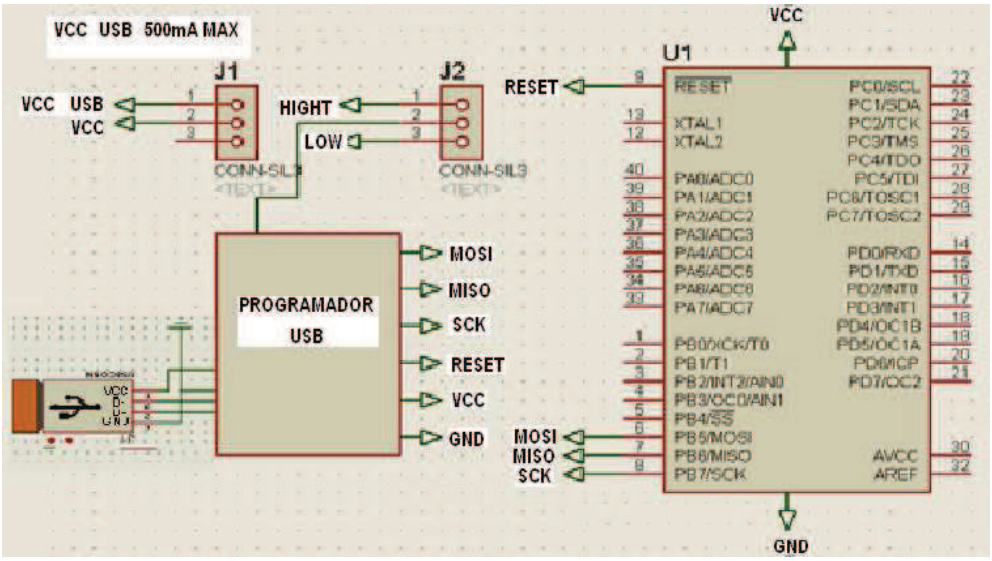


Figura 1.26: Programador USB Progisp SP 172

Los pines de conexión son: miso, mosi, sck, reset, vcc y gnd, del grabador se conectan directamente a los pines del micro que poseen la misma nomenclatura.

El grabador posee además un software amigable que permite seleccionar el microcontrolador a grabar, y sus respectivos “fuse bits”, los cuales permiten:

- Seleccionar el nivel de voltaje de funcionamiento.
- El oscilador con el cual trabajará el microcontrolador.
- Definir si el oscilador es interno o externo.

También posee opciones para la programación que permite:

- Leer el contenido del microcontrolador.
- Borrar el contenido del microcontrolador.
- Verificar si la grabación se realizó correctamente.
- Cargar automáticamente el archivo hexadecimal.
- Proteger al archivo hexadecimal contra lectura.

La figura 1.27 permite observar las configuraciones en el software de los “fuse bits” y sus opciones de programación para un integrado ATmega 164p.

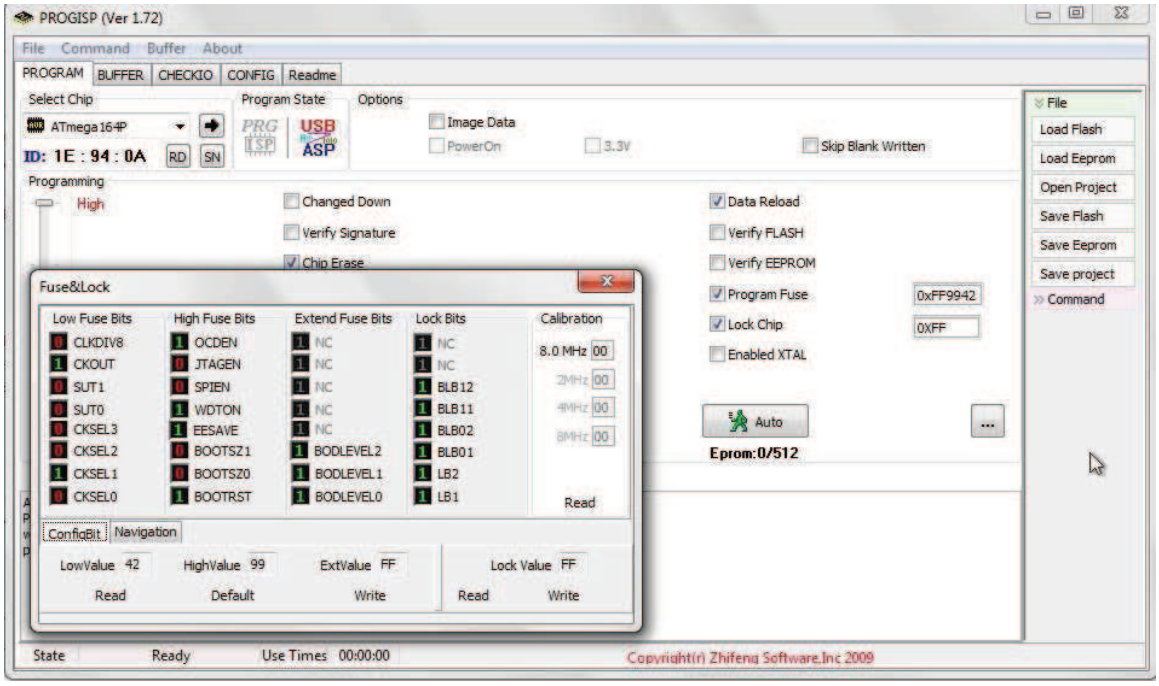


Figura 1.27: Pantalla de Grabación del Programador USB

En esta configuración al microcontrolador se asigna un oscilador externo de trabajo de 16 MHz mediante la selección de los casilleros CKSELX y se ha protegido el código del chip contra lectura mediante el uso de Lock Chip ubicado en las opciones de programación.

Se determina todas las configuraciones de los “fuse bits” mediante el uso de la respectiva hoja de datos del microcontrolador que puede ser descargada desde www.atmel.com , la tabla 1.9 indica las configuraciones de los fuse bits de CKSEL para determinar el tipo de oscilador que utilizará un microcontrolador ATmega 164p.

Tabla 1. 9: Configuración de Fases de Oscilador

FREQUENCY RANGE (MHz)	CKSEL 3.1	RECOMMENDED RANGE FOR CAPACITORS C1 and C2 (pF)
0.4 - 0.9	100	-
0.9 - 3.0	101	12 -22
3.0 - 8.0	110	12 -22
8.0 - 16.0	111	12 -22

Esta tabla determina que para un oscilador de 8 MHz a 16 MHz los pines menos significativos de CKSEL deben estar en 1, recomendando también los valores de los condensadores a utilizar con el oscilador.

1.9 LENGUAJE DE PROGRAMACIÓN VISUAL BASIC[22]

1.9.1 ENTORNO DE TRABAJO DE VISUAL BASIC

Visual Basic es un entorno de desarrollo diseñado para la creación de aplicaciones (Figura 1.28). Este lenguaje tiene las aplicaciones de un lenguaje de alto nivel con las herramientas de diseño gráfico.

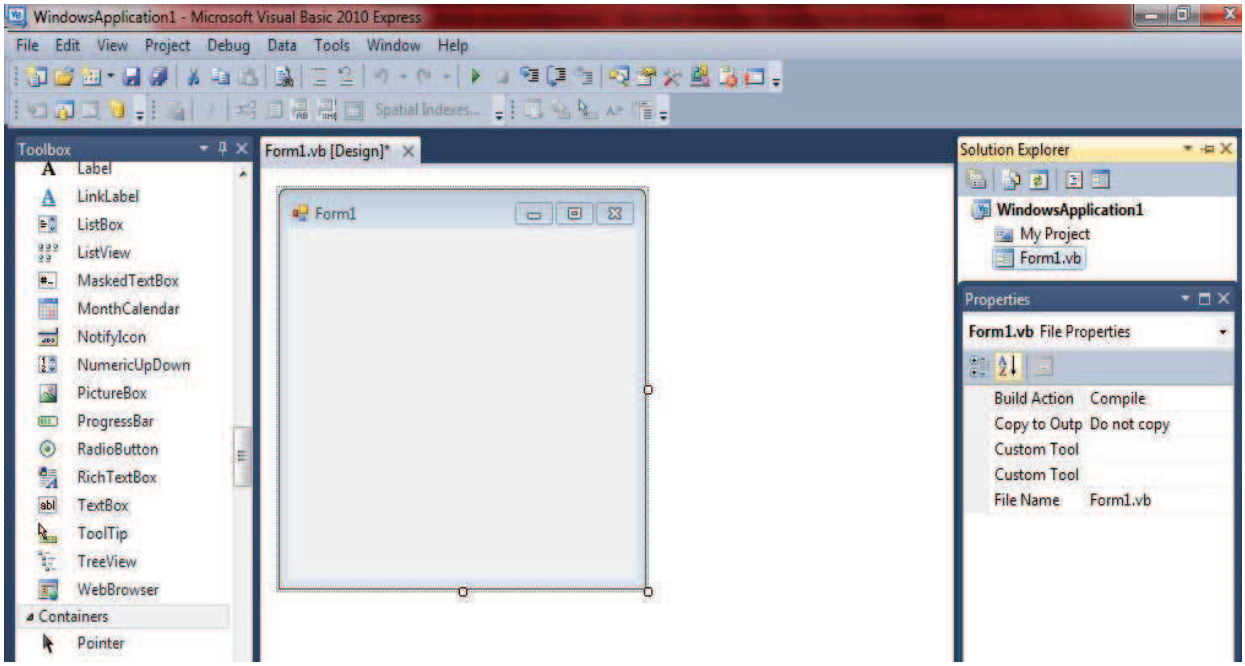


Figura 1.28: Entorno de Trabajo de Visual Basic

1.9.1.1 Barra de Herramientas

Permite un acceso rápido a los comandos más utilizados, en la figura 1.29 se ven loa iconos de los comandos.

---

[22] Pazmiño D. "Visual Basic"

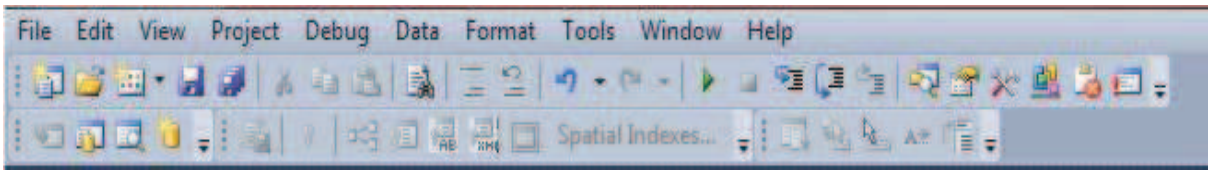


Figura 1.29: Barra de Herramientas de Visual Basic

1.9.1.2 Diseñador de Fórmulas

Es la ventana en la que se diseñará la interfaz de la aplicación (Figura 1.30), en ella se pueden agregar controles gráficos e imágenes. Cada formulario de una aplicación aparecerá en su propia ventana.

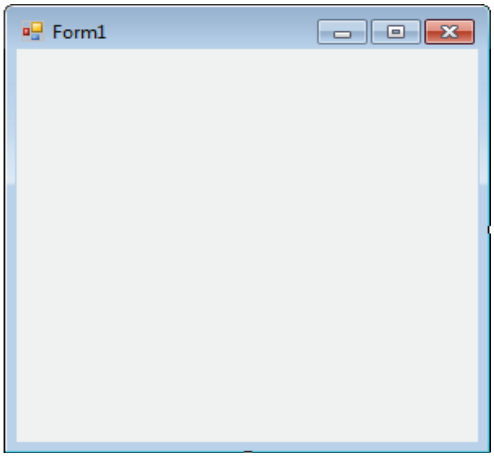


Figura 1.30: Diseñador de Formulas

1.9.1.3 Cuadro de Herramientas

En este podemos encontrar un conjunto de herramientas (Figura 1.31), que permiten insertar los objetos o controles en el formulario durante el tiempo de diseño, los objetos más comunes son botones (command), etiqueta (label), cuadro de imágenes (picture), imágenes (image), etc.

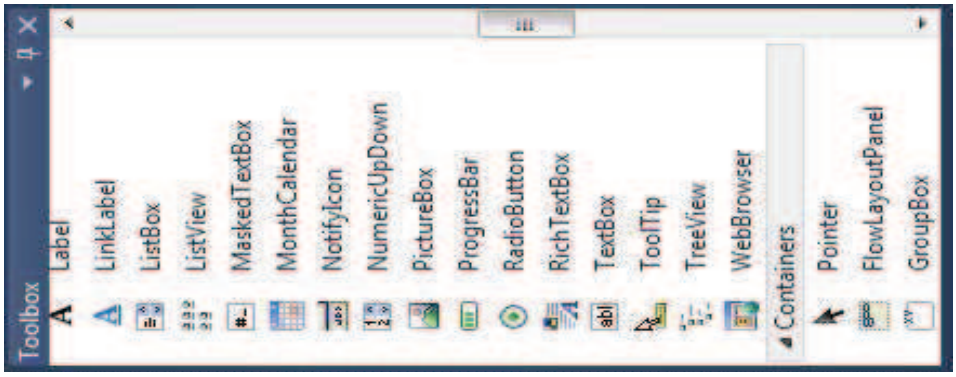


Figura 1.31: Cuadro de Herramientas

1.9.1.4 Ventana de Propiedades

Los objetos tienen asociados unas propiedades que describen sus atributos, valores, comportamiento y apariencia del objeto Al seleccionar un objeto con la lista desplegable, nos aparecerán las propiedades del mismo (name, visible, bordestyle, etc…) como se ve en la figura 1.32. En la lista de propiedades se pueden modificar las propiedades del objeto.

Las opciones de esta ventana son:

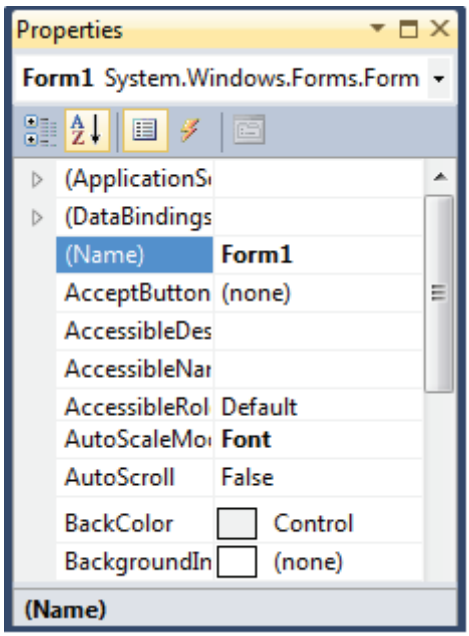


Figura 1.32: Ventana de Propiedades

1.9.1.5 Ventana de Proyecto

Contiene la lista de los archivos que forman parte de la aplicación como se ve en la figura 1.33.

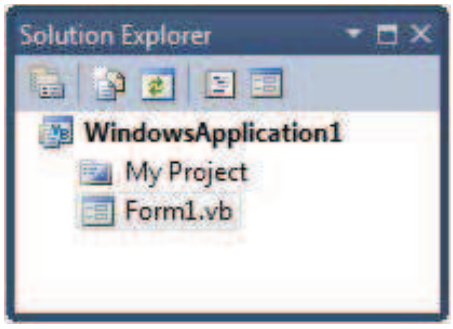


Figura 1.33: Ventana de Proyecto

Los tipos de archivos que se pueden incluir en un proyecto son:

- **Archivo de Proyecto:** Es el que realiza el seguimiento de todos los ficheros que forman parte de la aplicación con una extensión VBP.
- **Archivo de Recursos:** Aquí se guardan cadenas de texto, mapas de bits y demás datos que puedan modificarse sin tener que volver a modificar el código. Se guardan con la extensión RES.
- **Módulo de Formulario:** Contiene controles y código, solo hay uno por formulario. Se guardan con extensión FRM.

1.9.1.6 Ventana Editor de Código

En esta ventana es donde se incluye el código de la aplicación (Figura 1.34). Se creará una ventana de código para cada formulario o módulo de la aplicación, para tener acceso a la ventana de edición.

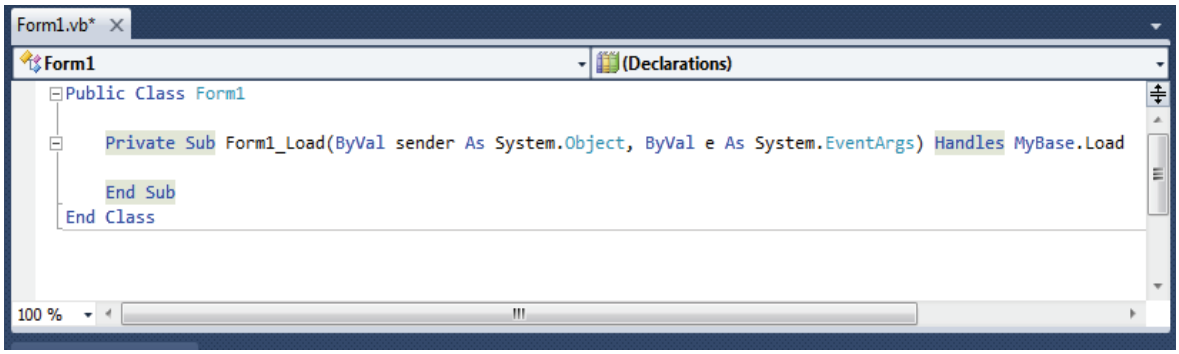


Figura 1.34: Ventana Editor de Código

1.9.2 PROGRAMACIÓN EN VISUAL BASIC

1.9.2.1 Estructura de Código

El código que se escriba en un proyecto aparecerá siempre en un módulo. En cada módulo, el código se divide en dos secciones: **declaraciones** y **procedimientos.**

- Procedimientos

Son unidades de código como pequeños programas, escritos para realizar funciones determinadas, con un propósito bien definido.

En cualquier módulo, el programador dispone de una sección especial llamada GENERAL en las que se sitúan las **declaraciones** y en las que se pueden incluir otros procedimientos creados por el programador.

- Declaraciones

En el apartado declaraciones se puede introducir las constantes, variables y tipos de datos que se necesite en la aplicación.

Los procedimientos pueden tener parámetros especificados entre paréntesis que le permiten comunicar al procedimiento alguna información que necesite o que sea el propio procedimiento quien devuelva algún valor.

En un módulo de formulario el código que se sitúa se refiere tanto a dicho formulario como al resto de objetos que estén dibujados en él. En este tipo de módulos cabe destacar los procedimientos de eventos que se encargan de dar una respuesta programada a los eventos que ocurren en la aplicación. Si ha escrito código para algún procedimiento de evento, éste aparece en negrita en la lista de procedimiento de la ventana de código.

El código de carácter general o que se puede compartir en más de un proyecto, se sitúa en un módulo general. El código que aparece en este tipo de módulo no se relaciona con un objeto determinado, sino que tiene carácter general. En los módulos generales no se pueden incluir eventos como en el caso de los módulos de formulario.

Es conveniente añadir comentarios a las líneas de código que se escribe, de esta forma se podrá entender los programas aunque pase mucho tiempo de haberlos escrito. Para añadir un comentario en una línea se utiliza el carácter “apóstrofe”.

1.9.2.1.1 Establecer Propiedades

Cuando se inserta objetos en un formulario se tiene que establecer algunas propiedades que presenta, las propiedades son aquellas características propias del objeto que hacen que se distingan de otro objeto.

- **BorderStytle:** Establecer el estilo del borde del formulario.
- **Caption:** Establece el texto que aparece en la barra de título del formulario.
- **ControlBox:** Permite mostrar o no el menú de control de las ventanas de Windows.
- **Enabled:** Establece si el formulario puede responder o no a los eventos que genere.
- **Font:** Establece las características de los objetos de texto que se sitúen en el formulario.

1.9.2.1.2 Conversión para Nombrar Objetos en Visual Basic

Tabla 1.10: Conversión para Nombrar Objetos en Visual Basic

OBJETO	OBJETO ESPAÑOL	PREFIJO
Form	Formulario	frm
CheckBox	Casilla de verificación	chk
ComboBox	Cuadro combinado enlazado a datos	cbo
Button	Botón de comando	cmd
Group Box	Marco	grb
Grid	Rejilla	grd
Label	Etiqueta	lbl
Text Box	Cuadro de texto	txt
Timer	Temporizador	tmr

1.9.2.2 Controles Básicos

Son los elementos gráficos que aparecen en los formularios y que sirven para obtener datos y presentar a la salida que produce la aplicación. Entre los numerosos controles cabe destacar los básicos, que aparecen en casi todas las aplicaciones.

1.9.2.2.1 Entrada de Datos

Dos controles muy relacionados y que se utilizan en la función de entrada de datos son:

- Etiquetas

Son controles que nos permitan mostrar texto en los formularios y que tienen la particularidad de que el usuario no puede modificar.

- Cuadros de Texto

Permiten al usuario modificar su contenido, pero el tamaño del texto es fijo. El texto que se introduce puede ser tanto numérico como alfanumérico (número y letras)

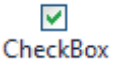
1.9.2.2.2 Control Marco

Se utiliza para estructurar el formulario en varias secciones, agrupando en éstas los controles para que la lectura sea más sencilla, la propiedad más interesante de un control marco es la propiedad **Caption,** que se refiere al texto. El control marco actúa como contenedor de otros controles.

1.9.2.2.3 Botones de Comando

Es muy sencillo, no se tiene que establecer muchas propiedades. Se puede crear teclas de acceso al botón. Su uso principal es realizar acciones en la aplicación. Para poder llevar a cabo las acciones sobre un botón pulsamos Intro, tenemos que dejar su propiedad Default a True.

1.9.2.2.4 Casillas de Verificación



Permiten establecer opciones que no son incluyentes entre sí, se puede seleccionar una o más, puede estar activa (checked), desactiva (unchecked) o atenuada (grayed) cuando el objeto no está disponible.

1.9.2.2.5 Botones de Opción

Permiten presentar opciones al usuario, pero con la particularidad que sólo se puede seleccionar solo una de las opciones al mismo tiempo.

1.9.2.2.6 Cuadro de Lista

El cuadro de lista es otra forma de presentar las opciones a través de una lista donde se sitúan dichas opciones.

1.9.2.3 Fundamentos de la Programación

1.9.2.3.1 Variables

Una variable es una ubicación temporal de memoria donde se almacenan datos que interesan tener durante la ejecución de la aplicación, éstas pueden contener texto, valores numéricos, fechas o propiedades de cierto objeto.

Las variables se caracterizan por un nombre que las identifican y por un tipo de datos, que establece el número de valores posibles que pueden contener y operaciones en las que pueden participar.

Es necesario declarar las variables para poder utilizarlas en el programa, en Visual Basic no es obligatorio pero si recomendable.

Ejemplo:

```
Dim Variable
Variable = txtEntrada.Text
txtSalida. Text = Variable
```

La forma de declarar una variable es a través de la instrucción **Dim.** En la línea se declara la variable de nombre **Variable.**

En la segunda línea ya se utiliza la variable, en este caso sirve para guardar el valor que existe en un cuadro de texto llamado **txtEntrada** (Representado por la propiedad Text).

En la tercera línea se hace justo lo contrario, se utiliza la variable para establecer el valor de la propiedad **Text** del cuadro del texto **txtSalida.**

Si añadimos las siguientes líneas de código:

Variable = 126
txtN&uacutemero. Text = Variable

Se ha establecido una variable de valor numérico, sin embargo en las anteriores líneas se había establecido un valor de texto ya que la propiedad Text es de ese tipo de datos, en la última línea se vuelve a utilizar la variable como origen de la propiedad Text del cuadro de texto txtNúmero.

1.9.2.3.2 Tipos de Datos

El tipo de datos de una variable establece el número de valores que ésta puede tener, así como el conjunto de operaciones en las que puede tomar parte como operando. Es importante indicar el tipo de datos ya que no todos tienes la misma representación en memoria, ocupando distinto espacio físico.

Así, al utilizar una instrucción como Dim Nombre Variable, se especifica implícitamente el tipo Variant para dicha variable, ésta es un tipo especial de datos que pueden contener cualquier clase de datos excepto cadenas de longitud fija y tipos definidos por el usuario.

Las variables de tipo Variant son muy flexibles, pero ocupan mucha memoria como se puede ver en la tabla 1.11.

Tabla 1.11: Tamaño del Tipo de Datos

TIPOS DE DATOS	TAMAÑO
Entero (Integer)	2 bytes
Entero Largo (Long)	4 bytes
Simple (Single)	4 bytes
Doble (Double)	8 bytes
Moneda (Currency)	8 bytes
Cadena de caracteres (String)	1 byte x carácter
Byte	1 byte
Boleano (Boolean)	2 bytes
Fecha (Date)	8 bytes
Objeto (Objest)	4 bytes
Variant	16 bytes+1 byte por cada carácter

1.9.2.3.3 Constantes

Las constantes son semejantes a las variables, pero su valor no puede cambiar a lo largo de la aplicación y para utilizarla hay que declararla previamente. La forma de declararla es a través de la instrucción:

Const NombreConstante = Expresión, donde la expresión será un valor literal o un conjunto de palabras que se evalúen a un valor válido.

1.9.2.3.4 Operadores

Existen un gran número de operadores que se pueden utilizar para crear fórmulas, en la tabla 1.12 podemos ver algunos de ellos.

Tabla 1.12: Operadores de Visual Basic

OPERADOR	OPERACIÓN QUE REALIZA
+	Suma/Concatenación de cadenas de caracteres.
-	Resta
*	Multiplicación
/	División
\	División entera.
Mod	Resto de la división entera.
^	Exponenciación
&	Concatenación de cadena de caracteres.

1.9.2.3.5 Estructuras de Decisión y Repetición

Visual Basic incorpora estructuras de control que permiten controlar el flujo de la ejecución de un programa.

Si no existen estas estructuras, el código se ejecutará de arriba hacia abajo y de derecha a izquierda según se haya escrito.

Entre las estructuras de control cabe citar las estructuras de decisión y las estructuras de repetición:

- **Estructuras de Decisión**

La instrucción If, Then, Else, es la estructura clásica de decisión.

if: En un algoritmo representativo de un problema real, es prácticamente imposible que todo sea secuencial. Es necesario tomar decisiones en función de los datos del problema.

Un ejemplo de la estructura if

```
if (condición1) then
[Acciones_por_Condición_Verdadera]….
else
[Acciones_por_Condicion_Falsa2]
End if
```

Es posible que existan más de una cláusula else, If, then en la misma instrucción If por ello los puntos suspensivos.

Si la condición 1 se cumple, entonces se ejecutará el bloque de las instrucciones 1, en caso contrario se ejecutará el bloque de instrucciones 2.

- **Estructuras de Repetición**

Otro tipo de estructuras que pueden modificar la ejecución de un programa son las estructuras de repetición o bucles. Estas estructuras sirven para repetir una y otra vez un conjunto de instrucciones.

Existirán 2 tipos de estructuras de repetición: aquellas en las que se conoce el número de repeticiones y en que dicho número se establece durante la ejecución.

La estructura de repetición For…Next es adecuada cuando conocemos el número de veces que debe repetirse un conjunto de instrucciones y deseamos reducir la cantidad de código escrito.

For contador = principio To fin [Step incremento]
[Instrucciones]
[Exit For]
[Instrucciones]
Next [contador]

Donde el contador es el nombre de una variable que sirve como contador de las veces que se tiene que ejecutar el bucle. A dicha variable se le asigna un valor inicial y un valor final en el que, una vez superado, el bucle no vuelve a repetirse. En el cuerpo del bucle estarán las instrucciones que deben ejecutarse existiendo la posibilidad de introducir Exit For para salir del bucle.

Otra estructura de repetición es Do…Loop, ésta estructura se utiliza cuando desconocemos cuantas veces se ha de ejecutar el bucle. La sintaxis es:

Do [While│Until] condición
[instrucciones]
[Exit Do]
[instrucciones]
Loop

1.9.2.3.6 Condición

Es una expresión, es decir, un conjunto de palabras que se evalúan a verdadero o falso sin posibilidad de poder tener otro valor. Se dice que una condición se cumple cuando se evalúa a verdadero y que fracasa en caso contrario (Tabla 1.13). Las condiciones también se conocen como expresiones lógicas.

Tabla 1.13: Condiciones de Visual Basic

EXPRESIONES LÓGICAS	
=	Igual a
<>	Distinto a
<	Menor que
>	Mayor que
<=	Menor o igual que
>=	Mayor o igual que

1.9.2.3.7 Operadores Lógicos

- **AND:** exp1 And exp2, donde se evalúa a verdadero s&oacutelo en el caso de que el exp1 como exp2 se evalúa a verdadero. En cualquier otro caso se evalúa a falso.
- **OR:** exp1 And exp2, donde se evalúa a verdadero cuando alguna de las expresiones exp1 And exp2 se evalúa a verdadero.
- **NOT:** Not exp1 aquí se evalúa a verdadero si exp1 es falso y se evalúa a falso si exp1 es verdadero.
- **XOR:** exp1 Xor exp2, se evalúa sólo en caso de que una y sólo una de las expresiones: exp1 o exp2 se evalúa a verdadero.

1.9.3 EAGLE

El Eagle Layout Editor es una poderosa herramienta que permite diseñar circuitos impresos de forma fácil.

El nombre Eagle es un acrónimo de “Easily Applicable Graphical Layout Editor”, el programa consiste de tres principales módulos editor de trazado, editor esquemático y auto ruteado, pudiendo editar con estos módulos los archivos que formarán parte de un circuito impreso.

En la tabla 1.14[23] se muestra un listado con los tipos de archivos más importantes que pueden ser editados con Eagle.

Tabla 1.14: Archivos de Eagle

Tipo	Ventana	Nombre
Placa	Editor de líneas de conexión	*.bdr
Esquema	Editor de esquemas	*.sch
Librería	Editor de librerías	*.lbr

[23] http://picmania.garcia-cuervo.net/eagle_tutlbr_i_library.php

1.9.3.1 Editor de Esquemas

En esta ventana se realiza el diagrama esquemático del circuito con todas las conexiones que requiere el diseño.

La figura 1.35 permite observar la ventana del editor de esquemas con un circuito interno.

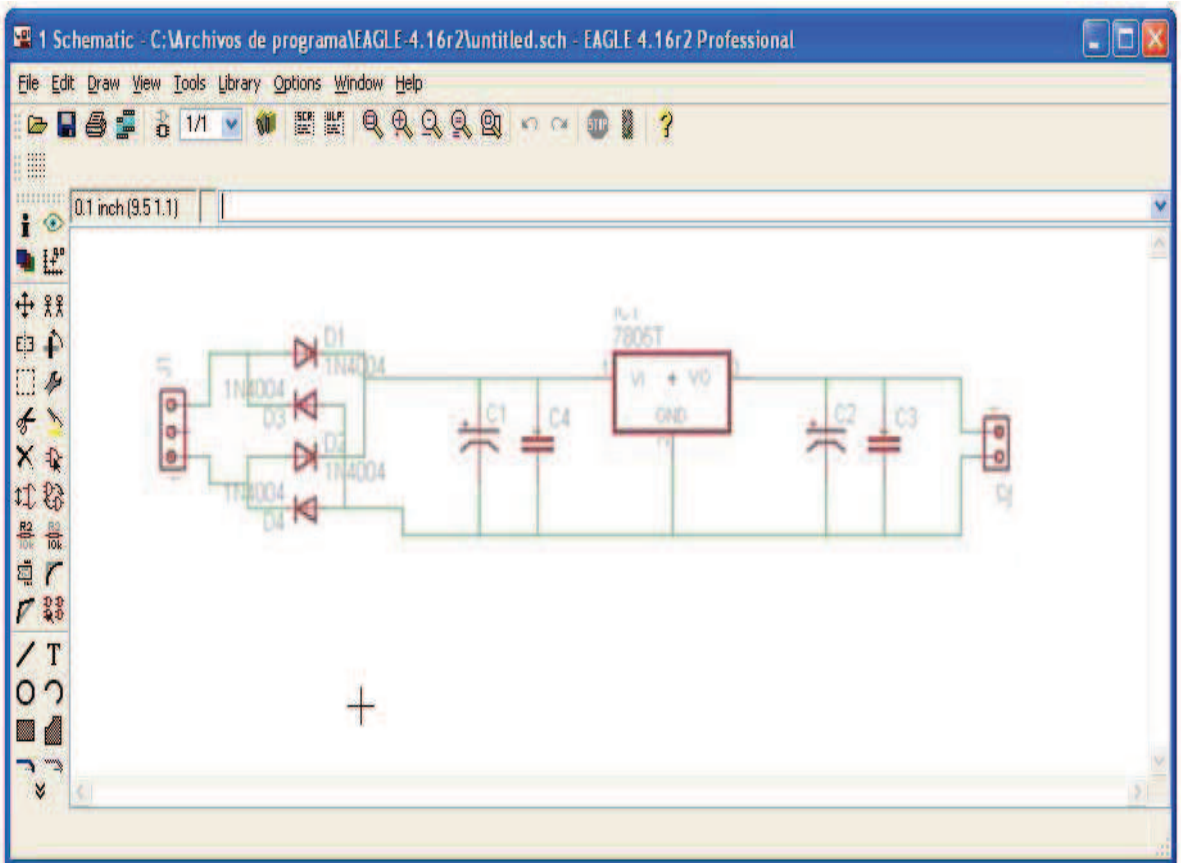


Figura 1.35: Archivos de Eagle

1.9.3.2 Editor de Líneas de Conexión

En este archivo los componentes del circuito, creados en el diagrama esquemático son observados conservando su forma real y ocupando un espacio definido.

El editor de líneas permite crear las pistas de conexión entre los elementos que forman parte del circuito estableciendo un tamaño y forma definida para que finalizado su diseño pueda este ser imprimido, transferido a una baquelita y quemado.

La figura 1.36 muestra el circuito del diagrama esquemático finalizado en el editor de líneas con sus respectivas conexiones.

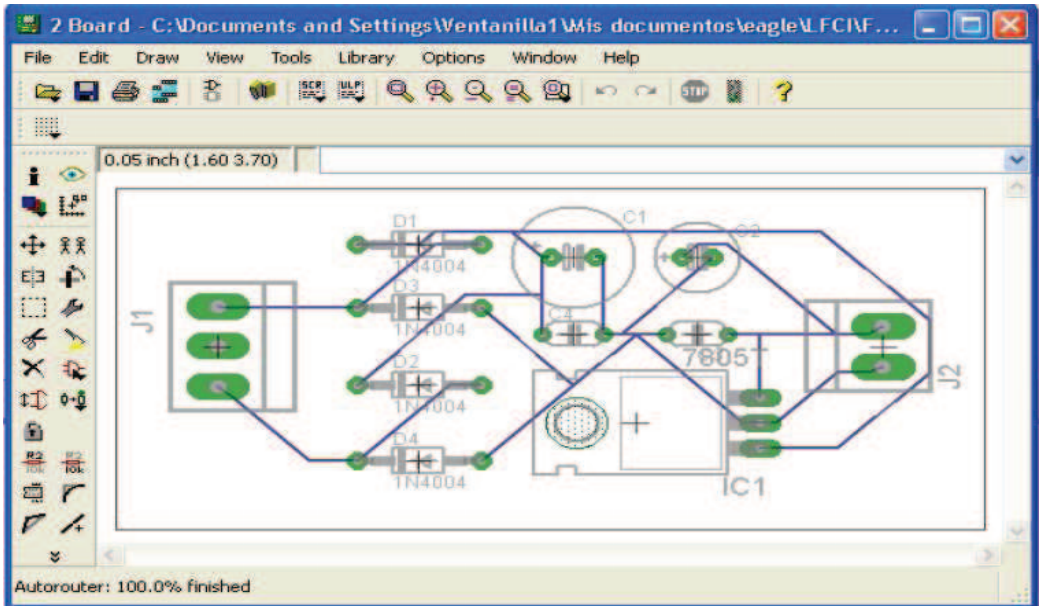


Figura 1.36: Editor de Líneas de Conexión

1.9.3.3 Editor de Librería

Eagle facilita el diseño de un circuito, debido a que permite crear librerías propias para elementos que no sean encontrados dentro de sus librerías disponibles o que necesiten ser modificados en alguna de sus características, permitiendo crear un componente con conexiones específicas.

Una librería está compuesta de uno o varios componentes electrónicos. Cada elemento tiene tres archivos, uno a utilizar en el editor esquemático, otro para el editor de líneas y uno que será presentado en el panel de control con sus características.

Todos estos archivos poseen una conexión común entre pines de entrada, salida o alimentación para formar un elemento.

En la figura 1.37 se puede observar el panel de control con la librería de un elemento presentando su diagrama esquemático y el paquete que será utilizado en el editor de líneas de conexión.

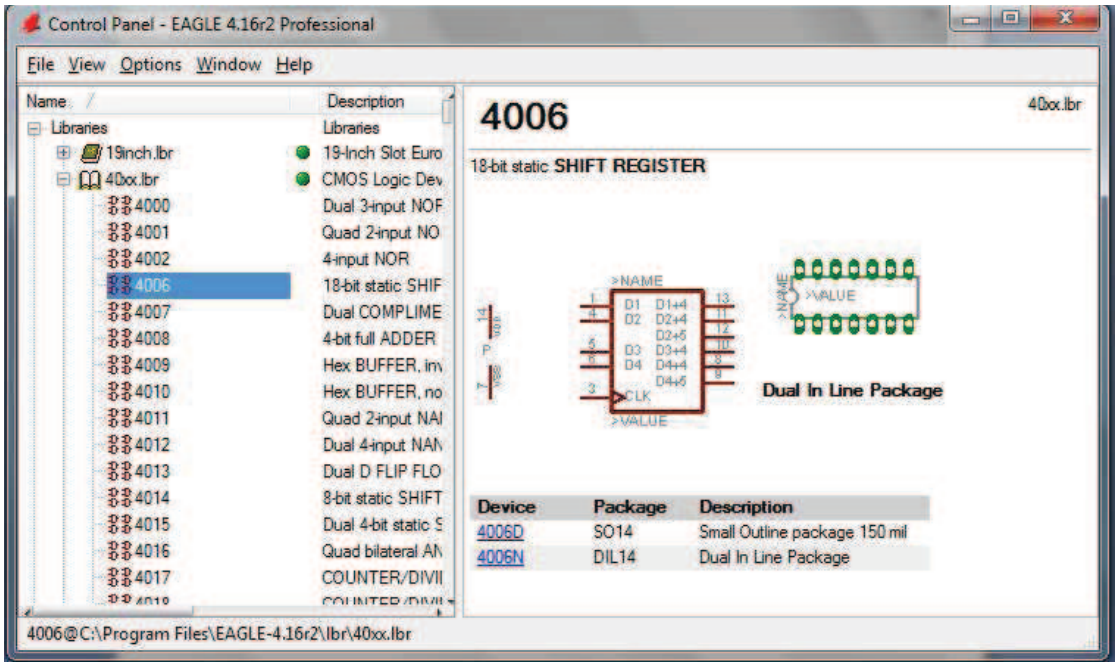


Figura 1.37: Panel de Control

CAPÍTULO II

CONSTRUCCIÓN DE UN PROTOTIPO DE CONTROL Y REGISTRO DE PERSONAL BASADO EN IDENTIFICACIÓN POR RADIOFRECUENCIA.

El presente capítulo describe el diagrama de bloques presentado en la figura 2.1 que conforman el prototipo de control y registro de personal, indicando la manera en la que cada bloque interactúa para que un usuario pueda ser registrado y de esta manera permitirle al mismo ingresar a un área de trabajo determinada.

2.1 DIAGRAMA DE BLOQUES

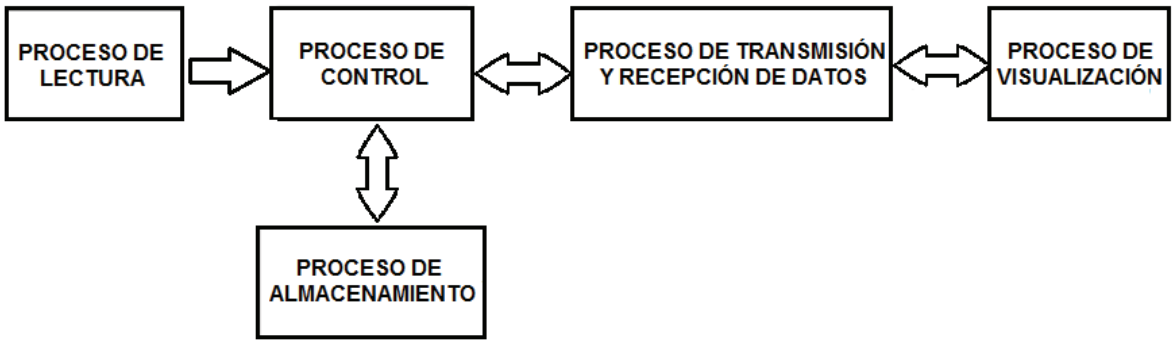


Figura 2.1: Diagrama de Bloques

2.1.1 PROCESO DE CONTROL

Es el principal módulo de este prototipo ya que permite controlar y ejecutar las instrucciones de procesamiento, almacenamiento y comunicación con los diferentes bloques del sistema permitiendo que interactúe con el módulo ID-12, la memoria 24LC256, el integrado DS1307 y se comunique serialmente con la computadora para enviar o recibir diferente información.

2.1.1.1 ATmega 164P

El microcontrolador ATmega 164P (Figura 2.2) controla todo el proceso que gobierna al control y registro del personal utilizando sus pines para realizar funciones definidas.

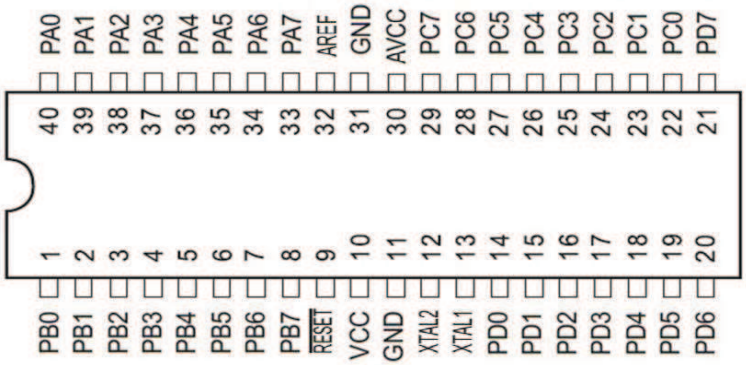


Figura 2.2: ATmega 164P

La Tabla 2.1 muestra el nombre o alias que tiene cada pin utilizado del microcontrolador.

Tabla 2.1: Funciones de Pines

# PIN	PIN	FUNCIÓN	DESCRIPCIÓN
1	B0		JUMPER 1 PRIMERA PUERTA
2	B1		JUMPER 2 SEGUNDA PUERTA
3	B2		JUMPER 3 TERCERA PUERTA
6	B5	MOSI	Pin para Programación
7	B6	MISO	Pin para Programación
8	B7	SCK	Pin para Programación
9	RS	Reset	Pin para Programación - Conectado a Vcc
10	VCC		Pin de habilitación del microcontrolador
11	GND		Pin de habilitación del microcontrolador
12	XT1		Oscilador
13	XT2		Oscilador
14	D0	RXD	Receptor de datos seriales
15	D1	TXD	Transmisor de datos seriales
16	D2	INT 0	DETECCCIÓN CÓDIGO WIEGAN
17	D3	INT 1	DETECCCIÓN CÓDIGO WIEGAN
19	D5		LED Puerta Abierta
20	D6		LED Puerta Cerrada
22	C0	SCL	Conexión I2C
23	C1	SDA	Conexión I2C
30	AVCC		Pin de habilitación del microcontrolador
31	GND		Pin de habilitación del microcontrolador
32	AREF		Pin de habilitación del microcontrolador
38	A2	Beep	Chicharra
40	A0	Entrada	Verifica si hay conexión a pc

El microcontrolador ATmega 164P posee el programa principal que determina que código ID ha enviado la tarjeta RFIT para posteriormente determinar si el código pertenece a un usuario registrado o en caso de no estar registrado, comunicarse con la computadora serialmente notificando que se ha encontrado un nuevo usuario.

2.1.2 PROCESO DE LECTURA

El proceso de lectura está conformado por la tarjeta RFID y el módulo ID-12.

Figura 2.3: Tarjeta y Módulo ID-12

En este proceso el módulo ID-12 es el encargado de polarizar a la tarjeta RFID para que internamente realice el proceso de control y a su vez la tarjeta envíe seriamente el código de su ID único.

Posteriormente el módulo ID-12[24] al detectar el ID de la tarjeta envía el código hacia el microcontrolador, mediante la comunicación serial o mediante el código Wiegand 26, configuración que se determina mediante la conexión de sus pines de alimentación y habilitación.

2.1.2.1 Comunicación Wiegand 26

La transmisión de datos WIEGAN 26 utilizado en el prototipo usa tres pines para el envío de información. El primer hilo (pin 8) envía los unos lógicos o **DATA1**, el

---

[24] Ver Anexo 4

segundo hilo (ping 9) envía los ceros lógicos o **DATA 0** y el tercer hilo (ping 1) es la línea de masa de referencia o **GND**.

La figura 2.4 presenta la descripción de los pines del módulo ID-12

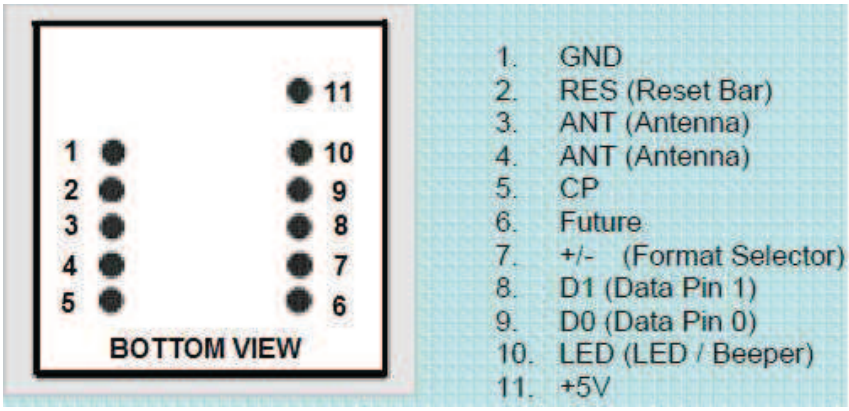


Figura 2.4: Módulo ID-12

En la tabla 2.2 indica la conexión que se debe realizar para enviar hacia el microcontrolador la ID de la tarjeta leída en formato WIEGAND26.

Tabla 2.2: Conexión de Pines para transmisión Wiegand 26

Pin No.	Descripción	Wiegand 26
Pin 1	Zero Volts and Tuning Capacitor Group	Gnd 0V
Pin 2	Strap to + 5V	Reset Bar
Pin 3	To External Antenna and Tuning Capacitor.	Disable
Pin 4	To External Antenna	Disable
Pin 5	Card Present	No function
Pin 6	Future	Future
Pin 7	Format Selector (+/-)	Strap to +5V
Ping 8	Data 1	One Output
Ping 9	Data 0	Zero Output
Ping 10	3.1 KHz Logic	Beeper/LED
Ping 11	DC Voltaje Supply	+5V

2.1.2.2 Interpretación de los Datos

En estado de reposo, sin trasmisión de datos, las líneas DATA1 y DATA0 están en nivel alto a voltaje TTL y cuando una tarjeta RFIT es detectada el módulo ID-12 envía a trevés del pin 8 y del pin 9 el ID de la tarjeta.

En esta transmisión de datos, el módulo envía 26 bits de información los cuales tienen una separación en el tiempo de 2 ms, una amplitud de 50us y un estado lógico de cero o GND.

La figura 2.5 indica el envío de dos bits mediante el código Wiegand 26.

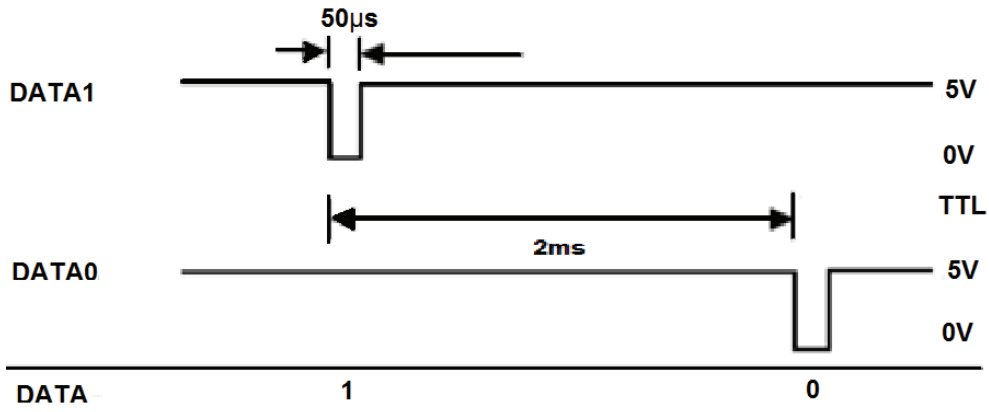


Figura 2.5: Esquema de Trasmisión de Bits

Los 26 bits de información tienen la estructura mostrada en la tabla 2.3.

Tabla 2.3: Estructura de los Datos de Salida.

FORMATO DE LA COMUNICACIÓN Wiegand 26

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
P	E	E	E	E	E	E	E	E	E	E	E	E	O	O	O	O	O	O	O	O	O	O	O	O	P
Even parity (E)													**Odd Parity (0)**												

En donde:

- El primer Bit, B0, es la Paridad Par de los primeros 12 bits transmitidos.
- Los bits B1 hasta B8 son un Byte, al que lo llaman “Facility Code”.
- Los bits desde B9 hasta B24 son dos Bytes, llamado “User Code”.
- El último bit, B25, es la paridad impar de los últimos 12 bits trasmitidos.

El módulo ID-12 también puede enviar el ID de la tarjeta RFID mediante la comunicación serial utilizando dos pines uno para transmisión y la tierra común pero se utilizó esta comunicación es porque el puerto serial del microcontrolador tiene otra función dentro de las comunicaciones con los módulos.

La figura 2.6 presenta el diagrama de conexiones entre el módulo ID-12 y el ATmega164P.

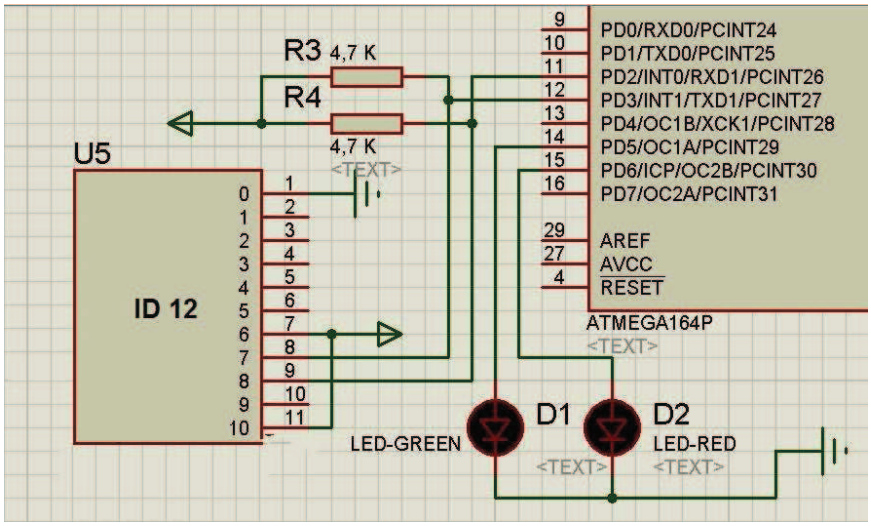


Figura 2.6: Conexiones Módulo ID-12 y ATmega 164P

2.1.3 PROCESO DE ALMACENAMIENTO

En este proceso se guarda la información de ingreso de cada usuario adjuntando la hora y fecha en la que ingresó o salió en la memoria 24LC256, para realizar esto el ATmega 164P primero determina el código que envía la tarjeta RFIT seguidamente busca en la memoria 24LC256 si el código está guardado y de ser así lee en el integrado de hora DS1307 la hora y la fecha para posteriormente guardar en la memoria el número de usuario, la hora y la fecha que ingresó o salió.

En este proceso la memoria de almacenamiento de datos y el integrado de reloj, son dispositivos que permiten la lectura y escritura de sus datos mediante la comunicación I2C por lo tanto es indispensable conocer como es utilizado este protocolo, para poder realizar una comunicación exitosa.

2.1.3.1 Control de Interfaz Integrada I2C

El protocolo I2C, es un estándar que fue diseñado por Philips en 1980, usa solo dos líneas llamadas SCL (Serial Clock) y SDA (Serial Data), facilita la

comunicación entre microcontroladores, memorias, reloj de tiempo real, sensores de temperatura y otros dispositivos.

La comunicación I2C se establece entre el maestro (master) y el esclavo (slave), representado en la figura 2.7[25], el maestro usualmente es un microcontrolador mientras que el esclavo es el dispositivo conectado al bus, El protocolo I2C establece la comunicación hasta un máximo de 128 esclavos conectados al mismo bus.

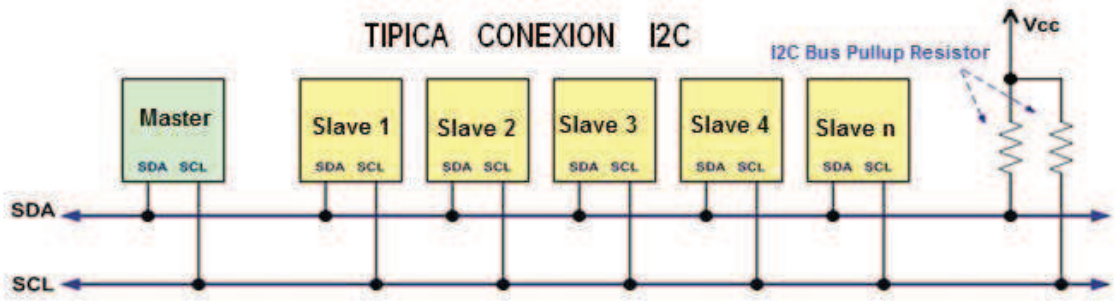


Figura 2.7: Comunicación I2C

La comunicación de datos del bus I2C es en serie y sincrónica. Una de las señales del bus marca el tiempo (pulsos de reloj) y la otra se utiliza para intercambiar datos

2.1.3.2 Descripción de las Señales

Todos los dispositivos conectados al bus I2C[26] son controlados mediante las líneas:

- **SCL (System Clock):** Es la línea de los pulsos de reloj que sincronizan el sistema.
- **SDA (System Data):** Es la línea por la que se mueven los datos entre los dispositivos.

---

[25] http://www.ermicro.com/blog/?p=744

[26] http://axxon.com.ar/rob/Comunicacion_busI2C.htm

- **GND (Ground):** Masa común de la interconexión entre todos los dispositivos "enganchados" al bus.

Las líneas SDA y SCL se deben polarizar en un nivel lógico alto (Conectando a la alimentación por medio de resistores "pull-up") lo que define una estructura de bus que permite conectar en paralelo múltiples entradas y salidas.

El valor de los resistores de polarización (pull up), puede ir desde 1.8 K Ω hasta 47K Ω . Un valor menor de resistencia incrementa el consumo de los integrados pero disminuye la sensibilidad al ruido y mejora el tiempo de los flancos de subida y bajada de las señales.

2.1.3.3 Protocolo de Comunicación del Bus I2C

Para que un dispositivo maestro pueda establecer la comunicación, el bus I2C debe estar libre, este estado se cumple cuando las señales SDA y SCL se encuentran en estado lógico alto.

En este estado cualquier dispositivo maestro puede ocupar el bus, primero establece la condición de inicio (start), poniendo en estado bajo la línea de datos (SDA), pero dejando en alto la línea de reloj (SCL), indicado en la figura 2.8[27].

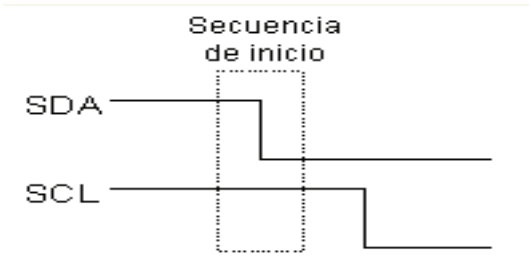


Figura 2.8: Condición de Inicio I2C

Luego de la condición de inicio, el primer byte que se transmite (figura 2.9) contiene siete bits que componen la dirección del dispositivo que se desea seleccionar, y un octavo bit que corresponde a la operación que se quiere realizar con él (lectura o escritura).

---

[27] http://axxon.com.ar/rob/Comunicacion_busI2C.htm

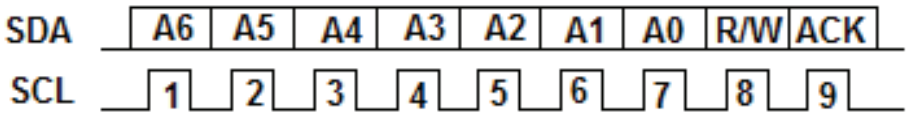


Figura 2.9: Dirección del Esclavo

Si el dispositivo cuya dirección corresponde a la que se indica en los siete bits está presente en el bus, éste contesta con un bit en bajo, ubicado inmediatamente luego del octavo bit que ha enviado el dispositivo maestro. Este bit de reconocimiento (ACK) en bajo le indica al dispositivo maestro que el esclavo reconoce la solicitud y está en condiciones de comunicarse. Aquí la comunicación se establece en firme y comienza el intercambio de información entre los dispositivos. Durante el intercambio de información los datos se transfieren en secuencias de 8 bits. Cada bit transmitido se encuentra acompañado por un pulso de reloj pues la comunicación es sincrónica.

En la figura 2.10 se puede observar el diagrama de tiempo entre las señales SDA y SCL representando los sincronismos existentes entre estas señales.

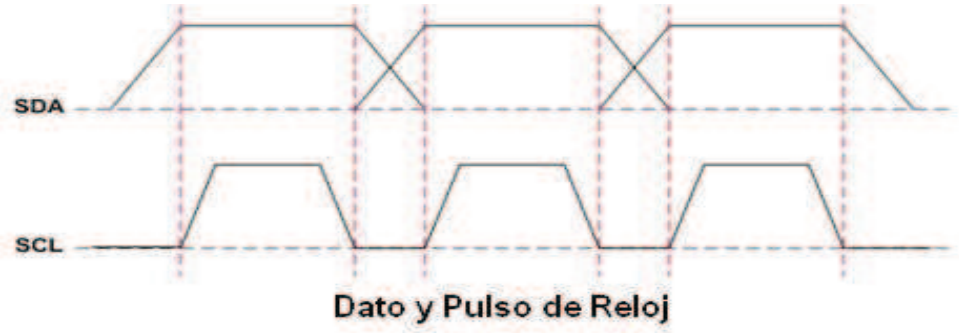


Figura 2.10: Diagrama de Tiempo

Terminado el proceso de lectura o escritura el dispositivo maestro puede dejar libre el bus generando una condición de **stop** (parada o detención).

La figura 2.11 presenta el diagrama de conexiones utilizado para conectar la memoria 24LC256 y el integrado de DS1307 con el ATmega 164P mediante el protocolo I2C.

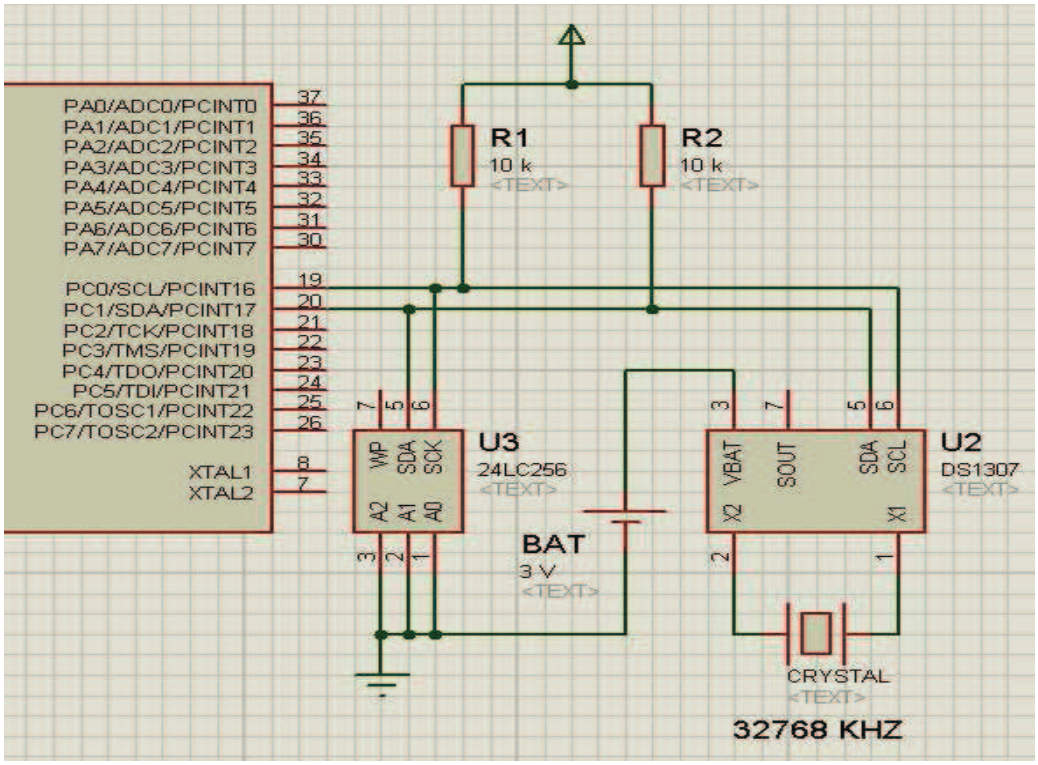


Figura 2.11: Conexiones Integrados I2C

2.1.3.4 Proceso de Escritura en un Dispositivo Esclavo

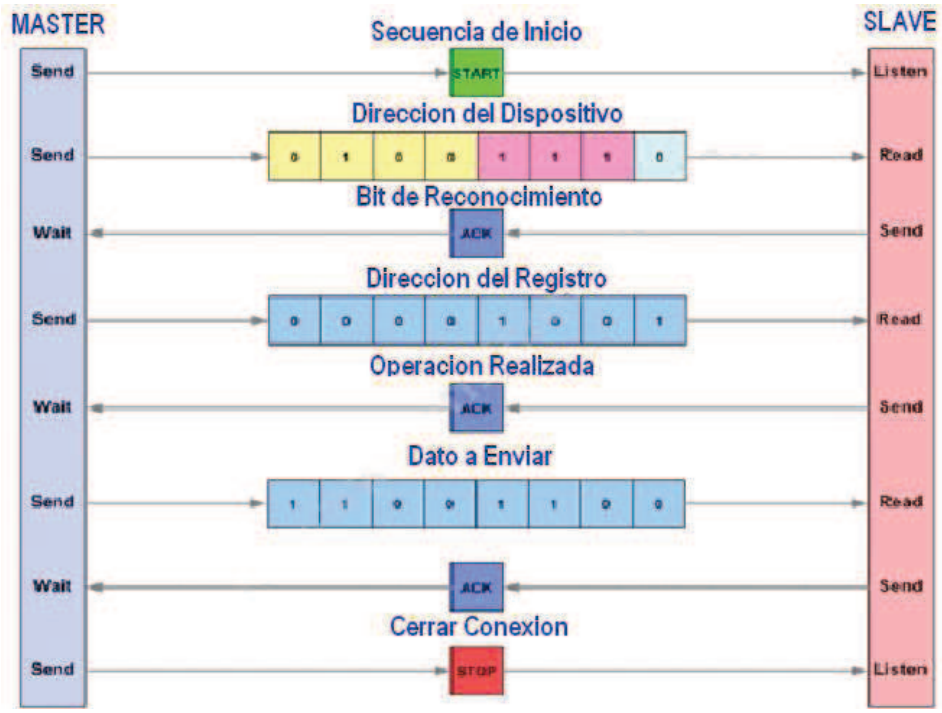


Figura 2.12: Proceso de Escritura

Cuando el maestro desea escribir en un dispositivo conectado al bus I2C debe realizar el proceso representado en la figura 2.12[28] que es descrito a continuación:

1. Enviar una secuencia de inicio.
2. Enviar la dirección de dispositivo (7 bits) con el bit de lectura/escritura (R/W) en bajo.
3. Si el dispositivo cuya dirección corresponde a la que se indica en los siete bits está presente en el bus, éste contesta con un bit de reconocimiento en bajo (ACK), ubicado inmediatamente luego del octavo bit que ha enviado el dispositivo maestro.
4. Enviar el número de registro interno en el que se desea escribir.
5. Enviar el byte de dato.
6. [Opcionalmente, enviar más bytes de dato].
7. Enviar la secuencia de parada.

2.1.3.5 Proceso de Lectura desde un Dispositivo Esclavo

La secuencia de lectura de bytes de un dispositivo esclavo representada en la figura 2.13[29], es un poco complicada pues inicia como una secuencia de escritura pero no dificulta el entender cómo funciona.

---

[28] http://www.ermicro.com/blog/?p=1239

[29] http://www.ermicro.com/blog/?p=1239

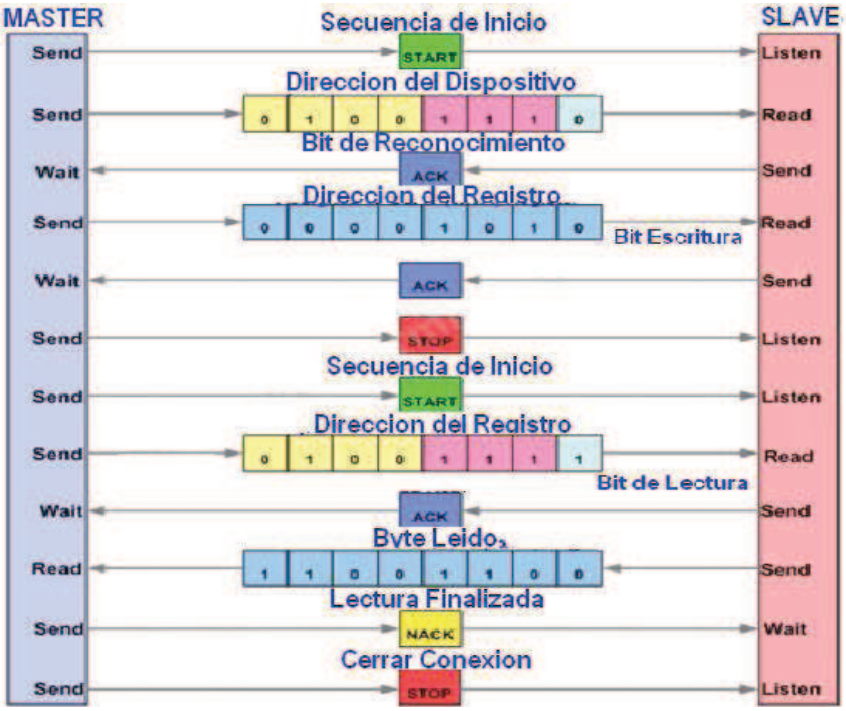


Figura 2.13: Proceso de Lectura

Los pasos a seguir para realizar la lectura son los siguientes:

1. Enviar una secuencia de inicio
2. Enviar la dirección de dispositivo con el bit de lectura/escritura (R/W) en bajo.
3. Si el dispositivo cuya dirección corresponde a la que se indica en los siete bits está presente en el bus, éste contesta con un bit de reconocimiento en bajo (ACK), ubicado inmediatamente luego del octavo bit que ha enviado el dispositivo maestro
4. Enviar el número de registro interno en el que se desea leer
5. Enviar nuevamente la dirección del dispositivo, pero esta vez con el bit de lectura en alto
6. Luego se leen todos los bytes necesarios. Durante la lectura el dispositivo a leer contesta con un bit de reconocimiento en bajo (ACK), preguntando si todavía existe más bytes por leer. El microcontrolador envía un bit (NACK), cuando desea finalizar la lectura, lo cual significa que el byte leído es el último.
7. Después se envía la secuencia de parada para terminar la transacción.

2.1.3.6 Integrado 24LC256

La memoria 24LC256 almacena información para controlar el ingreso de los usuarios, con sus respectivas características. En la tabla 2.4 se presentan los datos de la memoria organizados en un mapa de memoria.

Tabla 2.4: Mapa de Memoria

Dirección	Datos Almacenados	Cantidad de Bytes
1	Número de usuarios registrados	1 byte
2	Número de usuarios qe han ingresado	2 byte
10	Código de tag 1 y puerta que abrirá	9 bytes
20	Código de tag 2 y puerta que abrirá	9 bytes
..	NAABBCCDD	…
...	3AA23DD21	…
190	Código de tag 19 y puerta que abrirá	9 bytes
200	Código de tag 20 y puerta que abrirá	9 bytes
300	Código de ingreso del usuario	14 bytes
320	Código de ingreso del usuario	14 bytes
..	NUHHMMSSAMDDMMAA	…
...	02122456PM240712	…
2240	Código de ingreso del usuario	14 bytes
2260	Código de ingreso del usuario	14 bytes

La información que se almacena en la memoria se encuentra organizada para facilitar la realización de cálculos que permitan leer o escribir distintos datos de la misma. Desde la dirección 10 hasta la 200 corresponde al código del tag, número del usuario y puerta a la que tendrá acceso, por ejemplo la dirección 10 corresponde, el formato de bytes guardados a modo de ejemplo es el siguiente: **NAABBCCDD**

Donde:

N=Número de puerta
AABBCCDD= Código del tag

A partir de la dirección 300 se guarda el número de usuario que ingreso más la hora y la fecha de ingreso o salida, el formato de bytes guardados a modo de ejemplo es el siguiente: **NUHHMMSSAPDDMMAA**

Donde:

NU= Número de usuario que ingresó.
HHMMSS= Hora, minuto, segundo.
AP= Pude ser AM o PM.
DDMMAA= Día, mes, año.

2.1.4 PROCESO DE TRANSMISIÓN Y RECEPCIÓN DE DATOS

El microcontrolador para la transmisión y recepción de datos utiliza la comunicación serial a niveles TTL y tiene que conectarse con una computadora mediante un puerto USB, para realizar esto es necesario adquirir un convertidor de USB a RS-232 que posteriormente se conectará con el microcontrolador utilizando un MAX 232 para acoplar los niveles RS232 al TTL y viceversa.

2.1.4.1 Convertidor USB a Serial30

En la figura 2.14 se observa el convertidor de USB a Serial que permite conectar un dispositivo serial RS-232 a puerto USB en su PC de escritorio o portátil.

Figura 2.14: Convertidor USB a RS-232

[30] http://www.trendnet.com/langsp/products/proddetail.asp?prod=150_TU-S9&cat=32

Las características del convertidor son las siguientes:

- Compatible con las especificaciones USB 1.1
- Admite interfaz serial RS-232
- Admite hasta una transferencia de datos de 500kbps
- Detecta una condición de suspensión USB
- Compatible con Windows 7/Vista/XP/2000/ME/98SE/ Mac OS 10.1~10.6
- Se instala como un puerto COM de Windows estándar, señales de control de módem Full RS-232 , señales de datos RS-232; TxD, RxD, RTS, CTS, DSR, DTR, DCD, RI, GND
- Admite BUS-Power, no requiere de adaptador eléctrico externo
- Con funcionalidad Plug & Play y fácil instalación

El convertidor USB presenta la salida de datos en un conector DB9[31] macho y la información asociada a cada uno de los pines es la presentada en la tabla 2.5:

Tabla 2.5: Distribución de Pines DB9

Número de pin	Señal
1	DCD (Data Carrier Detect)
2	RX
3	TX
4	DTR (Data Terminal Ready)
5	GND
6	DSR (Data Sheet Ready)
7	RTS (Request To Send)
8	CTS (Clear To Send)
9	RI (Ring Indicator)

De estos pines solo se utiliza el pin2 (Rx) para la recepción de datos, el pin3 (Tx) para la transmisión de datos y la tierra común pin5 (GND) para conectarse con el MAX232..

[31] http://www.iearobotics.com/proyectos/cuadernos/ct1/ct1.html

2.1.4.2 MAX232, adaptador de niveles TTL a RS-232[32]

Este integrado es usado para comunicar un microcontrolador o sistema digital con un PC o sistema basado en el estándar RS232.El MAX232 convierte los niveles de las líneas de un puerto serie RS232 a niveles TTL.

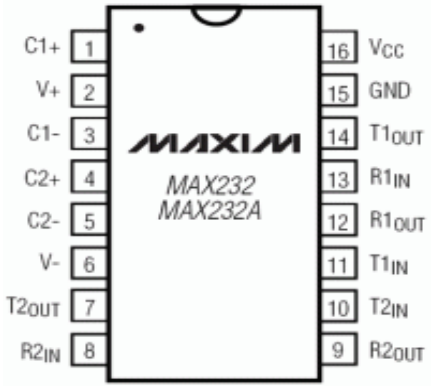


Figura 2. 15: MAX232[33]

Para que el **MAX232** funcione correctamente debemos realizar la conexión mostrada en la figura 2.16 en donde el valor de los condensadores es de 10 UF.

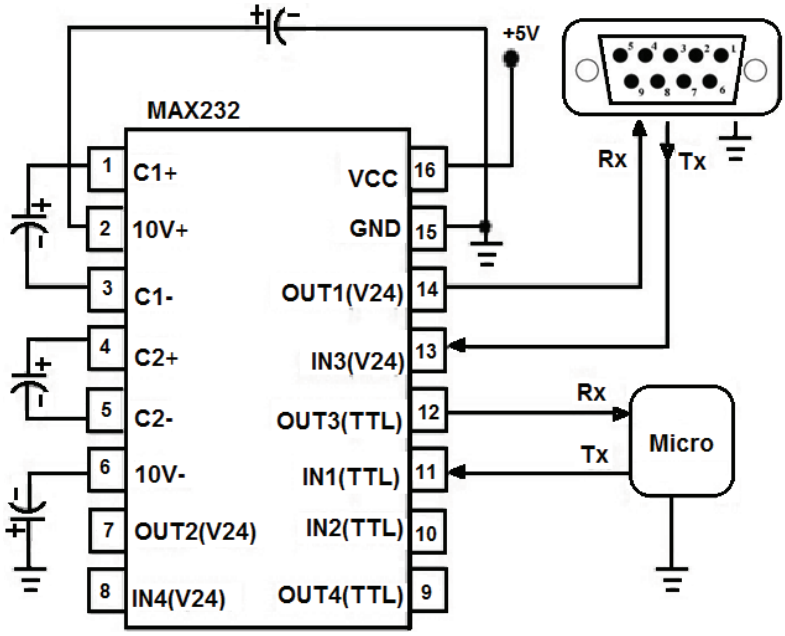


Figura 2.16: Diagrama de Conexiones el MAX232

---

[32] http://robots-argentina.com.ar/Comunicacion_max232.htm

[33] http://www.ucontrol.com.ar/wiki/index.php?title=MAX232

2.1.4.2.1 Distribuciones de pines del MAX 232

La figura 2.17 nos muestra la distribución de sus pines y en la tabla 2.6 muestra la función de cada uno de los pines de este circuito integrado:

Tabla 2.6: Funcionamiento de las Pines del MAX232

PIN	FUNCIONAMIENTO DE LOS PINES
$C1_+$	Conexión positiva del condensador C1 del doblador, voltaje de +5V a +10V.
$C1_-$	Conexión negativa del condensador C1 del doblador, voltaje de +5V a +10V.
$C2_+$	Conexión positiva del condensador C2 del inversor, voltaje de +10V a -10V.
$C2_-$	Conexión negativa del condensador C2 del inversor, voltaje de +10V a -10V.
V-	Conexión de salida del voltaje de -10V.
V+	Conexión de salida del voltaje de +10V.
$T1_{in}$, $T2_{in}$,$R1_{out}$,$R2_{out}$	Conexiones a niveles de voltaje de TTL o CMOS.
$T1_{out}$, $T2_{out}$,$R1_{in}$,$R2_{in}$	Conexiones a niveles de voltaje del protocolo RS-232.
VCC	Alimentación positiva del MAX232
GND	Alimentación negativa del MAX232

La figura 2.17 presenta las conexiones realizadas entre el ATmega 164P el MAX232 y el conector DB9 que se comunicará con la computadora.

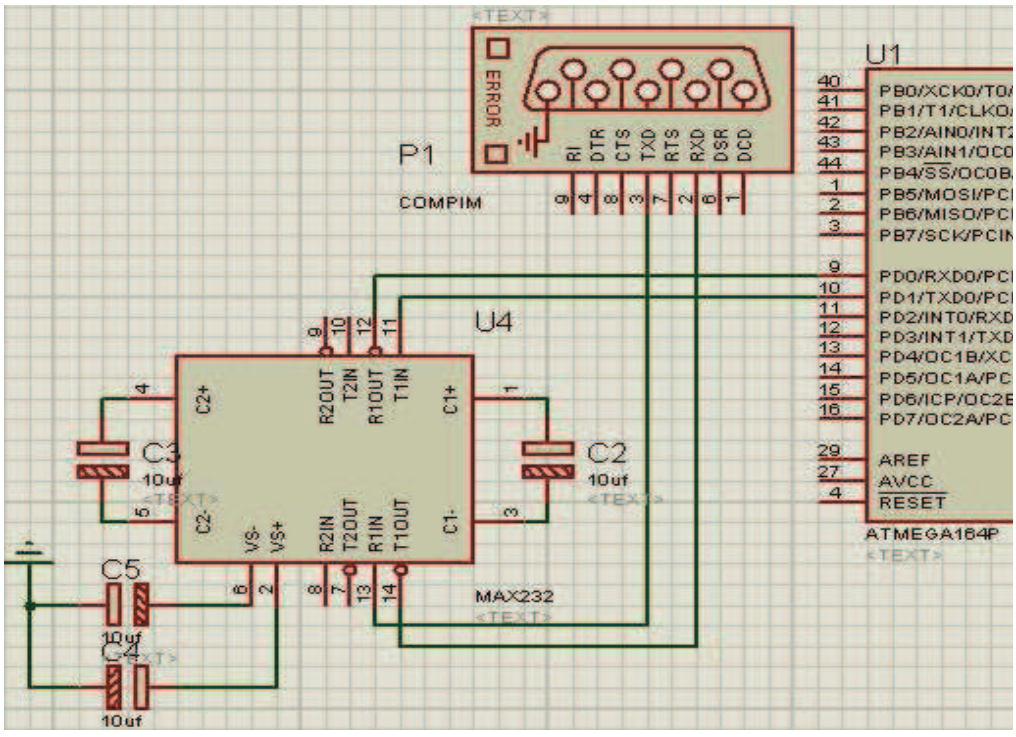


Figura 2.17: Diagrama de Conexiones del MAX232 y Convertidor USB/RS232

2.1.5 PROCESO DE VISUALIZACIÓN

En este proceso se realiza la interfaz que interactúa con el usuario, dicha interfaz, está realizada en Visual Basic 10.0[34] y contiene el programa que se comunica con el microcontrolador utilizando un puerto COM virtual, que es instalado por el convertidor de USB a RS232.

La figura 2.18 presenta los controles del programa que le permiten al usuario realizar las siguientes opciones:

1. Conectarse y desconectarse con el Puerto COM de la computadora.
2. Igualar la hora y la fecha del integrado DS1307.
3. Determinar cuál es la puerta que controla el prototipo de control.
4. Registrar un nuevo usuario.
5. Visualizar la nómina de usuarios registrados.
6. Actualizar los cambios en la memoria cuando se adiciona o elimina un nuevo usuario.
7. Borrar un solo usuario registrado.
8. Inicializar la tabla que contiene el registro de los usuarios que ingresaron y borrar la lista de todos los usuarios registrados en la memoria.
9. Leer el registro de los usuarios que han ingresado.

---

[34] Ver Anexo 8

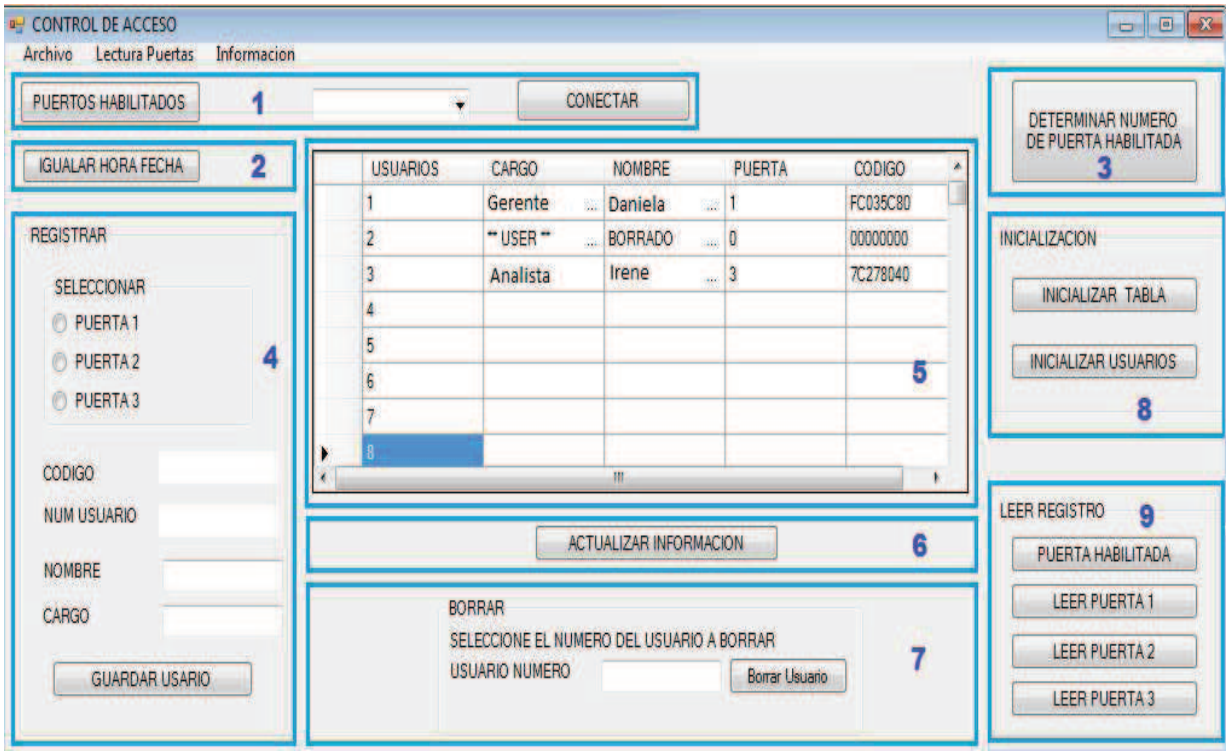


Figura 2.18: Interfaz de Usuario VB 10.0

La información guardada en la memoria 24IC256 es leída y presentada en con el formato que se muestra en la figura 2.19.

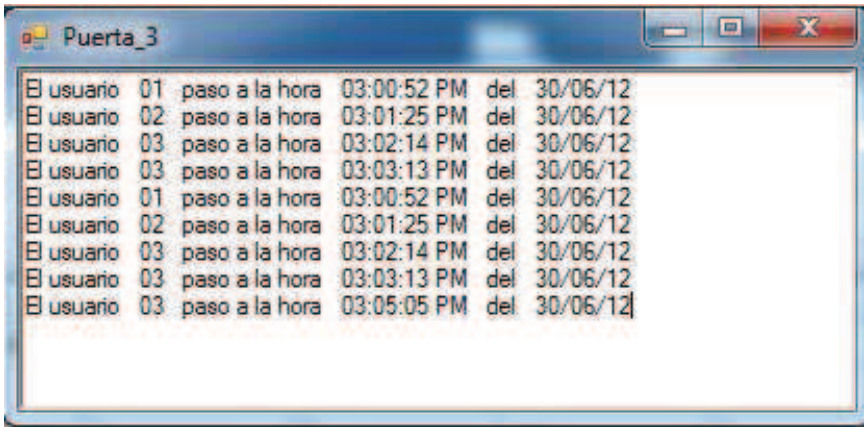


Figura 2.19: Presentación de Datos Leídos.

2.2 IMPLANTACIÓN DEL SISTEMA

En la implantación del sistema se presenta los diagramas del proceso que permitió realizar el prototipo del control de acceso.

2.2.1 IMPLANTACIÓN DEL SISTEMA PARA REALIZACIÓN DEL CIRCUITO DEL MÓDULO ID-12.

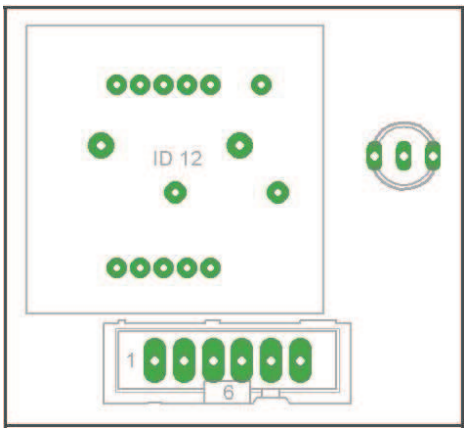


Figura 2.20: Diagrama de Elementos

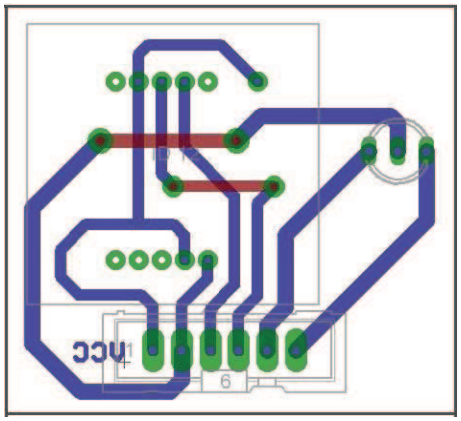


Figura 2.21: Diagrama de Conexiones del Módulo ID-12

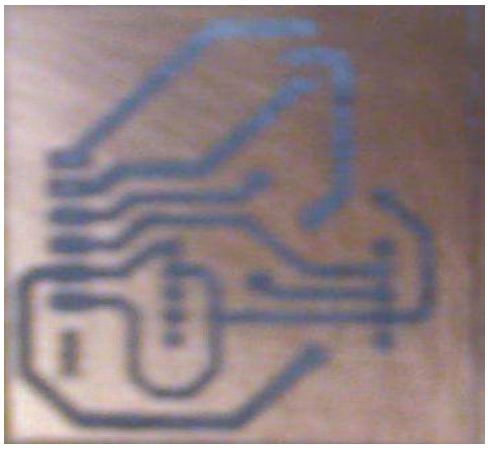


Figura 2.22: Circuito Planchado

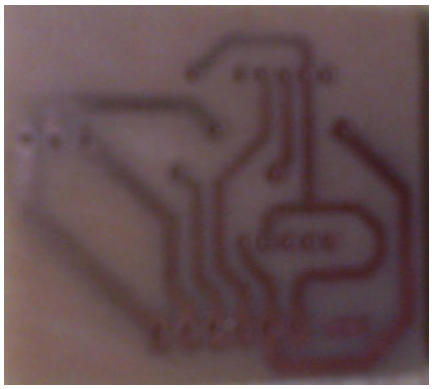


Figura 2.23: Circuito Quemado y Perforado

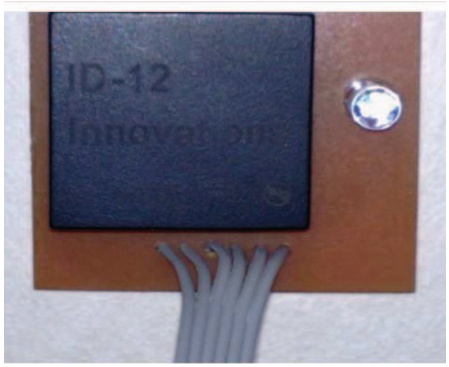


Figura 2.24: Circuito del Módulo ID-12

2.3 IMPLANTACIÓN DEL SISTEMA PARA REALIZACIÓN DEL CIRCUITO DE CONTROL.

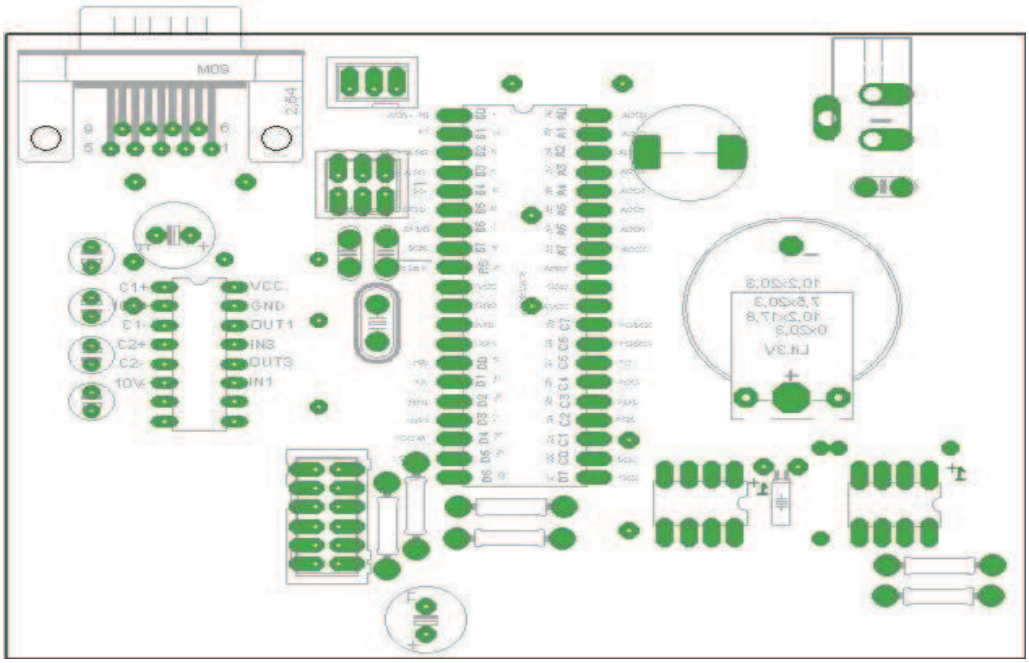


Figura 2.25: Diagrama de Elementos

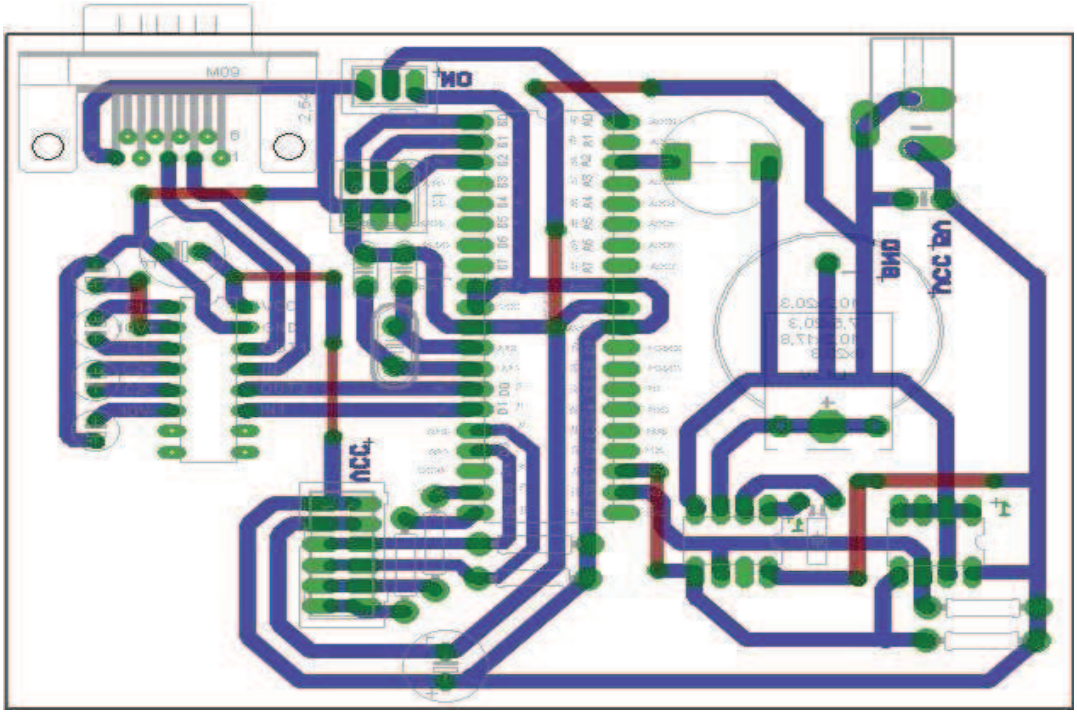


Figura 2.26: Diagrama de Conexiones del Módulo ID-12

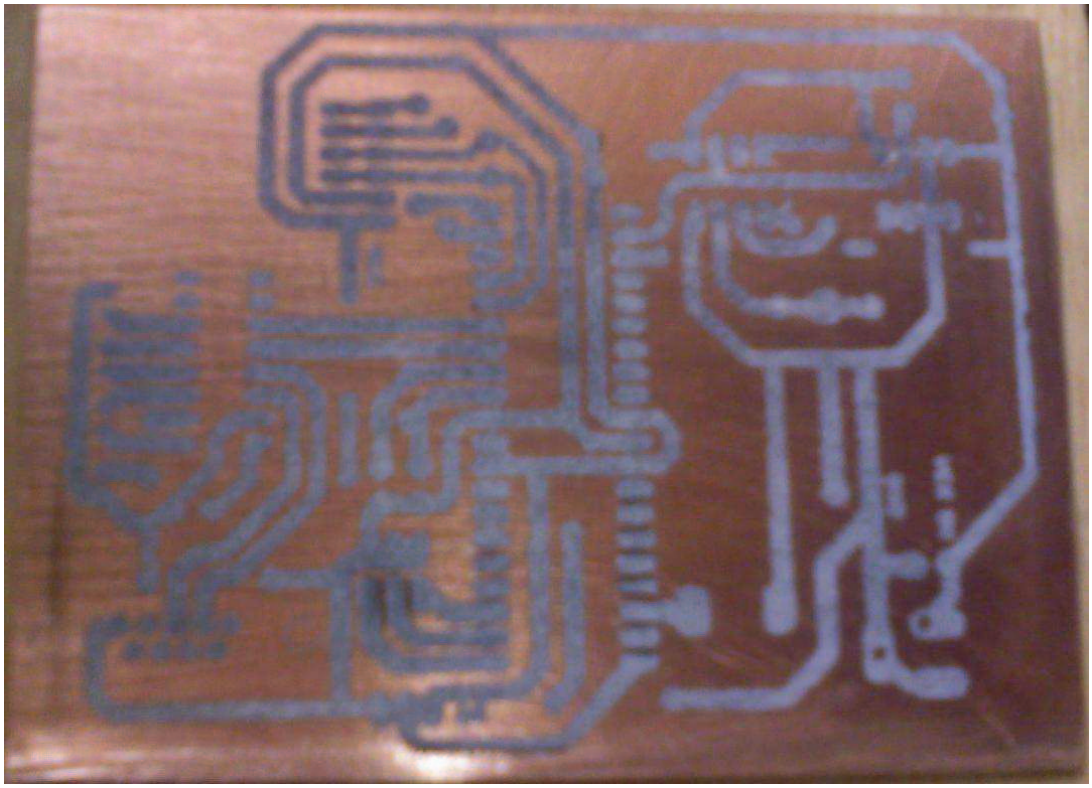


Figura 2.27: Circuito Planchado

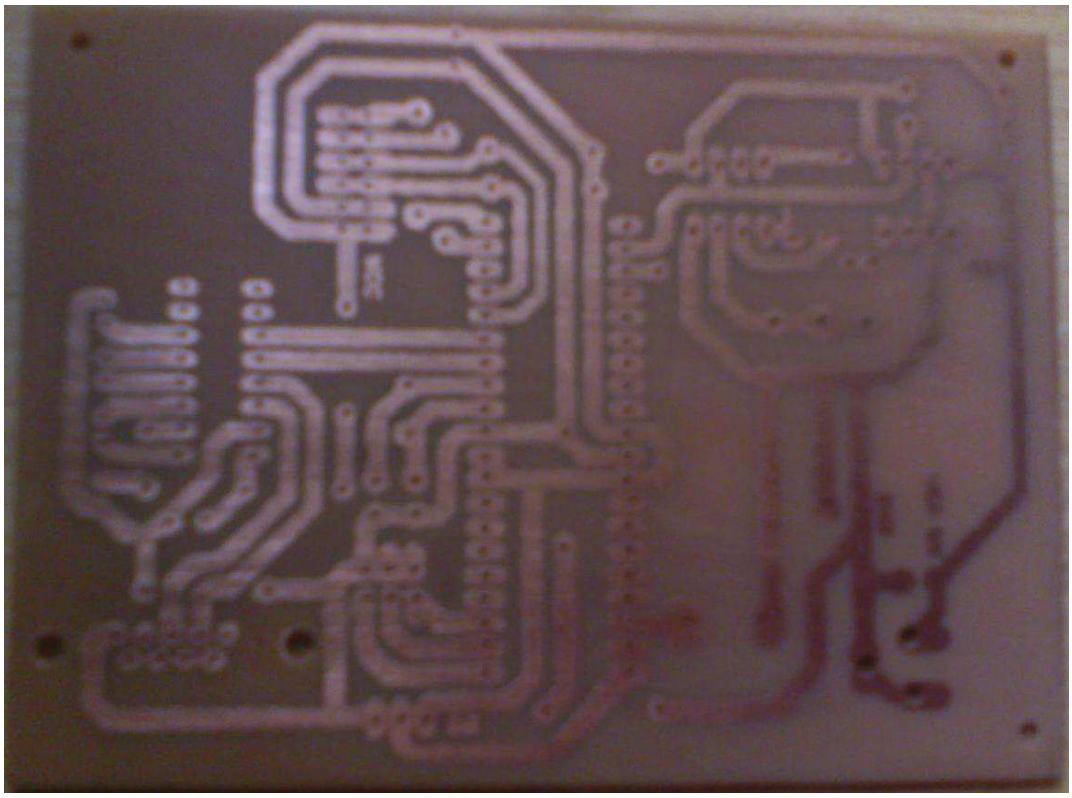


Figura 2.28: Circuito Quemado y Perforado

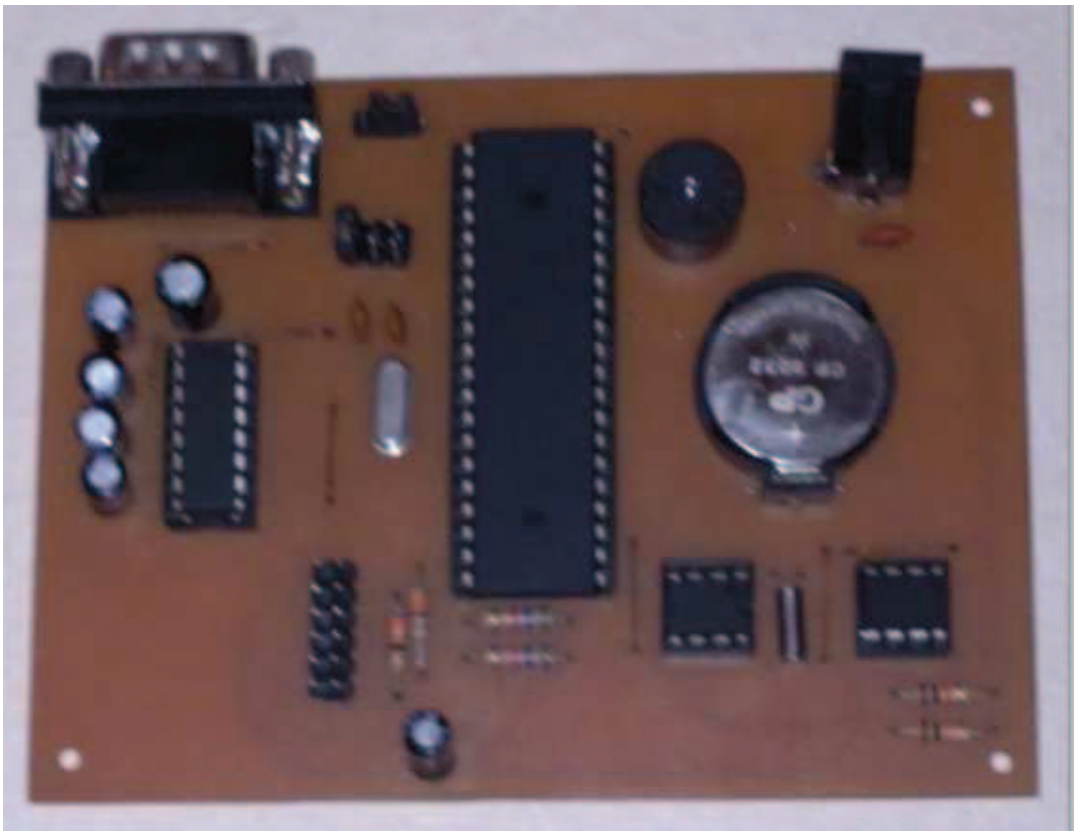


Figura 2.29: Circuito de Control

2.3.1 IMPLANTACIÓN DEL PROTOTIPO DE CONTROL DE ACCESO

La figura 2.30 presenta el diagrama de conexiones del prototipo de control de acceso.

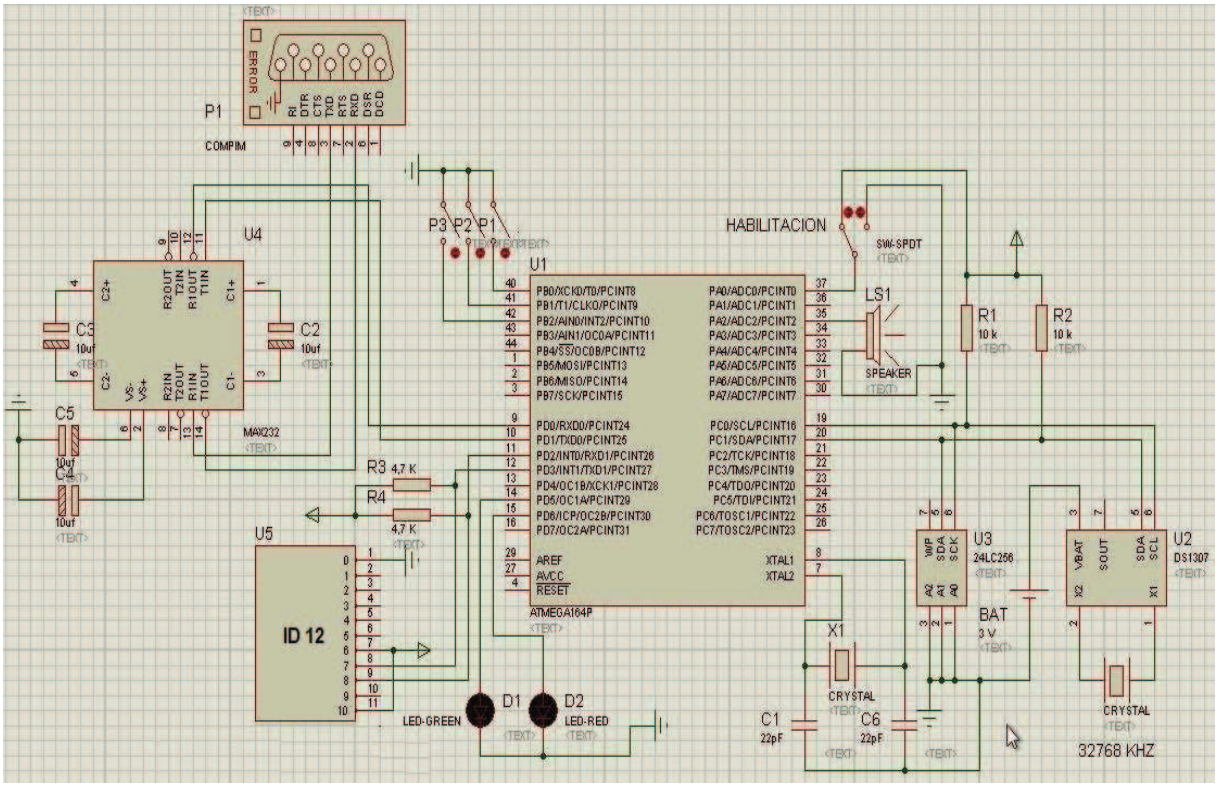


Figura 2.30: Prototipo de Control de Acceso

A continuación se presentan la figura 2.31 del prototipo implementado.

Figura 2.31: Prototipo de Control de Acceso

Figura 2.32: Circuito Implementado en la Maqueta.

2.4 DESCRIPCIÓN DEL SOFTWARE

Los programas que controlan al prototipo de control y registro de personal están realizados en BASCOM AVR y Visual Basic 10.0.

2.4.1 PROGRAMA PARA CONTROLAR EL ATMEGA 164P[35]

El software elegido para realizar el programa para el microcontrolador ATmega164P es el BASCOM AVR, ya que su lenguaje de programación es Basic y para aprender a utilizarlo, se encuentra en Internet gran cantidad de información.

---

[35] Ver Anexo 5

En BASCOM AVR se desarrolla todo el programa que permite que el microcontrolador determine cuál es el proceso que debe ejecutar, según el tipo de información que detecte.

El programa del microcontrolador está conformado de dos partes principales:

La primera parte se encarga de detectar cual fue el código ID de la tarjeta que envió el módulo ID-12 para determinar si el código detectado pertenece a un usuario registrado y en caso de no estarlo enviar este código hacia la computadora.

La segunda parte se encarga de detectar mediante la comunicación serial cual fue la cadena de datos recibida para determinar la acción a realizar.

2.4.1.1 Detección del Código de un Usuario.

El diagrama[36] indicado en el anexo 6, nos indica el proceso que se realiza para detectar si un usuario se encuentra o no registrado en la memoria.

El programa empieza declarando las variables e inicializando los valores de las mismas, a continuación cuando el microcontrolador ha detectado que en uno de sus pines de interrupción, INT0 (pin16) e INT1 (pin17), un cambio de estado a un valor lógico bajo, el microcontrolador determina que el módulo ID-12 está enviando un código de alguna tarjeta, seguido el microcontrolador guarda los datos recibidos en un string de datos que será comparado con el string guardado en la memoria anteriormente para determinar si el usuario está registrado y dependiendo de esto permitir que acceda a una área de trabajo determinada guardando en la memoria 24LC256 la hora y fecha a la que atravesó la puerta y emitiendo una señal auditiva de aviso mediante una chicharra.

En caso de que el usuario no está registrado, se determina si la computadora esta conectada al circuito de control, mediante el pin 40 (estado lógico alto) para

---

[36] Ver Anexo 6

proceder a enviar el ID de la tarjeta hacia la computadora, en caso contrario solo mantiene la puerta cerrada.

2.4.1.2 Detección de Datos Seriales

El diagrama[37] presentado en el anexo 7 presenta el proceso realizado para determinar que acción se tiene que realizar, dependiendo de la cadena de datos que detecte el microcontrolador.

Para detectar un dato se configuró la interrupción serial mediante la cual, cada vez que ingrese un dato, el mismo se guarda y ordena en una string, para que posteriormente el programa determine la acción a realizar.

El diagrama inicia declarando las variables e inicializando las mismas, para luego ingresar al programa principal donde primero determina y verifica si el módulo ID-12 detectó un nuevo usuario.

Posteriormente en el programa determina si alguna cadena de datos ingresó al microcontrolador verifica que los datos hayan llegado en el formato correcto, lo cual es realizado mediante la búsqueda de unas letras de ayuda que se adjuntan a la información que envió la computadora. Si el formato no es el correcto el programa envía hacia la computadora la cadena de datos “EEE” que representa error y la computadora deberá volver a enviar los datos, caso contrario el programa empieza a realizar las siguientes verificaciones:

1. Si encuentra “CX” en la cadena de datos recibida determina que el usuario desea conocer que puerta está controlando el microcontrolador y procede a transmitir “PC1”, “PC2” y “PC3”, según la puerta que controle.
2. Si encuentra “HF” en el inicio de la cadena de datos recibida, determina que los datos siguientes permitirán igualar la hora y la fecha del integrado DS1307, así que el programa procede a decodificar los datos e igualar la hora y la fecha para posteriormente transmitir “HGHG”, que es una cadena de

---

[37] Ver Anexo 7

datos que permitirá al programa desarrollado en Visual Basic informar al usuario que la hora y fecha se guardó con éxito.

3. Si encuentra “US” en el inicio de la cadena de datos recibida determina que los datos siguientes contienen la información para registrar a un nuevo usuario, así que el programa procede a decodificar los datos y guardarlos en la memoria 24LC256 para posteriormente transmitir “URUR”, que significa usuario registrado.
4. Si encuentra “SV” en el inicio de la cadena de datos recibida determina que los datos siguientes contienen la información para actualizar los datos de los usuarios registrados, así que el programa procede a decodificar los datos y guardarlos en la memoria 24LC256 para posteriormente transmitir “UGUG”, que significa usuario guardado.
5. Si se encuentra “US” al inicio de la cadena de datos es posible que se desee eliminar un usuario y para lo cual toda la información que contiene el código del usuario y la puerta que abrirá está llena de ceros “00”, así que el programa procede a guardar estos datos en la posición del usuario borrado, cuya información está incluida en la cadena de datos recibida, para posteriormente transmitir “UBUB”, que significa usuario borrado.
6. Si encuentra “BI” en la cadena de datos recibida determina que el usuario desea, borrar la información de los usuarios que han atravesado por la puerta para posteriormente transmitir “RBRB” que significa registro borrado.
7. Si encuentra “DR” en la cadena de datos recibida determina que el usuario desea, borrar la información de los usuarios y sus registros para posteriormente transmitir “RERE” que significa registros eliminados.
8. Si encuentra “ID” en la cadena de datos recibida, determina que el usuario desea conocer el número de usuarios que atravesaron la puerta y además cual es la puerta que está controlando el microcontrolador.
9. Si encuentra “LT” en la cadena de datos recibida por el microcontrolador, este determina que lo que se desea es leer todo el registro de los usuarios que atravesaron por una de las puertas y comienza a enviar la información de usuarios de dicha puerta dependiendo al computador, una vez que esta recibe la información la almacena el registro en un archivo

.txt para luego ser visualizado gracias a la interfaz realizada en Visual Basic.

2.4.2 PROGRAMA DE VISUAL BASIC 10.0[38]

La figura 2.33 presenta el entorno que se comunica con el microcontrolador y mediante el cual el usuario realizará cambios en el microcontrolador.

Al iniciar el programa se selecciona el puerto COM que se crea al instalar el convertidor USB a RS232, mediante el cual se transmitirá y recibirá la información para comunicarse con el microcontrolador, realizado esto se habilitan todos los controles de la interfaz del usuario.

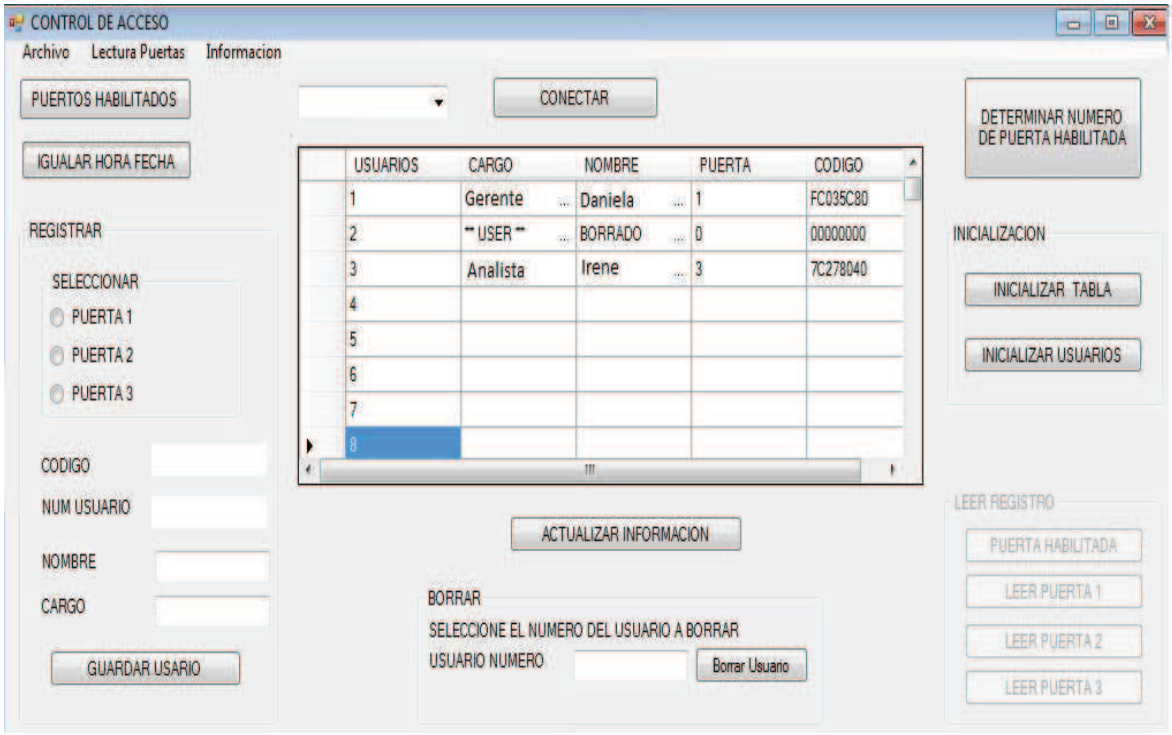


Figura 2.33: Entorno de Usuario

En esta interfaz el usuario realiza una acción cuando se da click sobre un botón para seleccionar lo que se desea ejecutar, el control que permite conectarse con el puerto COM de la computadora es el Puerto MSCom donde se selecciona la

[38] Ver Anexo 8

velocidad de transmisión y el tamaño en bits del dato a transmitir, mostrado en la figura 2.34.

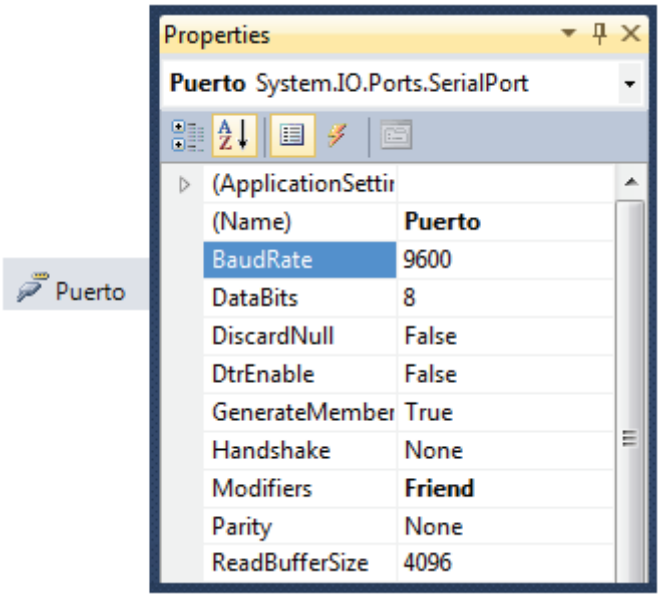


Figura 2.34: Control MSCom

También el programa posee 3 Timers que se encargan de verificar si ha ingresado algún dato en el buffer de entrada, leer los datos del personal que ingresa a un área y actualizar la información en la memoria 24LC256. Mediante los timers al detectar el tipo de información que recibió, se determina cual será el mensaje que se presentará al usuario.

2.4.2.1 Igualar el Horario

El anexo 9 presenta el diagrama[39] utilizado para igualar la hora y la fecha.

Para igualar el horario se da click sobre el botón mostrado en la figura 2.35 el cual lee la fecha y la hora de la computadora y ordena esta información en una cadena de datos mostrado en la tabla 2.7, en donde HF y FIN son caracteres de ayuda que le permiten al microcontrolador determinar la acción a realizar con esta información.

[39] Ver Anexo 9

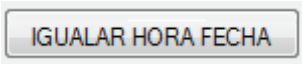


Figura 2.35: Botón para Horario

Tabla 2.7: Cadena de Datos para el Horario

H	F	09	55	A/P	24	11	23	FIN
		HORA		A	FECHA			
		hora	min	P	día	mes	año	

Posteriormente el programa espera recibir la cadena de caracteres HGHG lo cual le permite conocer que la hora se guardó exitosamente con lo cual desplegará el mensaje que se ve en la figura 2.36.

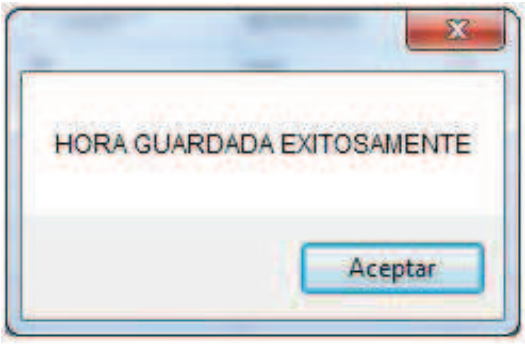


Figura 2.36: Mensaje de Operación Exitosa

Caso contrario desplegará el mensaje de error presentado en la figura 2.37, con lo cual el usuario debe volver a enviar el dato del horario.

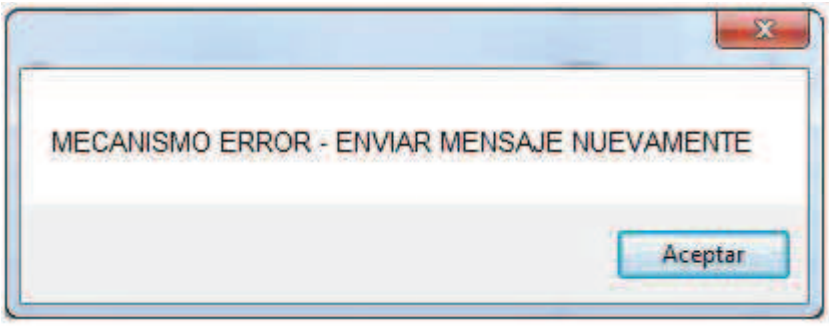


Figura 2.37: Mensaje de Error

2.4.2.2 Determinar Puerta Habilitada

El anexo 10 presenta el diagrama[40] utilizado para determinar la puerta habilitada.

Dando click sobre el botón mostrado en la figura 2.38, se envía la cadena de datos “CX CX CX”, y espera la respuesta PC1, PC2 y PC3 (Puerta Conectada 3) según la puerta que controle, mostrando posteriormente el mensaje de la figura 2.39, con esto se puede determinar, si es posible establecer un intercambio de datos entre el microcontrolador y la computadora.

Figura 2.38: Botón para Determinar Puerta Habilitada

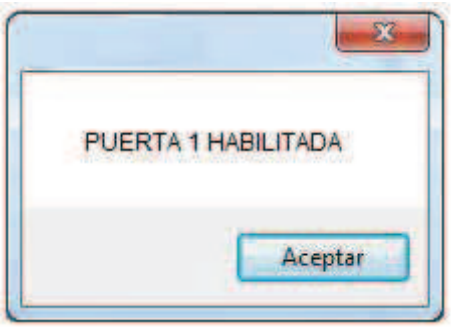


Figura 2.39: Mensaje de Puerta Habilitada

2.4.2.3 Registrar un Nuevo Usuario

El anexo 11, presenta el diagrama[41] para registrar un nuevo usuario.

Para registrar un nuevo usuario se espera que el microcontrolador envíe el código de un usuario no registrado con el formato mostrado en la tabla 2.8.

Tabla 2.8: Código de Usuario Detectado

NEW	F C 2 3 5 C 8 0	FIN
	CÓDIGO TAG	

[40] Ver Anexo 10

[41] Ver Anexo 11

Este código es detectado por el Timer1 para que el programa busque los caracteres “NEW” y “FIN”, para determinar si el formato del código es correcto, caso contrario presenta el mensaje de error mostrado en la figura 2.40, con lo cual se debe volver a pasar el tag por el módulo ID-12 para que el microcontrolador envíe el código del usuario no encontrado.

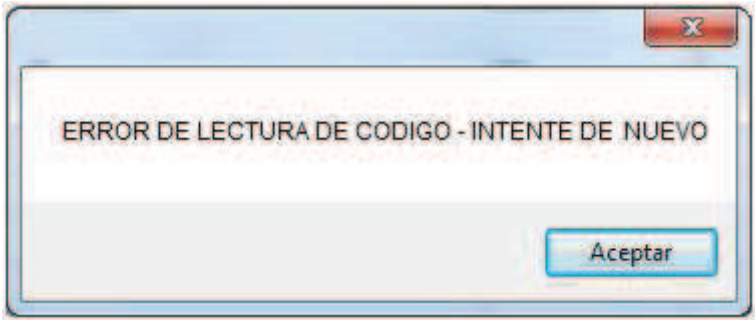


Figura 2.40: Mensaje de Error

Al ser el formato del código el correcto, el Timer1 carga la información en la figura 2.41 donde se debe seleccionar la puerta a la que tendrá acceso el nuevo usuario, escribir el nombre y su cargo, para proceder a enviar la información del usuario a guardar.

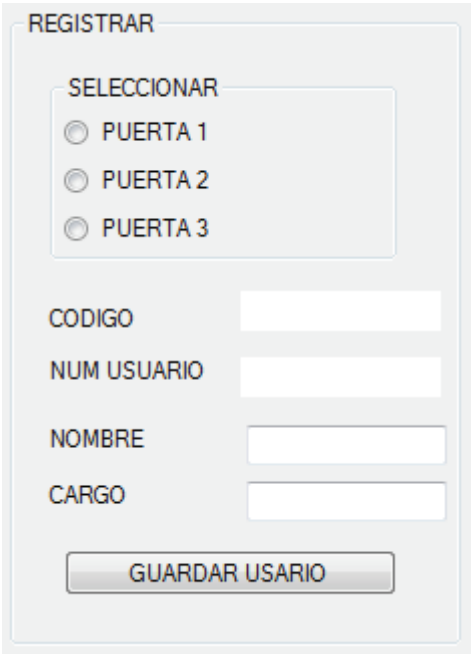


Figura 2.41: Controles para Guardar un Usuario

El formato enviado por la computadora es presentado en la tabla 2.9, este formato contiene:

1. Número total de usuarios registrados.
2. Dirección donde se guardará el código del Tag.
3. Número de la puerta a la cual tendrá el acceso el usuario.
4. Código del Tag.

Tabla 2.9: Formato de Código del Usuario.

U	S	9	9	9	9	0	3	A	B	3	4	F	F	A	D	F	I	N
U	S	1		2		3		4								F	I	N

El código es recibido por el microcontrolador y guardado en la memoria 24LC256 para seguidamente enviar la cadena de caracteres “EEE” que representa error de código recibido y desde la computadora se debe volver a enviar el código del usuario a guardar, o por el contrario envía la cadena de caracteres “URUR”, que significa usuario registrado con lo cual se presenta al usuario el mensaje de la figura 2.42 que representa una operación exitosa.

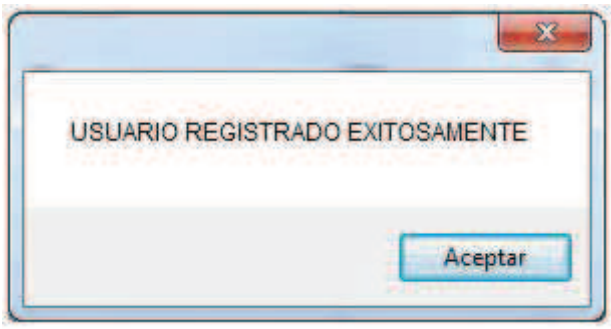


Figura 2.42: Mensaje de Operación Exitosa.

2.4.2.4 Actualizar Información

El botón presentado en la figura 2.43 permite que se guarde la información de todos los usuarios y esta opción se utiliza para guardar el registro total cuando se ha agregado o borrado algún usuario.

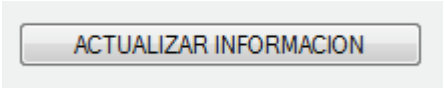


Figura 2.43: Botón Actualizar Información.

El anexo 12 presenta el diagrama[42] que permite guardar todos los usuarios.

En este diagrama se inicia presionando el botón de actualizar información, posteriormente el programa habilita el Timer 3 y se inicia el proceso de envío de la información de todos los usuarios con el formato mostrado en la tabla 2.10 que al ser detectado por el microcontrolador empieza a guardar los datos en la memoria 24LC256 hasta completar el envío de la información del último usuario y posteriormente presentar el mensaje de éxito presentado en la figura 2.44.

Tabla 2.10: Formato de Código del Usuario.

U	S	9	9	9	9	0	3	A	B	3	4	F	F	A	D	F	I	N
U	S	1		2		3		4								F	I	N

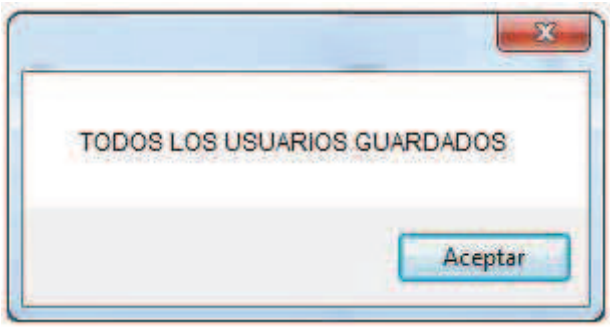


Figura 2.44: Mensaje de Operación Exitosa.

2.4.2.5 Borrar Información

Para borrar un usuario se selecciona el número del usuario a borrar y se da click sobre Borrar Usuario figura 2.45.

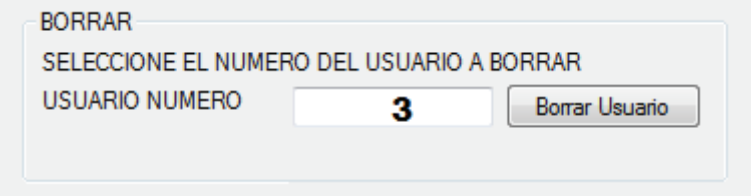


Figura 2.45: Borrar un Usuario.

[42] Ver Anexo 12

El proceso para borrar un usuario es similar al de registrar un nuevo usuario, solo que en este caso en la cadena de caracteres enviados hacia el microcontrolador, la información que contiene el número de puerta y el código del tag es llenado con ceros (tabla 2.11), así cuando el módulo ID-12 detecte a la tarjeta del usuario borrado simplemente no permitirá que ingrese porque no se encuentra en los registros de la memoria 24LC256.

Tabla 2.11: Formato de Código del Usuario a Borrar

U	S	9	9	9	9	0	0	0	0	0	0	0	0	0	0	F	I	N
U	S	1		2		3		4								F	I	N

Posteriormente si la operación es exitosa se presenta en la pantalla el mensaje de la figura 2.46 o de lo contrario se solicita al usuario que vuelva a enviar la información.

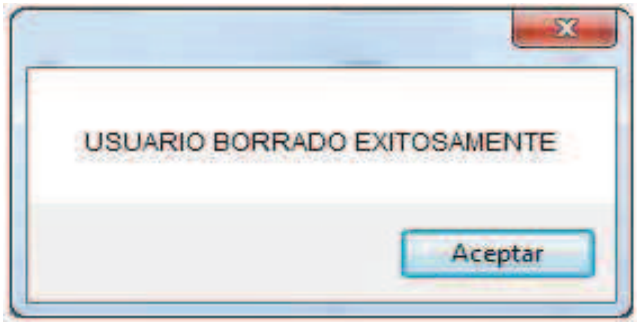


Figura2.46: Mensaje de Operación Exitosa.

2.4.2.6 Iniciar Tabla

Este proceso permite que el registro de ingreso de los usuarios sea borrado, lo que permite que el microcontrolador guarde nuevamente otro registro de ingreso, previamente guardando el registro a ser borrado en la computadora, esto se realiza al presionar el botón de la figura 2.47.

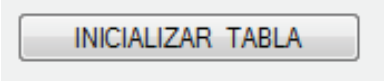


Figura 2.47: Botón para Borrar Registro de Ingreso.

Al presionar este botón la computadora envía la cadena de caracteres **"BIBIBI",** que significa borrar ingreso y cuando la operación se ha realizado con éxito en microcontrolador envía **"RBRBRB",** que significa registro borrado y se despliega el mensaje presentado en la figura 2.48 en la pantalla o de lo contrario se solicita al usuario que vuelva a enviar la información.

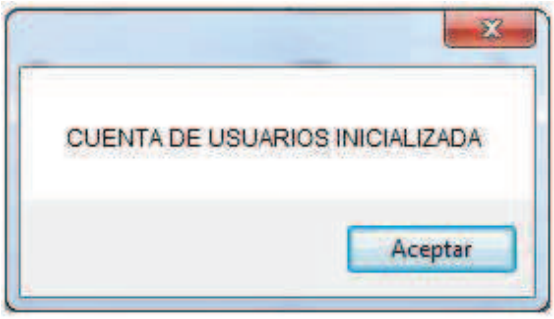


Figura 2.48: Mensaje de Operación Exitosa.

2.4.2.7 Iniciar Usuarios

Este proceso permite que el registro de los usuarios y su ingreso sean borrados, operación que se realiza cuando se desea ingresar una nueva nómina total de usuarios o al iniciar el proceso de registro de los usuarios, esto se realiza al presionar el botón de la figura 2.49.

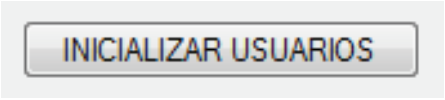


Figura 2.49: Botón para Borrar Registro Total.

Al presionar este botón la computadora envía la cadena de caracteres **"DRDRDR"**, que significa DELETE registro y cuando la operación se ha realizado con éxito el microcontrolador envía **"RERERE",** que significa registro eliminado y se despliega el mensaje presentado en la figura 2.50 en la pantalla o de lo contrario se solicita al usuario que vuelva a enviar la información.

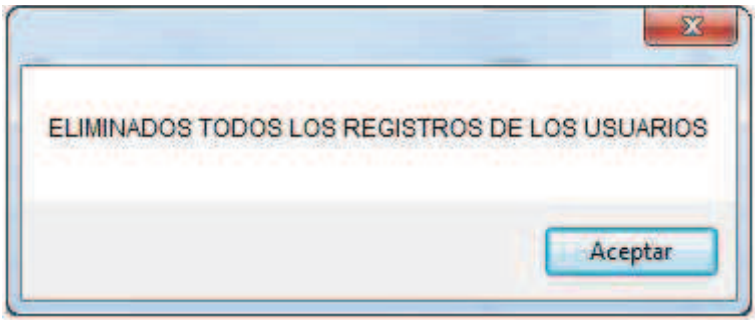


Figura 2.50: Mensaje de Operación Exitosa.

2.4.2.8 Leer Registro de Ingreso

En el anexo 13 se presenta el diagrama[43] de la operación realizada para leer los registros almacenados en los .txt de los usuarios que ingresaron a un área de trabajo determinada.

Para leer el registro de ingreso primero se da click en el botón puerta habilitada presentado en la figura 2.51 el cual envía hacia el microcontrolador la cadena de datos **"IDID ID",** pidiendo la identificación de la puerta que controla y a su vez el número total de usuarios que ingresaron por dicha puerta.

Figura 2.51: Controles para Leer el Registro de Ingreso.

Cuando la petición es exitosa el Timer1 se deshabilita, activa el botón de la puerta que controla el microcontrolador y habilita al Timer 2, el cual procederá a leer el registro de ingreso del personal y escribirlo en el archivo .txt creado para guardar el registro de cada puerta, cuando se presione el botón Leer Puerta 1,2 o 3 según corresponda.

---

[43] Ver Anexo 13

Finalmente se observa el mensaje de lectura mostrada en la figura 2.52 y se procede a visualizar laventana con la información del registro como se presenta en la figura 2.53.

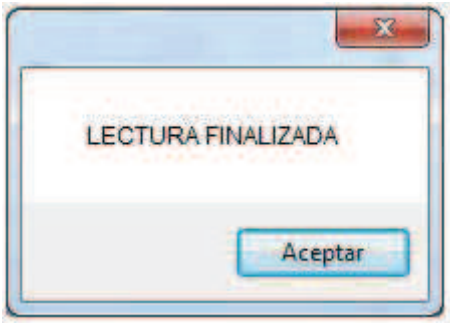


Figura 2.52: Mensaje de Lectura Finalizada

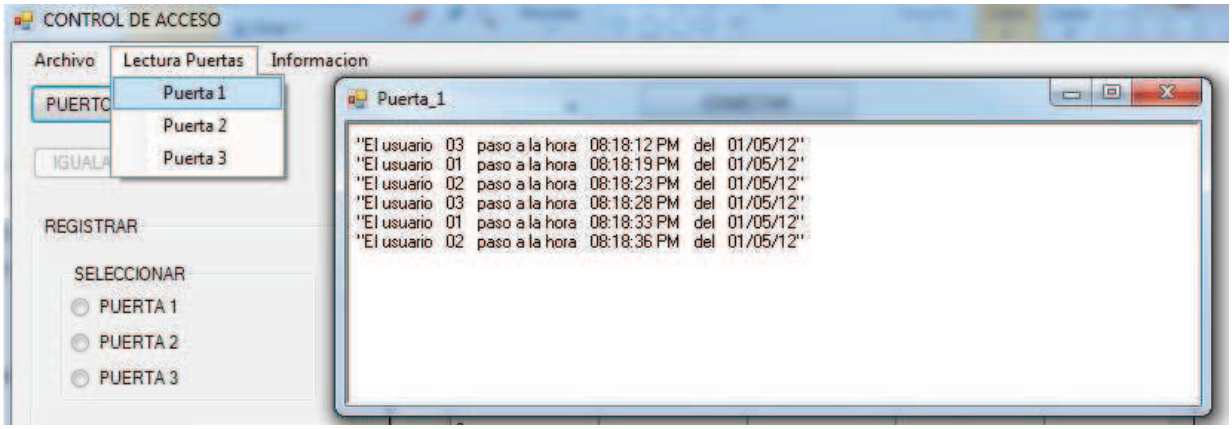


Figura 2.53: Presentación del Registro Leído.

2.5 PRUEBAS Y RESULTADOS

El prototipo de control y registro personal, posee una chicharra que se activa cuando el registro de la persona, es encontrado en la memoria 24LC256, esto es de gran ayuda ya que la señal auditiva permite que el usuario conozca que puede ingresar al área de trabajo.

En la baquelita donde se encuentra el módulo ID-12 se adicionó un led bicolor el cual cambia de rojo a verde cuando una persona puede ingresar a un área de trabajo, esto permite que el usuario conozca que puede ingresar al área de trabajo en caso de no escuchar la señal emitida por la chicharra.

En la memoria 24LC256 es almacenada toda la información de los mensajes escritos de manera ordenada y en direcciones específicas, que son previamente calculadas mediante el programa, la verificación de la escritura de los datos en sus direcciones correspondientes, se realizó escribiendo varios mensajes con diferentes características en la memoria y verificando la posición de sus datos, mediante la utilización de un LCD utilizado solo para pruebas.

El programa realizado en BASCOM AVR para el microcontrolador permite que cuando se detecta que un usuario no está registrado, el microcontrolador transmita el código del tag encontrado, esto se comprobó observando en el programa desarrollado en Visual Basic, en el cual es posible observar el código del tag encontrado.

El programa realizado en Visual Basic permite borrar un usuario y se comprobó que esto se realizaba de manera correcta borrando un usuario registrado y luego pasando el tag correspondiente al código borrado, comprobando que efectivamente su registro no era encontrado.

En el programa realizado en Visual Basic tiene la opción para guardar todos los datos de registros o cambios realizados, esto se comprobó utilizando un circuito del prototipo de control para realizar cambios y luego grabando estos cambios en los otros dos circuitos, se verificó que los cambios se habían realizado ya que los circuitos solo permitían que ingresen los usuarios según los cambios guardados y en las puertas determinadas.

El prototipo de control y registro de personal, posee un pin que permite deshabilitar la opción para enviar la cadena de datos del código del tag encontrado, en caso de que un usuario no esté registrado, se comprobó que esta opción funciona de manera correcta, con lo cual un usuario puede ser registrado solo por el personal que conozca el funcionamiento del prototipo de control y cuando en realidad se requiera registrar un nuevo usuario.

CAPÍTULO III

CONCLUSIONES Y RECOMENDACIONES

3.1 CONCLUSIONES

- El lenguaje para programar el microcontrolador BASCOM AVR y el entorno de usuario desarrollado en Visual Basic 10.0, son lenguajes de programación similares, lo cual resulta de gran ayuda ya que las sentencias de programación utilizadas son muy parecidas.
- El programa creado en Visual Basic 10.0, fue desarrollado para que el usuario interactúe con el microcontrolador de una manera sencilla, con esto se logra que el usuario conozca que es lo que se debe realizar para registrar o eliminar un usuario y además los mensajes que el programa presenta, permiten conocer si una acción se realizó de manera exitosa o si es necesario volver a repetirla.
- El ATmega 164P es utilizado por poseer grandes ventajas, en su estructura interna (memoria flash, memoria EEPROM, memoria SRAM, contadores, líneas de entrada/salida, etc.), también porque actualmente puede ser encontrado en el mercado con facilidad.
- El prototipo desarrollado e implementado es una herramienta que permite controlar y registrar el acceso de personas a determinadas áreas de una institución y al conocer la hora de ingreso se puede tener un control de puntualidad del personal.
- Cuando se intento ingresar con un Tag o Tarjeta RFID que estaba trabajando a la misma frecuencia, marca o de características similares a las que se usaron en el prototipo, no fue posible debido a que el sistema permite el acceso siempre y cuando el número único o código de la tarjeta que es leído por el ID-12, datos personales del usuario estén guardados en la memoria del microcontrolador, caso contrario nos indica que se a encontrado un nuevo usuario, denegando así el ingreso.

3.2 RECOMENDACIONES

- Verificar que el voltaje de la fuente que alimentará al prototipo de control de acceso tenga 5 voltios, antes de poner en funcionamiento el prototipo y verificar que la polaridad se encuentre correcta.
- El prototipo está realizado para ser instalado cerca de la puerta que controlará, así que el lugar de su instalación debe tener una protección contra la humedad para asegurar el correcto funcionamiento del mismo.
- En caso de no existir comunicación entre el prototipo y el computador, primero revisar si las conexiones están bien aseguradas de modo que permitan la transmisión y recepción de la información.
- En caso de que la tarjeta de los usuarios no puedan ser detectados por el ID-12, se debe revisar primero si la conexión entre el ID-12 y el microcontrolador está bien realizada y verificar que no exista ningún cable roto.
- El uso de una base de datos permitirá remplazar el uso de archivos txt que fueron utilizados en este proyecto, así se podrá tener de manera centralizada y en una única estructura la información receptada por el circuito de radiofrecuencia.
- Dar continuidad al proyecto base de radiofrecuencia, de manera que se pueda construir los tres circuitos en uno solo, optimizando así recursos, administración y adquisición de este tipo de soluciones.

BIBLIOGRAFÍA

[19] [20] [21] Valencia R. (2008). Aplicaciones Electrónicas con Microcontroladores AVR. Ibarra: Graficolor

[22] Pazmiño D. Visual Basic

REFERENCIAS

[1] http://es.wikipedia.org/wiki/RFID

[2] http://www.libera.net/uploads/documents/whitepaper_rfid.pdf

[3]http://www.novenca.com/site/index.php?option=com_content&view=article&id=166&Itemid=123

[4] http://www.subtel.gob.cl/prontus_subtel/site/edic/base/port/inicio.html

[5] http://www.electronicamagnabit.com/tienda/kits-rf/155-lector-rfid-id-12.html

[6] http://www.electronicamagnabit.com/tienda/kits-rf/155-lector-rfid-id-12.html

[7] http://es.kioskea.net/contents/rfid/rfid-intro.php3

[8] http://es.wikipedia.org/wiki/RFID

[9] http://r-luis.xbot.es/pic1/pic01.html

[10] http://www.mikroe.com/eng/chapters/view

[11] http://perso.wanadoo.es/pictob/micropic.htm

[13] http://perso.wanadoo.es/pictob/microcr.htm

[14] http://picaxe.electronicasimple.com/2009/03/reloj-tiempo-real-ds1307.html

[16] http://electronicapractica.crearblog.com/2010/11/reloj-con-el-ds1307/#more-306

[18] http://robots-argentina.com.ar/Comunicacion_max232.htm

[23] http://picmania.garcia-cuervo.net/eagle_tutlbr_i_library.php

[25] http://www.ermicro.com/blog/?p=744

[26] http://axxon.com.ar/rob/Comunicacion_busI2C.htm

[27] http://axxon.com.ar/rob/Comunicacion_busI2C.htm

[28] http://www.ermicro.com/blog/?p=1239

[29] http://www.ermicro.com/blog/?p=1239

[30]http://www.trendnet.com/langsp/products/proddetail.asp?prod=150_TU-S9&cat=32

[31] http://www.iearobotics.com/proyectos/cuadernos/ct1/ct1.html

[32] http://robots-argentina.com.ar/Comunicacion_max232.htm

[33] http://www.ucontrol.com.ar/wiki/index.php?title=MAX232

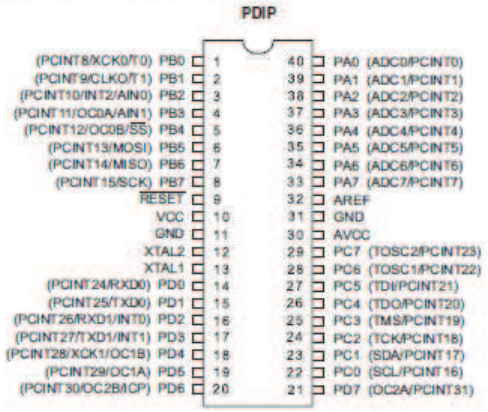
ANEXO 1: HOJA DE DATOS ATMEGA-164P

DATASHEET ATMEGA-164P

1. Pin Configurations

1.1 Pinout - PDIP/TQFP/VQFN/QFN/MLF

Figure 1-1. Pinout ATmega164P/324P/644P



Note: The large center pad underneath the VQFN/QFN/MLF package should be soldered to ground on the board to ensure good mechanical stability.

The ATmega164P/324P/644P provides the following features: 16K/32K/64K bytes of In-System Programmable Flash with Read-While-Write capabilities, 512B/1K/2K bytes EEPROM, 1K/2K/4K bytes SRAM, 32 general purpose I/O lines, 32 general purpose working registers, Real Time Counter (RTC), three flexible Timer/Counters with compare modes and PWM, 2 USARTs, a byte oriented 2-wire Serial Interface, a 8-channel, 10-bit ADC with optional differential input stage with programmable gain, programmable Watchdog Timer with Internal Oscillator, an SPI serial port, IEEE std. 1149.1 compliant JTAG test interface, also used for accessing the On-chip Debug system and programming and six software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next interrupt or Hardware Reset. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except Asynchronous Timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the Crystal/Resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low power consumption. In Extended Standby mode, both the main Oscillator and the Asynchronous Timer continue to run.

The device is manufactured using Atmel's high-density nonvolatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed in-system through an SPI serial interface, by a conventional nonvolatile memory programmer, or by an On-chip Boot program running on the AVR core. The boot program can use any interface to download the application program in the application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega164P/324P/644P is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATmega164P/324P/644P AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.

2.2 Comparison Between ATmega164P, ATmega324P and ATmega644P

Table 2-1. Differences between ATmega164P and ATmega644P

Device	Flash	EEPROM	RAM
ATmega164P	16 Kbyte	512 Bytes	1 Kbyte
ATmega324P	32 Kbyte	1 Kbyte	2 Kbyte
ATmega644P	64 Kbyte	2 Kbyte	4 Kbyte

2.3 Pin Descriptions

2.3.1 VCC

Digital supply voltage.

2.3.2 GND

Ground.

2.3.3 Port A (PA7:PA0)

Port A serves as analog inputs to the Analog-to-digital Converter.

Port A also serves as an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port A pins that are externally pulled low will source current if the pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port A also serves the functions of various special features of the ATmega164P/324P/644P as listed on page 80.

2.3.4 Port B (PB7:PB0)

Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B also serves the functions of various special features of the ATmega164P/324P/644P as listed on page 82.

2.3.5 Port C (PC7:PC0)

Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port C also serves the functions of the JTAG interface, along with special features of the ATmega164P/324P/644P as listed on page 85.

2.3.6 Port D (PD7:PD0)

Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port D also serves the functions of various special features of the ATmega164P/324P/644P as listed on page 87.

2.3.7 RESET

Reset input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in "System and Reset Characteristics" on page 331. Shorter pulses are not guaranteed to generate a reset.

2.3.8 XTAL1

Input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

2.3.9 XTAL2

Output from the inverting Oscillator amplifier.

2.3.10 AVCC

AVCC is the supply voltage pin for Port A and the Analog-to-digital Converter. It should be externally connected to V_{CC}, even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter.

2.3.11 AREF

This is the analog reference pin for the Analog-to-digital Converter.

ANEXO 2: HOJA DE DATOS DS 1307

DATASHEET DS 1307

DALLAS SEMICONDUCTOR

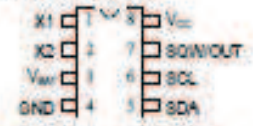
DS1307
64 x 8 Serial Real-Time Clock

www.maxim-ic.com

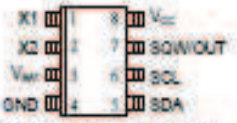
FEATURES

- Real-time clock (RTC) counts seconds, minutes, hours, date of the month, month, day of the week, and year with leap-year compensation valid up to 2100
- 56-byte, battery-backed, nonvolatile (NV) RAM for data storage
- Two-wire serial interface
- Programmable squarewave output signal
- Automatic power-fail detect and switch circuitry
- Consumes less than 500nA in battery backup mode with oscillator running
- Optional industrial temperature range: -40°C to +85°C
- Available in 8-pin DIP or SOIC
- Underwriters Laboratory (UL) recognized

PIN ASSIGNMENT



DS1307 8-Pin DIP (300-mil)



DS1307 8-Pin SOIC (150-mil)

ORDERING INFORMATION

DS1307	8-Pin DIP (300-mil)
DS1307Z	8-Pin SOIC (150-mil)
DS1307N	8-Pin DIP (Industrial)
DS1307ZN	8-Pin SOIC (Industrial)

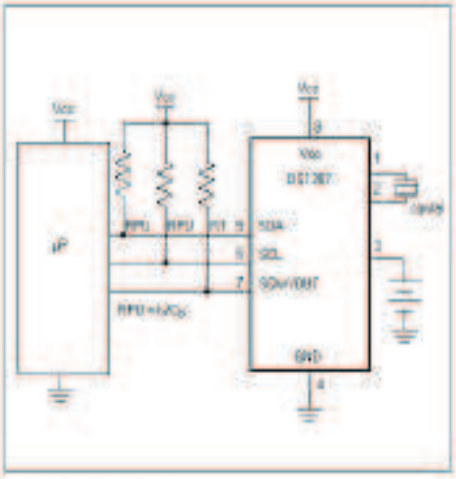
PIN DESCRIPTION

V_{CC}	- Primary Power Supply
X1, X2	- 32.768kHz Crystal Connection
V_{BAT}	- +3V Battery Input
GND	- Ground
SDA	- Serial Data
SCL	- Serial Clock
SQW/OUT	- Square Wave Output Driver

DESCRIPTION

The DS1307 Serial Real-Time Clock is a low-power, full binary-coded decimal (BCD) clock/calendar plus 56 bytes of NV SRAM. Address and data are transferred serially via a 2-wire, bi-directional bus. The clock/calendar provides seconds, minutes, hours, day, date, month, and year information. The end of the month date is automatically adjusted for months with fewer than 31 days, including corrections for leap year. The clock operates in either the 24-hour or 12-hour format with AM/PM indicator. The DS1307 has a built-in power sense circuit that detects power failures and automatically switches to the battery supply.

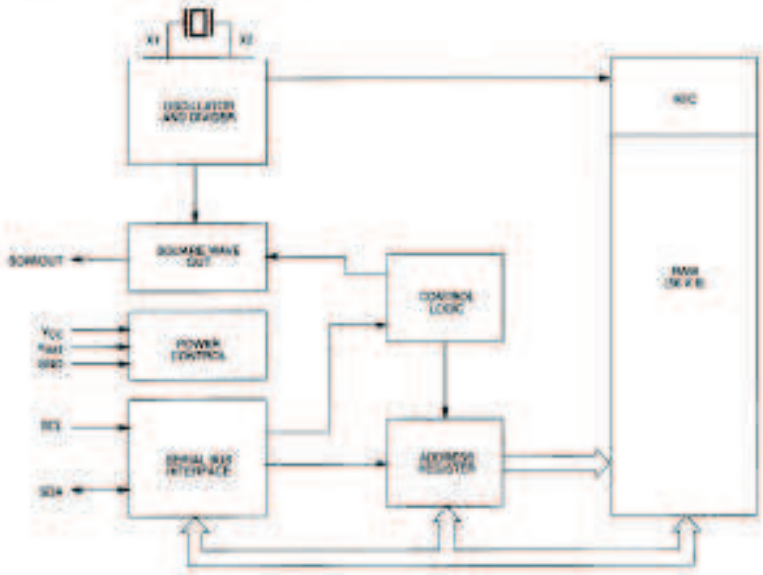
TYPICAL OPERATING CIRCUIT



OPERATION

The DS1307 operates as a slave device on the serial bus. Access is obtained by implementing a START condition and providing a device identification code followed by a register address. Subsequent registers can be accessed sequentially until a STOP condition is executed. When V_{CC} falls below 1.25 x V_{BAT} the device terminates an access in progress and resets the device address counter. Inputs to the device will not be recognized at this time to prevent erroneous data from being written to the device from an out of tolerance system. When V_{CC} falls below V_{BAT} the device switches into a low-current battery backup mode. Upon power-up, the device switches from battery to V_{CC} when V_{CC} is greater than V_{BAT} + 0.2V and recognizes inputs when V_{CC} is greater than 1.25 x V_{BAT}. The block diagram in Figure 1 shows the main elements of the serial RTC.

DS1307 BLOCK DIAGRAM Figure 1



SIGNAL DESCRIPTIONS

V_{CC}, GND – DC power is provided to the device on these pins. V_{CC} is the +5V input. When 5V is applied within normal limits, the device is fully accessible and data can be written and read. When a 3V battery is connected to the device and V_{CC} is below 1.25 x V_{BAT}, reads and writes are inhibited. However, the timekeeping function continues unaffected by the lower input voltage. As V_{CC} falls below V_{BAT} the RAM and timekeeper are switched over to the external power supply (nominal 3.0V DC) at V_{BAT}.

V_{BAT} – Battery input for any standard 3V lithium cell or other energy source. Battery voltage must be held between 2.0V and 3.5V for proper operation. The nominal write protect trip point voltage at which access to the RTC and user RAM is denied is set by the internal circuitry as 1.25 x V_{BAT} nominal. A lithium battery with 48mAhr or greater will back up the DS1307 for more than 10 years in the absence of power at 25°C. UL recognized to ensure against reverse charging current when used in conjunction with a lithium battery.

See "Conditions of Acceptability" at http://www.maxim-ic.com/TechSupport/QA/ntrl.htm

SCL (Serial Clock Input) – SCL is used to synchronize data movement on the serial interface.

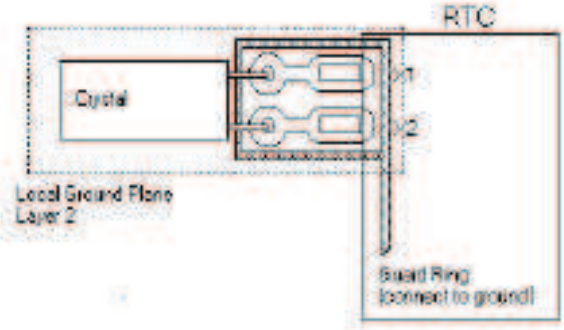
SDA (Serial Data Input/Output) – SDA is the input/output pin for the 2-wire serial interface. The SDA pin is open drain which requires an external pullup resistor.

SQW/OUT (Square Wave/Output Driver) – When enabled, the SQWE bit set to 1, the SQW/OUT pin outputs one of four square wave frequencies (1Hz, 4kHz, 8kHz, 32kHz). The SQW/OUT pin is open drain and requires an external pull-up resistor. SQW/OUT will operate with either Vcc or Vbat applied.

X1, X2 – Connections for a standard 32.768kHz quartz crystal. The internal oscillator circuitry is designed for operation with a crystal having a specified load capacitance (CL) of 12.5pF.

For more information on crystal selection and crystal layout considerations, please consult Application Note 58, "Crystal Considerations with Dallas Real-Time Clocks." The DS1307 can also be driven by an external 32.768kHz oscillator. In this configuration, the X1 pin is connected to the external oscillator signal and the X2 pin is floated.

RECOMMENDED LAYOUT FOR CRYSTAL



CLOCK ACCURACY

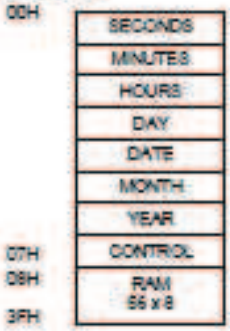
The accuracy of the clock is dependent upon the accuracy of the crystal and the accuracy of the match between the capacitive load of the oscillator circuit and the capacitive load for which the crystal was trimmed. Additional error will be added by crystal frequency drift caused by temperature shifts. External circuit noise coupled into the oscillator circuit may result in the clock running fast. See Application Note 58, "Crystal Considerations with Dallas Real-Time Clocks" for detailed information.

Please review Application Note 95, "Interfacing the DS1307 with a 8051-Compatible Microcontroller" for additional information.

RTC AND RAM ADDRESS MAP

The address map for the RTC and RAM registers of the DS1307 is shown in Figure 2. The RTC registers are located in address locations 00h to 07h. The RAM registers are located in address locations 08h to 3Fh. During a multi-byte access, when the address pointer reaches 3Fh, the end of RAM space, it wraps around to location 00h, the beginning of the clock space.

DS1307 ADDRESS MAP Figure 2



CLOCK AND CALENDAR

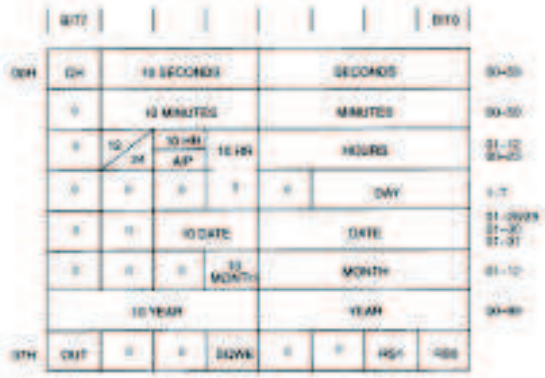
The time and calendar information is obtained by reading the appropriate register bytes. The RTC registers are illustrated in Figure 3. The time and calendar are set or initialized by writing the appropriate register bytes. The contents of the time and calendar registers are in the BCD format. Bit 7 of register 0 is the clock halt (CH) bit. When this bit is set to a 1, the oscillator is disabled. When cleared to a 0, the oscillator is enabled.

Please note that the initial power-on state of all registers is not defined. Therefore, it is important to enable the oscillator (CH bit = 0) during initial configuration.

The DS1307 can be run in either 12-hour or 24-hour mode. Bit 6 of the hours register is defined as the 12- or 24-hour mode select bit. When high, the 12-hour mode is selected. In the 12-hour mode, bit 5 is the AM/PM bit with logic high being PM. In the 24-hour mode, bit 5 is the second 10 hour bit (20-23 hours).

On a 2-wire START, the current time is transferred to a second set of registers. The time information is read from these secondary registers, while the clock may continue to run. This eliminates the need to re-read the registers in case of an update of the main registers during a read.

DS1307 TIMEKEEPER REGISTERS Figure 3



CONTROL REGISTER

The DS1307 control register is used to control the operation of the SQW/OUT pin.

BIT 7	BIT 6	BIT 5	BIT 4	BIT 3	BIT 2	BIT 1	BIT 0
OUT	0	0	SQWE	0	0	RS1	RS0

OUT (Output control): This bit controls the output level of the SQW/OUT pin when the square wave output is disabled. If SQWE = 0, the logic level on the SQW/OUT pin is 1 if OUT = 1 and is 0 if OUT = 0.

SQWE (Square Wave Enable): This bit, when set to a logic 1, will enable the oscillator output. The frequency of the square wave output depends upon the value of the RS0 and RS1 bits. With the square wave output set to 1Hz, the clock registers update on the falling edge of the square wave.

RS (Rate Select): These bits control the frequency of the square wave output when the square wave output has been enabled. Table 1 lists the square wave frequencies that can be selected with the RS bits.

SQUAREWAVE OUTPUT FREQUENCY Table 1

RS1	RS0	SQW OUTPUT FREQUENCY
0	0	1Hz
0	1	4.096kHz
1	0	8.192kHz
1	1	32.768kHz

ANEXO 3: HOJA DE DATOS 24LC 256

DATASHEET 24LC 256

MICROCHIP **24AA256/24LC256/24FC256**

256K I^2C^{TM} CMOS Serial EEPROM

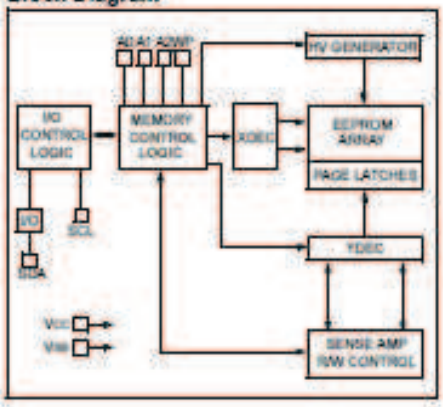
Features

- Low power CMOS technology
  - Maximum write current 3 mA at 5.5 V
  - Maximum read current 400 µA at 5.5 V
  - Standby current 100 nA typical at 5.5 V
- 2-wire serial interface bus, I²C compatible
- Cascadable for up to eight devices
- Self-timed ERASE/WRITE cycle
- 64-byte page-write mode available
- 5 ms max write-cycle time
- Hardware write protect for entire array
- Output slope control to eliminate ground bounce
- Schmitt trigger inputs for noise suppression
- 1,000,000 erase/write cycles
- Electrostatic discharge protection > 4000 V
- Data retention > 200 years
- 8-pin PDIP, SOIC, TSSOP, MSOP, and DFN packages
- 14-lead TSSOP package
- Temperature ranges:
  - Industrial (I): -40°C to +85°C
  - Automotive (E): -40°C to +125°C

Description

The Microchip Technology Inc. 24AA256/24LC256/24FC256 (24XX256*) is a 32K x 8 (256 Kbit) Serial Electrically Erasable PROM, capable of operation across a broad voltage range (1.8 V to 5.5 V). It has been developed for advanced, low power applications such as personal communications or data acquisition. This device also has a page-write capability of up to 64 bytes of data. This device is capable of both random and sequential reads up to the 256K boundary. Functional address lines allow up to eight devices on the same bus, for up to 2 Mbit address space. This device is available in the standard 8-pin plastic DIP, SOIC, TSSOP, MSOP, DFN and 14-lead TSSOP packages.

Block Diagram



Device Selection Table

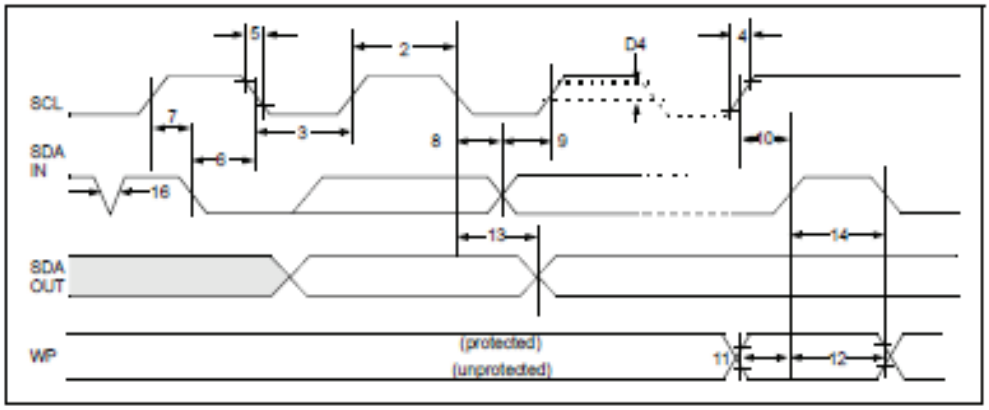
Part Number	Vcc Range	Max. Clock Frequency	Temp. Ranges
24AA256	1.8-5.5 V	400 kHz(1)	I
24LC256	2.5-5.5 V	400 kHz	I, E
24FC256	2.5-5.5 V	1 MHz	I

Note 1: 100 kHz for Vcc < 2.5 V.

Package Types

*24XX256 is used in this document as a generic part number for the 24AA256/24LC256/24FC256 devices.

FIGURE 1-1: BUS TIMING DATA



5.0 DEVICE ADDRESSING

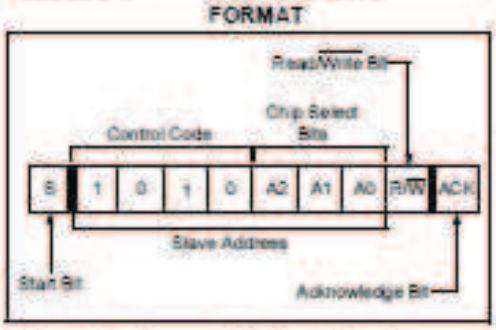
A control byte is the first byte received following the start condition from the master device (Figure 5-1). The control byte consists of a 4-bit control code. For the 24XX256, this is set as 1010 binary for read and write operations. The next three bits of the control byte are the chip select bits (A2, A1, A0). The chip select bits allow the use of up to eight 24XX256 devices on the same bus and are used to select which device is accessed. The chip select bits in the control byte must correspond to the logic levels on the corresponding A2, A1 and A0 pins for the device to respond. These bits are, in effect, the three most significant bits of the word address.

For the MSOP package, the A0 and A1 pins are not connected. During device addressing, the A0 and A1 chip select bits (Figures 5-1 and 5-2) should be set to '0'. Only two 24XX256 MSOP packages can be connected to the same bus.

The last bit of the control byte defines the operation to be performed. When set to a one, a read operation is selected. When set to a zero, a write operation is selected. The next two bytes received define the address of the first data byte (Figure 5-2). Because only A14...A0 are used, the upper address bits are a don't care. The upper address bits are transferred first, followed by the less significant bits.

Following the start condition, the 24XX256 monitors the SDA bus checking the device type identifier being transmitted. Upon receiving a 1010 code and appropriate device select bits, the slave device outputs an acknowledge signal on the SDA line. Depending on the state of the R/W bit, the 24XX256 will select a read or write operation.

FIGURE 5-1: CONTROL BYTE FORMAT

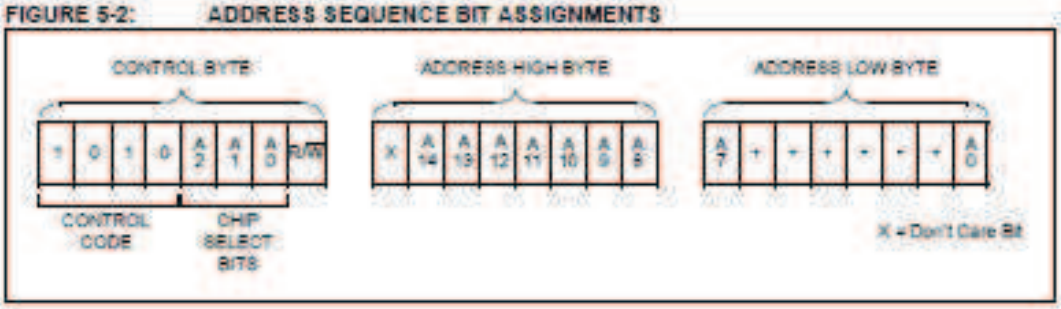


5.1 Contiguous Addressing Across Multiple Devices

The chip select bits A2, A1, A0 can be used to expand the contiguous address space for up to 2 Mbit by adding up to eight 24XX256s on the same bus. In this case, software can use A0 of the control byte as address bit A15; A1 as address bit A16; and A2 as address bit A17. It is not possible to sequentially read across device boundaries.

For the MSOP package, up to two 24XX256 devices can be added for up to 512 Kbit of address space. In this case, software can use A2 of the control byte as address bit A17. Bits A0 (A15) and A1 (A16) of the control byte must always be set to a logic '0' for the MSOP.

FIGURE 5-2: ADDRESS SEQUENCE BIT ASSIGNMENTS



6.0 WRITE OPERATIONS

6.1 Byte Write

Following the start condition from the master, the control code (four bits), the chip select (three bits) and the R/W bit (which is a logic low) are clocked onto the bus by the master transmitter. This indicates to the addressed slave receiver that the address high byte will follow after it has generated an acknowledge bit during the ninth clock cycle. Therefore, the next byte transmitted by the master is the high-order byte of the word address and will be written into the address pointer of the 24XX256. The next byte is the least significant address byte. After receiving another acknowledge signal from the 24XX256, the master device will transmit the data word to be written into the addressed memory location. The 24XX256 acknowledges again and the master generates a stop condition. This initiates the internal write cycle and during this time, the 24XX256 will not generate acknowledge signals (Figure 6-1). If an attempt is made to write to the array with the WP pin held high, the device will acknowledge the command but no write cycle will occur, no data will be written, and the device will immediately accept a new command. After a byte write command, the internal address counter will point to the address location following the one that was just written.

6.2 Page Write

The write control byte, word address and the first data byte are transmitted to the 24XX256 in much the same way as in a byte write. The exception is that instead of generating a stop condition, the master transmits up to 63 additional bytes, which are temporarily stored in the on-chip page buffer and will be written into memory once the master has transmitted a stop condition. Upon receipt of each word, the six lower address pointer bits are internally incremented by one. If the master should transmit more than 64 bytes prior to generating the stop condition, the address counter will roll over and the previously received data will be overwritten. As with the byte write operation, once the stop condition is received, an internal write cycle will begin (Figure 6-2). If an attempt is made to write to the array with the WP pin held high, the device will acknowledge the command but no write cycle will occur, no data will be written and the device will immediately accept a new command.

6.3 Write Protection

The WP pin allows the user to write-protect the entire array (0000-7FFF) when the pin is tied to Vcc. If tied to Vss or left floating, the write protection is disabled. The WP pin is sampled at the STOP bit for every write command (Figure 1-1). Toggling the WP pin after the STOP bit will have no effect on the execution of the write cycle.

Note: Page write operations are limited to writing bytes within a single physical page, regardless of the number of bytes actually being written. Physical page boundaries start at addresses that are integer multiples of the page buffer size (or 'page size') and end at addresses that are integer multiples of [page size - 1]. If a page write command attempts to write across a physical page boundary, the result is that the data wraps around to the beginning of the current page (overwriting data previously stored there), instead of being written to the next page, as might be expected. It is, therefore, necessary for the application software to prevent page write operations that would attempt to cross a page boundary.

FIGURE 6-1: BYTE WRITE

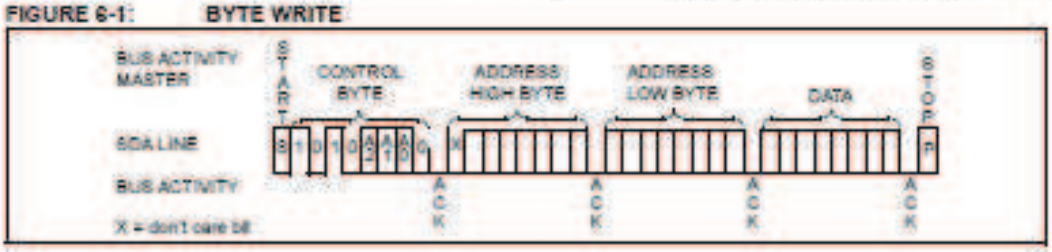
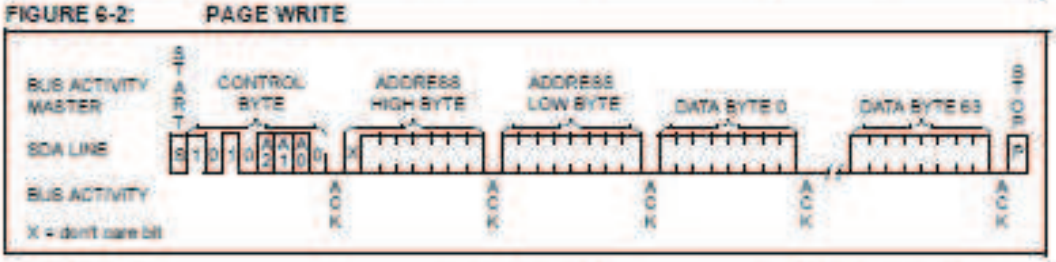


FIGURE 6-2: PAGE WRITE



ANEXO 4: DESCRIPCIÓN DEL LECTOR RFID ID-12

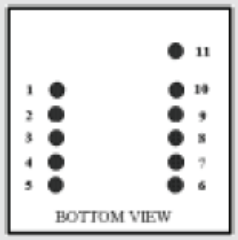
DESCRIPCIÓN DEL LECTOR RFID ID-12

ID SERIES DATASHEET MAR 01, 2005

ID-2/ID-12 Brief Data

The ID2. ID12 and ID20 are similar to the obsolete ID0, ID10 and ID15 MK(ii) series devices, but they have extra pins that allow Magnetic Emulation output to be included in the functionality. The ID-12 and ID-20 come with internal antennas, and have read ranges of 12+ cm and 16+ cm, respectively. With an external antenna, the ID-2 can deliver read ranges of up to 25 cm. All three readers support ASCII, Wiegand26 and Magnetic ABA Track2 data formats.

ID2 / ID12 / ID20 PIN-OUT



1. GND
2. RES (Reset Bar)
3. ANT (Antenna)
4. ANT (Antenna)
5. CP
6. Future
7. +/- (Format Selector)
8. D1 (Data Pin 1)
9. D0 (Data Pin 0)
10. LED (LED / Beeper)
11. +5V

Operational and Physical Characteristics

Parameters	ID-2	ID-12	ID-20
Read Range	N/A (no internal antenna)	12+ cm	16+ cm
Dimensions	21 mm x 19 mm x 6 mm	26 mm x 25 mm x 7 mm	40 mm x 40 mm x 9 mm
Frequency	125 kHz	125 kHz	125 kHz
Card Format	EM 4001 or compatible	EM 4001 or compatible	EM 4001 or compatible
Encoding	Manchester 64-bit, modulus 64	Manchester 64-bit, modulus 64	Manchester 64-bit, modulus 64
Power Requirement	5 VDC @ 13mA nominal	5 VDC @ 30mA nominal	5 VDC @ 65mA nominal
I/O Output Current	+/-200mA PK	-	-
Voltage Supply Range	+4.6V through +5.4V	+4.6V through +5.4V	+4.6V through +5.4V

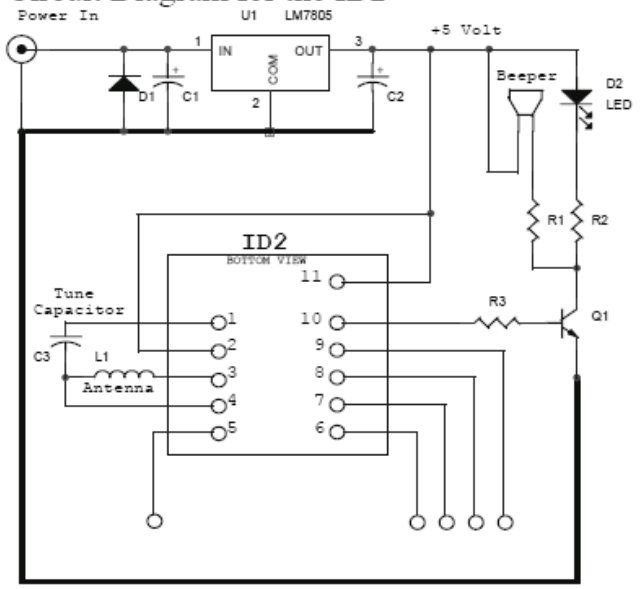
Pin Description & Output Data Formats

Pin No.	Description	ASCII	Magnet Emulation	Wiegand26
Pin 1	Zero Volts and Tuning Capacitor Ground	GND 0V	GND 0V	GND 0V
Pin 2	Strap to +5V	Reset Bar	Reset Bar	Reset Bar
Pin 3	To External Antenna and Tuning Capacitor	Antenna	Antenna	Antenna
Pin 4	To External Antenna	Antenna	Antenna	Antenna
Pin 5	Card Present	No function	Card Present *	No function

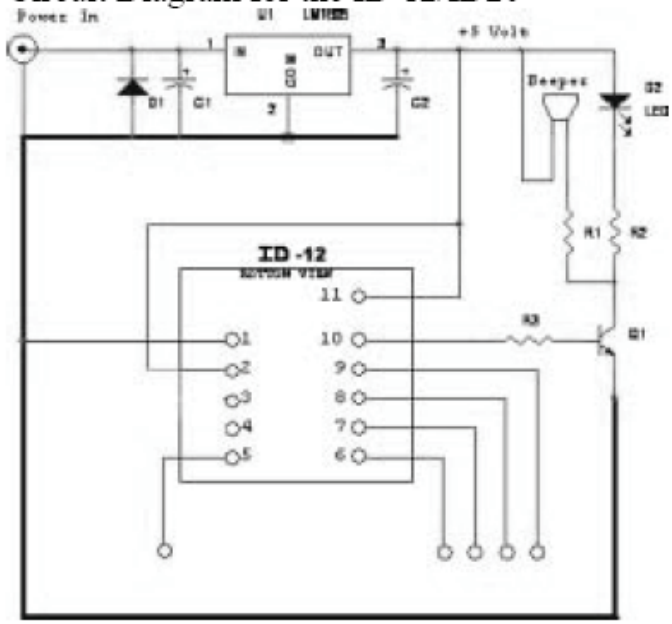
Pin 6	Future	Future	Future	Future
Pin 7	Format Selector (+/-)	Strap to GND	Strap to Pin 10	Strap to +5V
Pin 8	Data 1	CMOS	Clock *	One Output *
Pin 9	Data 0	TTL Data (inverted)	Data *	Zero Output *
Pin 10	3.1 kHz Logic	Beeper / LED	Beeper / LED	Beeper / LED
Pin 11	DC Voltage Supply	+5V	+5V	+5V

* Requires 4K7 Pull-up resistor to +5V

Circuit Diagram for the ID2



Circuit Diagram for the ID-12/ID20



DATA FORMATS

Output Data Structure – ASCII

STX (02h)	DATA (10 ASCII)	CHECK SUM (2 ASCII)	CR	LF	ETX (03h)

[The 1byte (2 ASCII characters) Check sum is the "Exclusive OR" of the 5 hex bytes (10 ASCII) Data characters.]

Output Data Structure – Wiegand26

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26
P	E	E	E	E	E	E	E	E	E	E	E	E	O	O	O	O	O	O	O	O	O	O	O	O	P
Even parity (E)													Odd parity (O)												

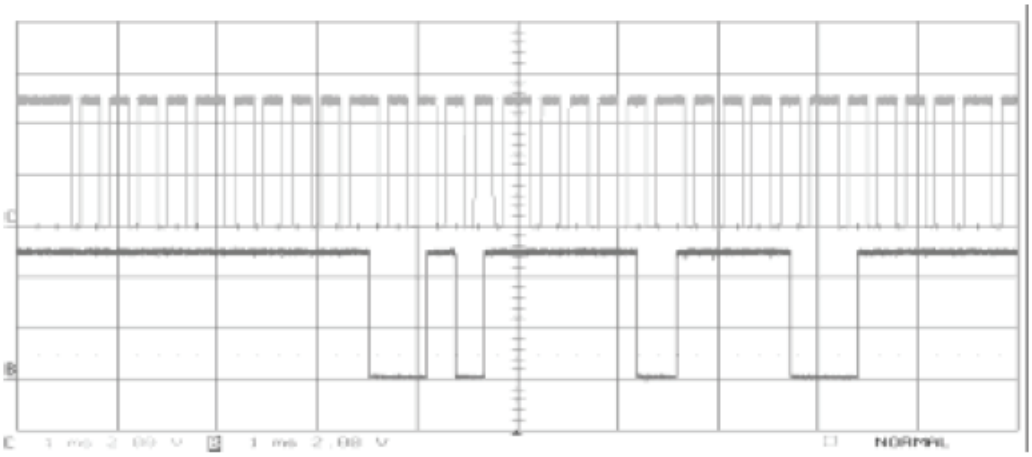
P = Parity start bit and stop bit

Output Data Magnetic ABA Track2

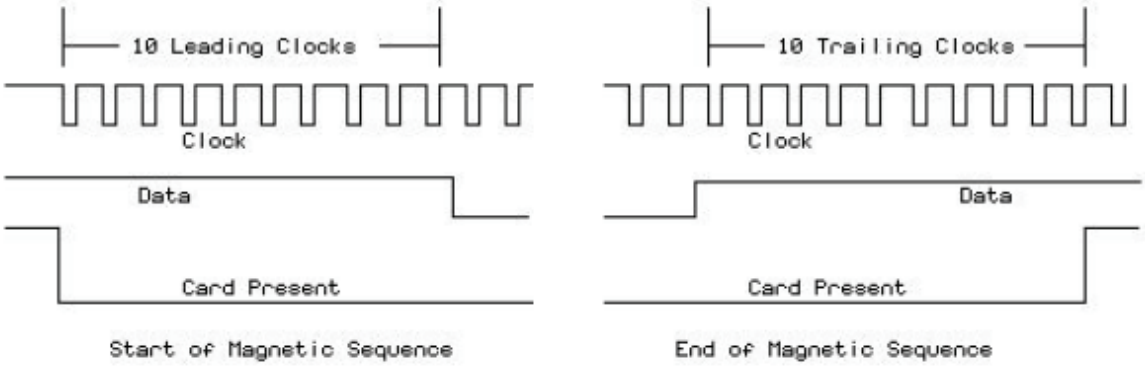
10 Leading Zeros	SS	Data	ES	LCR	10 Ending Zeros

[SS is the Start Character of 11010, ES is the end character of 11111, LRC is the Longitudinal Redundancy Check.]

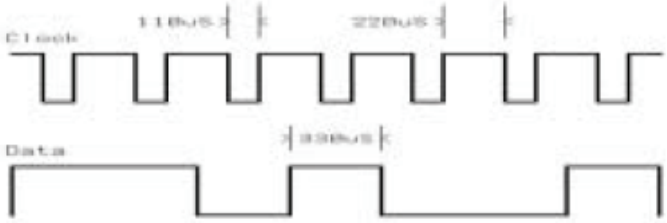
Magnetic Emulation Waveforms



Start and End Sequences For Magnetic Timing

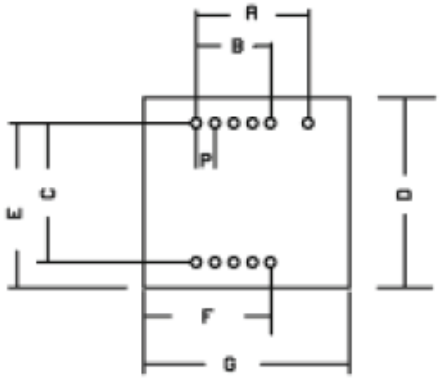


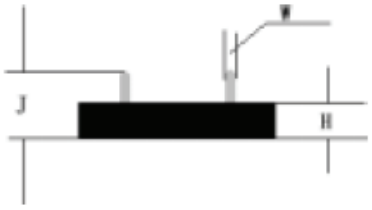
DATA TIMINGS FOR MAGNETIC EMULATION



The magnetic Emulation Sequence starts with the Card Present Line going active (down). There next follows 10 clocks with Zero '0' data. At the end of the 10 leading clocks the start character (11010) is sent and this is followed by the data. At the end of the data the end character is sent followed by the LCR. Finally 10 trailing clocks are sent and the card present line is raised.
The data bit duration is approximately 330uS. The approximate clock duration is 110uS. Because of the symmetry data can be clocked off either the rising or falling edge of the clock.

Dimensions (Top View) (mm)





	ID-0/ID-2wr			ID-10/ID-12wr			ID-15/ID-20wr		
	Nom.	Min.	Max.	Nom.	Min.	Max.	Nom.	Min.	Max
A	12.0	11.6	12.4	12.0	11.6	12.4	12.0	11.6	12.4
B	8.0	7.6	8.4	8.0	7.6	8.4	8.0	7.6	8.4
C	15.0	14.6	15.4	15.0	14.6	15.4	15.0	14.6	15.4
D	20.5	20.0	21.5	25.3	24.9	25.9	40.3	40.0	41.0
E	18.5	18.0	19.2	20.3	19.8	20.9	27.8	27.5	28.5
F	14.0	13.0	14.8	16.3	15.8	16.9	22.2	21.9	23.1
G	22.0	21.6	22.4	26.4	26.1	27.1	38.5	38.2	39.2
P	2.0	1.8	2.2	2.0	1.8	2.2	2.0	1.8	2.2
H	5.92	5.85	6.6	6.0	5.8	6.6	6.8	6.7	7.0
J	9.85	9.0	10.5	9.9	9.40	10.5	9.85	9.4	10.6
W	0.66	0.62	0.67	0.66	0.62	0.67	0.66	0.62	0.67

Note – measurements do not include any burring of edges.
NOTICE - Innovated Devices reserve the right to change these specifications without prior notice.

Designing Coils for ID2

The recommended Inductance is 1.08mH to be used with an internal tuning capacitor of 1n5. In general the bigger the antenna the better, provided the reader is generating enough field strength to excite the tag. The ID-2 is relatively low power so a maximum coil size of 15x15cm is recommended if it is intended to read ISO cards. If the reader is intended to read glass tags the maximum coil size should be smaller, say 10x10cm.

There is a science to determine the exact size of an antenna but there are so many variables that in general it is best to get a general idea after which a degree of 'Try it and see' is unavoidable.

If the reader is located in a position where there is a lot of heavy interference then less range cannot be avoided. In this situation the coil should be made smaller to increase the field strength and coupling.

It is difficult to give actual examples of coils for hand wounding because the closeness and tightness of the winding will significantly change the inductance. A professionally wound coil will have much more inductance than a similar hand wound coil.

For those who want a starting point into practical antenna winding it was found that 63 turns on a 120mm diameter former gave an inductance of 1.08mH. For those contemplating adding an additional tuning capacitor it was found that 50 turns on a 120mm diameter former gave 700uH. The wire diameter is not important.
Anybody who wishes to be more theoretical we recommend a trip to the Microchip Website where we found an application sheet for Loop Antennas.
http://ww1.microchip.com/downloads/en/AppNotes/00831b.pdf

The Tuning Capacitor

It is recommended that the internal 1n% capacitor is used for tuning, however a capacitor may be also be added externally. The combined capacitance should not exceed 2n7. Do not forget that the choice of tuning capacitor can also substantially affect the quality of your system. The Id12 is basically an ID2 with an internal antenna. The loss in an ID12 series antenna is required to be fairly high to limit the series current. A low Q will hide a lot of the shortcomings of the capacitor, but for quality and reliability and repeatability the following capacitors are recommend.

Polypropylene	Good Readily available. Ensure AC voltage at 125kHz is sufficient.
COG/NPO	Excellent. Best Choice
Silver Mica	Excellent but expensive
Polycarbonate	Good Readily available. Ensure AC voltage at 125kHz is sufficient.

Voltage Working.

A capacitor capable of withstanding the RMS voltage at 125KHz MUST be chosen. The working voltage will depend on the coil design. I suggest the designer start with rugged 1n5 Polypropylene 630v capacitor to do his experiments and the come down to a suitable size/value. The capacitor manufacturer will supply information on their capacitors. Do not simply go by the DC voltage. This means little. A tolerance of 2% is preferable. A tolerance of 5% is acceptable.

Fine Tuning

We recommend using an oscilloscope for fine-tuning. Connect the oscilloscope to observe the 125KHz AC voltage across the coil. Get a sizeable piece of ferrite and bring it up to the antenna loop. If the voltage increases then you need more inductance (or more capacitance). If the voltage decreases as you bring the ferrite up to the antenna then the inductance is too great. If you have no ferrite then a piece of aluminum

ANEXO 5: PROGRAMA FUENTE DE BASCOM AVR

PROGRAMA FUENTE DE BASCOM AVR

```
'funciona con VB Control de Acceso 08

$regfile = "m164pdef.dat"
$crystal = 20000000
$baud = 9600

Config Com1 = Dummy , Synchrone = 0 , Parity = None , Stopbits = 1 , Databits = 8 , Clockpol = 0

Cls : Cursor Off Noblink

'Definicion Para Manejo De Memorias Eeprom
Const Cod_esc = 160 : Const Cod_lec = 161              'Codigo de Escritura y Lectura de EEPROM
Config Scl = Portc.0 : Config Sda = Portc.1            'Seleccion de pines de comunicacion para memoria EEPROM
Dim Direccion As Word At $100 , Direccion_h As Byte At $101 Overlay , Direccion_l As Byte At $100 Overlay      'declara las variables en
el mismo espacio de memoria
Dim Dato_memoria As Byte

'Variables Reloj
Const Ds1307w = &HD0 : Const Ds1307r = &HD1            'Configura Direcciones I2c Del Ds1307
Dim _weekday As Byte
Dim _hour As Byte , _min As Byte , _sec As Byte
Dim Am_pm As Bit                                       ' Si Am_pm = 1 then es PM
Dim _day As Byte , _month As Byte , _year As Byte

'declaracion de constantes
Const Buffer_in_long = 21                              'indica la longitud del buffer de entrada
Dim Buffer_mat_in As String * Buffer_in_long At $110       'guarda la informacion recibida
Dim Pc_horario As String * 20 At $110 Overlay
Dim Pc_horario_mat(20) As Byte At $110 Overlay

Ddrb.0 = 0 : Portb.0 = 1 : Jumper_p1 Alias Pinb.0       'Configura Entrada, Resistecias Pull up, Alias
Ddrb.1 = 0 : Portb.1 = 1 : Jumper_p2 Alias Pinb.1
Ddrb.2 = 0 : Portb.2 = 1 : Jumper_p3 Alias Pinb.2
Ddra.0 = 0 : Porta.0 = 1 : Pc_conectada Alias Pina.0

Ddrd.5 = 1 : Portd.5 = 1 : Led_open_puerta Alias Portd.5 'Configura Salida, salida en 1 Logico, Alias
Ddrd.6 = 1 : Portd.6 = 1 : Led_close_puerta Alias Portd.6
Ddra.2 = 1 : Porta.2 = 1 : Chicharra Alias Porta.2

'variables seriales

Dim Serial_char As Byte                                'recibe el caracter serial
Dim Con_char_in As Byte                                'determina la posicion el la cual guardar los bytes recibidos
Dim For_serial As Byte

'Variables    'WIEGAND
Dim W_cont As Byte                                     'cuenta 26 bits de entrada
Dim Cont_bit As Byte                                   'posiciona los bits ingresados
Dim W1 As Byte                                         'guarda los bits ingresados
Dim W2 As Byte
Dim W3 As Byte
Dim W4 As Byte

Dim W1_men As Byte                                     'guarda los bits ingresados
Dim W2_men As Byte
Dim W3_men As Byte
Dim W4_men As Byte
Dim W5_men As Byte                                     'guarda los bits ingresados
Dim W6_men As Byte
Dim W7_men As Byte
Dim W8_men As Byte
```

```
'Datos usados para igualar el relog
Dim Buscar_fin As Byte
Dim Buscar_init As Byte
Dim Var_dat_msb As Byte
Dim Inicio_user_dir As Byte
Dim Var_dat_lsb As Byte
Dim H_f As String * 2
Dim B_i As String * 2
Dim I_d As String * 2
Dim D_r As String * 2
Dim L_t As String * 2
Dim S_v As String * 2
'determina el numero de puerta conectada
Dim C_x As String * 2
'variables para control de usuario
Dim U_s As String * 2
Dim Band_serial As Bit
Dim Band_wiegan As Bit

'lectura de variables de la  memoria
Dim Num_user_registed As Byte
Dim Num_puerta As Byte

Dim For_memoria As Byte
Dim Band_cod_hallado As Byte                'determina si el codigo de una tarjeta se encuentra grabado en la memoria
Dim Ingreso_usuario As Byte                 'determina cual fue el usuario que ingreso
Dim Dir_ingreso_user As Word                'DETERMINA LA DIRECCION DONDE GUARDAR LOS DATOS DE CADA USUARIO
Dim Num_user_ing As Byte                    'DETERMINA EL NUMERO DE USUARIOS QUE HANINGRESADO MAXIMO 255
Dim Char_men As String * 1                  'GUARDA LOS CARACTERES EN LA MEMEORIA

Dim St_w1_men As String * 2
Dim St_w2_men As String * 2
Dim St_w3_men As String * 2
Dim St_w4_men As String * 2

Dim String_w1 As String * 2
Dim String_w2 As String * 2
Dim String_w3 As String * 2
Dim String_w4 As String * 2

Dim Save_men As String * 2                  'utilizada para guardar la informacion del usuario que ingreso en la memoria
Dim Convertir_valor As Byte

'variables Usadas Para Enviar El Ingreso De Los Usuarios
Dim Buffer_out As String * 21

'variables para mootor
Dim For_motor As Byte

'WIEGAND   habilita interruciones para Deteccion de Codigo
On Int0 Wiegand26_0 : Enable Int0 : On Int1 Wiegand26_1 : Enable Int1
Config Int0 = Rising : Config Int1 = Rising

'Habilita Interrucion SERIAL
On Urxc Conexion_pc                         'ACTIVAR LA SUBRUTINA DE INTERRUPCION
Enable Urxc

Gosub Init_serial
Gosub Init_wiegand
Gosub Verificar_horario                     'verifica que el integrado de reloj se encuentre con datos validos

'MENU PRINCIPAL

Waitms 500

Led_open_puerta = 0
Led_close_puerta = 1
Chicharra = 0
```

```
Enable Interrupts                              'ACTIVAR TODAS LAS INTERRUPCIONES

Do

If Con_char_in > 14 And Band_serial = 1 Then          'ingresa aqui cuando algun dato serial a ingresado al micro

'determino que que tipo de accion tiene que ser realizada
'mediante la busquedas de los comandos dentro de la string
'que ingreso

Buscar_init = Instr(pc_horario , "HF")           'comando para igualar hora

If Buscar_init = 0 Then
Buscar_init = Instr(pc_horario , "US")          'para grabar un usuario
End If

If Buscar_init = 0 Then
Buscar_init = Instr(pc_horario , "BM")          'para borrar el numero de usuarios que ingresaron
End If

If Buscar_init = 0 Then
Buscar_init = Instr(pc_horario , "ID")          'para pedir el numero de puerta conectada
End If

If Buscar_init = 0 Then
Buscar_init = Instr(pc_horario , "DM")          'para borrar el registro de usuarios
End If

If Buscar_init = 0 Then
Buscar_init = Instr(pc_horario , "LT")          'para leer el registro de usuarios que ingresaron
End If

If Buscar_init = 0 Then
Buscar_init = Instr(pc_horario , "SV")          'para guardar los usarios en la memoria
End If

If Buscar_init = 0 Then
Buscar_init = Instr(pc_horario , "CX")          'para guardar los usarios en la memoria
End If

H_f = Mid(pc_horario , Buscar_init , 2)
U_s = H_f : B_i = H_f : I_d = H_f : D_r = H_f : L_t = H_f : S_v = H_f : C_x = H_f

Waitms 300                                     'PAUSA IMMOVIBLE

If Buscar_init = 0 Then
Print "EEE"                                   'envia en caso de error
Else
Goto Continuar
End If

Continuar:

If B_i = "BI" Then                             'BI BORRAR INGRESO
Direccion = 2 : Dato_memoria = 0 : Gosub Write_eeprom     'borra la cantidad de usuarios QUE INGRESARON
Print "RBRB"                                  'REGISTRO BORRADO
End If

If D_r = "DR" Then
Direccion = 1 : Dato_memoria = 0 : Gosub Write_eeprom     'borra numero total de usuarios registrados
Direccion = 2 : Dato_memoria = 0 : Gosub Write_eeprom     'borra la cantidad de usuarios registrados
Print "RERE"                                  'REGISTRO ELIMINADO
End If

If I_d = "ID" Then
Direccion = 2 : Gosub Read_eeprom : Num_user_ing = Dato_memoria
If Num_user_ing < 10 Then : Convertir_valor = Num_user_ing : Gosub Asignar_valor : Else : Save_men = Str(num_user_ing) : End If

If Jumper_p1 = 0 Then
Print "111AA" ; Save_men ; "BB"
```

```
End If

If Jumper_p2 = 0 Then
Print "222AA" ; Save_men ; "BB"
End If

If Jumper_p3 = 0 Then
Print "333AA" ; Save_men ; "BB"
End If

If C_x = "CX" Then                          'CX CONEXION


If Jumper_p1 = 0 Then
Print "PC1"                          'PUERTA CONECTADA 1
End If

If Jumper_p2 = 0 Then
Print "PC2"
End If

If Jumper_p3 = 0 Then
Print "PC3"
End If

If H_f = "HF" Then
Buscar_fin = Instr(pc_horario , "FIN")

If Buscar_fin > 0 Then

'hora
Var_dat_msb = Pc_horario_mat(2 + Buscar_init) : Var_dat_msb = Var_dat_msb -48
Var_dat_lsb = Pc_horario_mat(3 + Buscar_init) : Var_dat_lsb = Var_dat_lsb -48

Var_dat_msb = Var_dat_msb * 10
 _hour = Var_dat_msb + Var_dat_lsb

'minuto
Var_dat_msb = Pc_horario_mat(4 + Buscar_init) : Var_dat_msb = Var_dat_msb -48
Var_dat_lsb = Pc_horario_mat(5 + Buscar_init) : Var_dat_lsb = Var_dat_lsb -48

Var_dat_msb = Var_dat_msb * 10
 _min = Var_dat_msb + Var_dat_lsb
 _sec = 0

If Pc_horario_mat(6 + Buscar_init) = 80 Then : Am_pm = 1 : Else : Am_pm = 0 : End If
'80 valor decimal de "P"

'dia
        Var_dat_msb = Pc_horario_mat(7 + Buscar_init) : Var_dat_msb = Var_dat_msb -48
        Var_dat_lsb = Pc_horario_mat(8 + Buscar_init) : Var_dat_lsb = Var_dat_lsb -48

        Var_dat_msb = Var_dat_msb * 10
        _day = Var_dat_msb + Var_dat_lsb

        'mes
        Var_dat_msb = Pc_horario_mat(9 + Buscar_init) : Var_dat_msb = Var_dat_msb -48
        Var_dat_lsb = Pc_horario_mat(10 + Buscar_init) : Var_dat_lsb = Var_dat_lsb -48

        Var_dat_msb = Var_dat_msb * 10
        _month = Var_dat_msb + Var_dat_lsb

        'año
        Var_dat_msb = Pc_horario_mat(11 + Buscar_init) : Var_dat_msb = Var_dat_msb -48
        Var_dat_lsb = Pc_horario_mat(12 + Buscar_init) : Var_dat_lsb = Var_dat_lsb -48

        Var_dat_msb = Var_dat_msb * 10
        _year = Var_dat_msb + Var_dat_lsb

        Gosub Igualar_hora
```

```
    Gosub Igualar_fecha

    Print "HGHG"

  Else
    Print "EEE"
  End If

End If

If U_s = "US" Or S_v = "SV" Then

  Buscar_fin = Instr(pc_horario , "FIN")

  If Buscar_fin > 0 Then

    Var_dat_msb = Pc_horario_mat(2 + Buscar_init) : Var_dat_msb = Var_dat_msb -48
    Var_dat_lsb = Pc_horario_mat(3 + Buscar_init) : Var_dat_lsb = Var_dat_lsb -48

    Var_dat_msb = Var_dat_msb * 10
    Num_user_registed = Var_dat_msb + Var_dat_lsb   'numero de usuarios registrados

    Direccion = 1 : Dato_memoria = Num_user_registed : Gosub Write_eeprom

    'determina direccion donde guardar el codigo del usuario

    Var_dat_msb = Pc_horario_mat(4 + Buscar_init) : Var_dat_msb = Var_dat_msb -48
    Var_dat_lsb = Pc_horario_mat(5 + Buscar_init) : Var_dat_lsb = Var_dat_lsb -48

    Var_dat_msb = Var_dat_msb * 10
    Inicio_user_dir = Var_dat_msb + Var_dat_lsb     'numero de usuarios registrados

    '------------------------------------------------------------

    Var_dat_msb = Pc_horario_mat(6 + Buscar_init) : Var_dat_msb = Var_dat_msb -48
    Var_dat_lsb = Pc_horario_mat(7 + Buscar_init) : Var_dat_lsb = Var_dat_lsb -48

    Var_dat_msb = Var_dat_msb * 10
    Num_puerta = Var_dat_msb + Var_dat_lsb

    Direccion = 10 * Inicio_user_dir : Dato_memoria = Num_puerta : Gosub Write_eeprom

    Incr Direccion : Dato_memoria = Pc_horario_mat(8 + Buscar_init) : Gosub Write_eeprom
    Incr Direccion : Dato_memoria = Pc_horario_mat(9 + Buscar_init) : Gosub Write_eeprom
    Incr Direccion : Dato_memoria = Pc_horario_mat(10 + Buscar_init) : Gosub Write_eeprom
    Incr Direccion : Dato_memoria = Pc_horario_mat(11 + Buscar_init) : Gosub Write_eeprom
    Incr Direccion : Dato_memoria = Pc_horario_mat(12 + Buscar_init) : Gosub Write_eeprom
    Incr Direccion : Dato_memoria = Pc_horario_mat(13 + Buscar_init) : Gosub Write_eeprom
    Incr Direccion : Dato_memoria = Pc_horario_mat(14 + Buscar_init) : Gosub Write_eeprom
    Incr Direccion : Dato_memoria = Pc_horario_mat(15 + Buscar_init) : Gosub Write_eeprom

    If Num_puerta = 0 And U_s = "US" Then
      Print "UBUB"                        'USUARIO BORRADO
    Else

      If U_s = "US" Then
        Print "URUR"                      'USUARIO REGISTRADO

      Elseif S_v = "SV" Then
        Print "UGUG"                       'USUARIOS GUARDADOS
      End If

    End If

  Else
    Print "EEE"
  End If

End If
```

```
  If L_t = "LT" Then
  'cuando se lee los usarios que ingresaron
  'dependiendo de la direccion que envie la pc
  'se detrmina los datos a enviar

    Buscar_fin = Instr(pc_horario , "FIN")

    If Buscar_fin > 0 Then

      Var_dat_msb = Pc_horario_mat(2 + Buscar_init) : Var_dat_msb = Var_dat_msb -48
      Var_dat_lsb = Pc_horario_mat(3 + Buscar_init) : Var_dat_lsb = Var_dat_lsb -48

      Var_dat_msb = Var_dat_msb * 10
      Direccion = Var_dat_msb + Var_dat_lsb

      Direccion = Direccion * 20
      Direccion = Direccion + 300

      Mid(buffer_out , 1 , 2) = "US"
      Gosub Read_eeprom : Mid(buffer_out , 3 , 1) = Dato_memoria
      Incr Direccion : Gosub Read_eeprom : Mid(buffer_out , 4 , 1) = Dato_memoria
      Incr Direccion : Gosub Read_eeprom : Mid(buffer_out , 5 , 1) = Dato_memoria
      Incr Direccion : Gosub Read_eeprom : Mid(buffer_out , 6 , 1) = Dato_memoria
      Incr Direccion : Gosub Read_eeprom : Mid(buffer_out , 7 , 1) = Dato_memoria
      Incr Direccion : Gosub Read_eeprom : Mid(buffer_out , 8 , 1) = Dato_memoria
      Incr Direccion : Gosub Read_eeprom : Mid(buffer_out , 9 , 1) = Dato_memoria
      Incr Direccion : Gosub Read_eeprom : Mid(buffer_out , 10 , 1) = Dato_memoria
      Incr Direccion : Gosub Read_eeprom : Mid(buffer_out , 11 , 1) = Dato_memoria
      Incr Direccion : Gosub Read_eeprom : Mid(buffer_out , 12 , 1) = Dato_memoria
      Incr Direccion : Gosub Read_eeprom : Mid(buffer_out , 13 , 1) = Dato_memoria
      Incr Direccion : Gosub Read_eeprom : Mid(buffer_out , 14 , 1) = Dato_memoria
      Incr Direccion : Gosub Read_eeprom : Mid(buffer_out , 15 , 1) = Dato_memoria
      Incr Direccion : Gosub Read_eeprom : Mid(buffer_out , 16 , 1) = Dato_memoria
      Incr Direccion : Gosub Read_eeprom : Mid(buffer_out , 17 , 1) = Dato_memoria
      Incr Direccion : Gosub Read_eeprom : Mid(buffer_out , 18 , 1) = Dato_memoria
      Mid(buffer_out , 19 , 2) = "XW"

      Print Buffer_out

    Else
      Print "EEE"
    End If

  End If

  Gosub Init_serial                    'limpia matriz e inicializa contadores
  Gosub Init_wiegand
End If

If W_cont > 24 And Band_wiegan = 1 Then          'entra aqui cuando el micro ha detectado una tarjeta

  Direccion = 1 : Gosub Read_eeprom : Num_user_registed = Dato_memoria

  If Num_user_registed > 20 Then Num_user_registed = 20 'deve buscar un maximo de 20 usuarios

  Band_cod_hallado = 0

  For For_memoria = 1 To Num_user_registed

    'determino la direccion y leo el codigo de tag que contiene
    'el codigo esta guardado como una string de 8 caracteres

    Direccion = For_memoria * 10

    Incr Direccion : Gosub Read_eeprom : W1_men = Dato_memoria
    Incr Direccion : Gosub Read_eeprom : W2_men = Dato_memoria
    Incr Direccion : Gosub Read_eeprom : W3_men = Dato_memoria
    Incr Direccion : Gosub Read_eeprom : W4_men = Dato_memoria
    Incr Direccion : Gosub Read_eeprom : W5_men = Dato_memoria
    Incr Direccion : Gosub Read_eeprom : W6_men = Dato_memoria
```

```
    Incr Direccion : Gosub Read_eeprom : W7_men = Dato_memoria
    Incr Direccion : Gosub Read_eeprom : W8_men = Dato_memoria

    'w1 a w4 tienen el codigo del tag en decimal asi que se transforma a string

    String_w1 = Hex(w1)
    String_w2 = Hex(w2)
    String_w3 = Hex(w3)
    String_w4 = Hex(w4)
    'lleno los caracteres de cada tag de la memoria en strings para poder comparalas con el valor detectado
    Mid(st_w1_men , 1 , 1) = W1_men
    Mid(st_w1_men , 2 , 1) = W2_men

    Mid(st_w2_men , 1 , 1) = W3_men
    Mid(st_w2_men , 2 , 1) = W4_men

    Mid(st_w3_men , 1 , 1) = W5_men
    Mid(st_w3_men , 2 , 1) = W6_men

    Mid(st_w4_men , 1 , 1) = W7_men
    Mid(st_w4_men , 2 , 1) = W8_men
    'comparo si el valor existe y de ser asi salgo
    If String_w1 = St_w1_men And String_w2 = St_w2_men And String_w3 = St_w3_men And String_w4 = St_w4_men Then
      Band_cod_hallado = 1
      Exit For

    End If

  Next For_memoria

  If Band_cod_hallado = 1 Then                    'ingresa aquie siexsite el codigo de la tarjeta grabada en la memoria

    'determino a que puerta corresponde el codigo
    Direccion = For_memoria * 10 : Gosub Read_eeprom : Num_puerta = Dato_memoria
    Ingreso_usuario = For_memoria
    Led_open_puerta = 0

    'determino a que puerta esta conectada el microntrolador para verificar si puede o no ingresar

    If Num_puerta = 3 Or Num_puerta = 2 Or Num_puerta = 1 And Jumper_p1 = 0 Then
      Gosub Save_user_ingreso
      Led_close_puerta = 0 : Led_open_puerta = 1 : Chicharra = 1 : Wait 2 : Chicharra = 0 : Led_open_puerta = 0 : Led_close_puerta =
1
    End If

    If Num_puerta = 3 Or Num_puerta = 2 And Jumper_p2 = 0 Then
      Gosub Save_user_ingreso
       Led_close_puerta = 0 : Led_open_puerta = 1 : Chicharra = 1 : Wait 2 : Chicharra = 0 : Led_open_puerta = 0 : Led_close_puerta =
1
    End If

    If Num_puerta = 3 And Jumper_p3 = 0 Then
      Gosub Save_user_ingreso
       Led_close_puerta = 0 : Led_open_puerta = 1 : Chicharra = 1 : Wait 2 : Chicharra = 0 : Led_open_puerta = 0 : Led_close_puerta =
1
    End If

  Else

    'ingresa aqui si no existe el codigo detectado grabado en la memoria

    If Pc_conectada = 1 Then                    'verifica si la computadora se encuentra conectada  para enviar el codigo detectado

      Print "NEW" ; Hex(w1) ; Hex(w2) ; Hex(w3) ; Hex(w4) ; "FIN"

    Else
    'mantengo la puerta cerrada

      Led_open_puerta = 0 : Led_close_puerta = 1 : Chicharra = 0
```

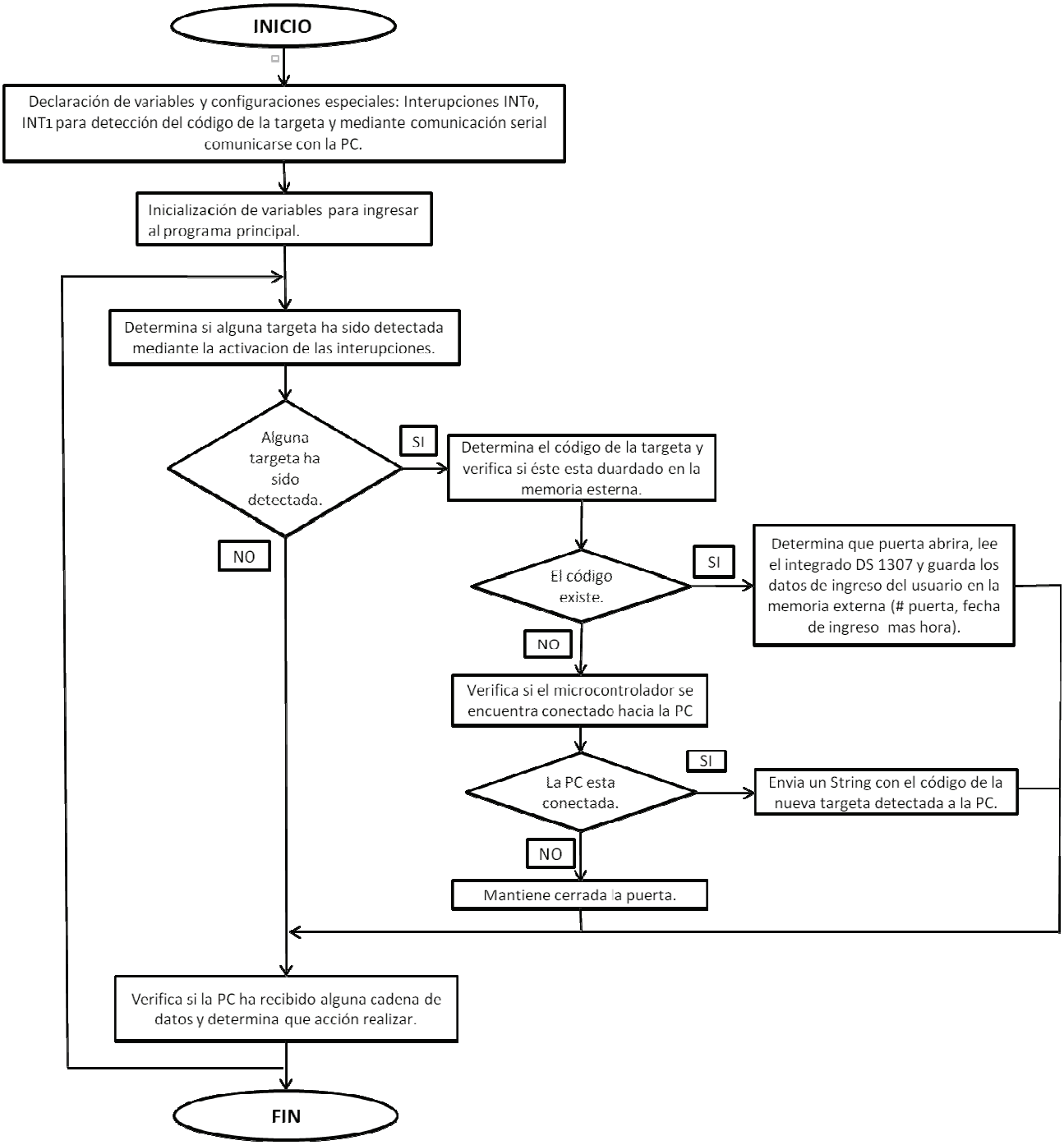
```
        End If

      End If

     Gosub Init_serial
     Gosub Init_wiegand

   End If

Loop


Save_user_ingreso:

    Gosub Leer_horario

    Direccion = 2 : Gosub Read_eeprom : Num_user_ing = Dato_memoria
    Dir_ingreso_user = Num_user_ing * 20
    Incr Num_user_ing : Direccion = 2 : Dato_memoria = Num_user_ing : Gosub Write_eeprom
    Direccion = 300 + Dir_ingreso_user

    If Ingreso_usuario < 10 Then : Convertir_valor = Ingreso_usuario : Gosub Asignar_valor : Else : Save_men = Str(ingreso_usuario) : End
If

    Char_men = Save_men : Dato_memoria = Char_men : Gosub Write_eeprom
    Incr Direccion : Char_men = Mid(save_men , 2 , 1) : Dato_memoria = Char_men : Gosub Write_eeprom


    If _hour < 10 Then : Convertir_valor = _hour : Gosub Asignar_valor : Else : Save_men = Str(_hour) : End If : Gosub Guardar_memoria
    If _min < 10 Then : Convertir_valor = _min : Gosub Asignar_valor : Else : Save_men = Str(_min) : End If : Gosub Guardar_memoria
    If _sec < 10 Then : Convertir_valor = _sec : Gosub Asignar_valor : Else : Save_men = Str(_sec) : End If : Gosub Guardar_memoria
    If Am_pm = 0 Then : Save_men = "AM" : Else : Save_men = "PM" : End If : Gosub Guardar_memoria
    If _day < 10 Then : Convertir_valor = _day : Gosub Asignar_valor : Else : Save_men = Str(_day) : End If : Gosub Guardar_memoria
    If _month < 10 Then : Convertir_valor = _month : Gosub Asignar_valor : Else : Save_men = Str(_month) : End If : Gosub
Guardar_memoria
    If _year < 10 Then : Convertir_valor = _year : Gosub Asignar_valor : Else : Save_men = Str(_year) : End If : Gosub Guardar_memoria

    Return

    Guardar_memoria:

   Incr Direccion : Char_men = Save_men : Dato_memoria = Char_men : Gosub Write_eeprom
   Incr Direccion : Char_men = Mid(save_men , 2 , 1) : Dato_memoria = Char_men : Gosub Write_eeprom

   Return

   Asignar_valor:

   Select Case Convertir_valor

      Case 0 : Save_men = "00"
      Case 1 : Save_men = "01"
      Case 2 : Save_men = "02"
      Case 3 : Save_men = "03"
      Case 4 : Save_men = "04"
      Case 5 : Save_men = "05"
      Case 6 : Save_men = "06"
      Case 7 : Save_men = "07"
      Case 8 : Save_men = "08"
      Case 9 : Save_men = "09"

   End Select

Return

'Interrupcion Int0 detecta entrada de CEROS
Wiegand26_0:
Nop : Nop : Nop : Nop : Nop : Nop : Nop : Nop
Incr W_cont
```

```
  If W_cont < 8 Then
    W1.cont_bit = 0
  Elseif W_cont < 16 Then
    W2.cont_bit = 0
  Elseif W_cont < 24 Then
    W3.cont_bit = 0
  Elseif W_cont < 27 Then
    W4.cont_bit = 0
  End If

  Decr Cont_bit
  If Cont_bit > 8 Then : Cont_bit = 7 : End If

  Band_serial = 0
  Band_wiegan = 0
  If W_cont > 25 Then Band_wiegan = 1

Return

'Interrupcion Int1 detecta entrada de UNOS
Wiegand26_1:
  Nop : Nop : Nop : Nop : Nop : Nop : Nop
  Incr W_cont

  If W_cont < 8 Then
    W1.cont_bit = 1
  Elseif W_cont < 16 Then
    W2.cont_bit = 1
  Elseif W_cont < 24 Then
    W3.cont_bit = 1
  Elseif W_cont < 27 Then
    W4.cont_bit = 1
  End If

  Decr Cont_bit
  If Cont_bit > 8 Then : Cont_bit = 7 : End If

  Band_serial = 0
  Band_wiegan = 0
  If W_cont > 25 Then Band_wiegan = 1

Return

Init_wiegand:

  Band_wiegan = 0
  W_cont = 0 : Cont_bit = 7 : W1 = 0 : W2 = 0 : W3 = 0 : W4 = 0      'inicializa variables que guardan el dato de la tarjeta id12
Return

Conexion_pc:                                  'INTERRUPCION SERIAL

 Incr Con_char_in
 Serial_char = Inkey()                         'SE ESPERA RECIBIR UN CARACTER
 Mid(buffer_mat_in , Con_char_in , 1 ) = Serial_char

 Band_serial = 0
 If Con_char_in > 14 Then Band_serial = 1

Return

Init_serial:
  Con_char_in = 0
  For For_serial = 1 To Buffer_in_long
     Mid(buffer_mat_in , For_serial , 1 ) = " "
  Next For_serial
  Band_serial = 0
Return

Verificar_horario:
```

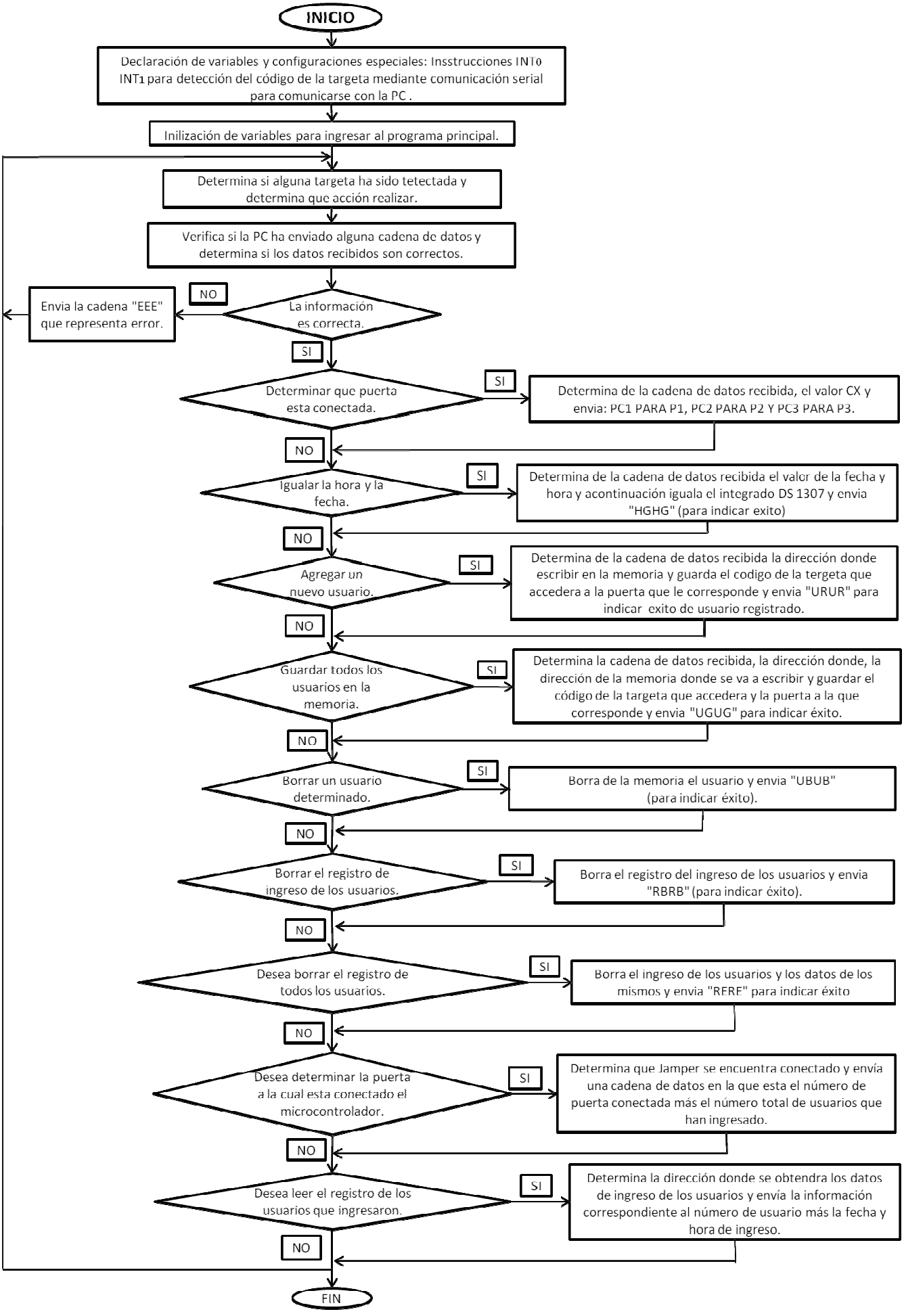
```
  'Verifica que el horario se encuentre dentro de rangos permitidos
  Gosub Leer_horario
  If _hour > 12 Then _hour = 1
  If _min > 59 Then _min = 1
  If _sec > 59 Then _sec = 1

  If _day < 1 Or _day > 31 Then _day = 1
  If _month < 1 Or _month > 12 Then _month = 1
  If _year < 11 Or _year > 30 Then _year = 11

  If _weekday < 1 Or _weekday > 7 Then _weekday = 1

  Gosub Igualar_hora
  Gosub Igualar_fecha

Return

'ESCRIBE LA  NUEVA FECHA
Igualar_fecha:
    _day = Makebcd(_day) : _month = Makebcd(_month) : _year = Makebcd(_year)
    I2cstart                          'GENERA BIT DE INICIO  I2C
    I2cwbyte Ds1307w                    'MAESTRO ENVIA LA DIRECCION DEL ESCLAVO
    I2cwbyte 4                        'DIRECCION DE INICIO------------------------------
    I2cwbyte _day
    I2cwbyte _month
    I2cwbyte _year
    I2cstop
    Waitms 10
Return

'ESCRIBE LA  NUEVA HORA
Igualar_hora:
    _sec = Makebcd(_sec) : _min = Makebcd(_min) : _hour = Makebcd(_hour)

    _hour.5 = Am_pm                     'establesco en el horario si es AM o PM   con ( Am_pm=1 then es PM   )
    _hour.6 = 1                       'establesco un formato de 12 horas

    I2cstart
    I2cwbyte Ds1307w
    I2cwbyte 0
    I2cwbyte _sec
    I2cwbyte _min
    I2cwbyte _hour
    I2cstop
    Waitms 10
Return

'LEE LA HORA Y LA FECHA
Leer_horario:
    I2cstart                          'I2C BIT DE INICIO
    I2cwbyte Ds1307w                    'MAESTRO ENVIA LA DIRECCION DEL ESCLAVO
    I2cwbyte 0                        'DIRECCION DE INICIO------------------------------
    I2cstart                          'I2C BIT DE INICIO
    I2cwbyte Ds1307r                    'MAESTRO ENVIA LA DIRECCION DEL ESCLAVO
    I2crbyte _sec , Ack                 'SEGUNDOS
    I2crbyte _min , Ack                 'MINUTOS
    I2crbyte _hour , Ack
    I2crbyte _weekday , Ack
    I2crbyte _day , Ack
    I2crbyte _month , Ack
    I2crbyte _year , Nack
    I2cstop

    Am_pm = _hour.5                     'recupero el valor de AM o PM
    _hour.7 = 0 : _hour.6 = 0 : _hour.5 = 0       'Pongo en cero estos valores para que la conversion de la hora sea la correcta

    _sec = Makedec(_sec) : _min = Makedec(_min) : _hour = Makedec(_hour)
    _day = Makedec(_day) : _month = Makedec(_month) : _year = Makedec(_year) : _weekday = Makedec(_weekday)
Return
```

```
Write_eeprom:
   I2cstart                          'start condition
   I2cwbyte Cod_esc                        'slave address
   I2cwbyte Direccion_h                     'asdress of EEPROM
   I2cwbyte Direccion_l                    'asdress of EEPROM
   I2cwbyte Dato_memoria                        'value to write
   I2cstop                           'stop condition
   Waitms 10
Return

Read_eeprom:
  I2cstart                           'generate start
  I2cwbyte Cod_esc                        'slave adsress
  I2cwbyte Direccion_h                     'address of EEPROM
  I2cwbyte Direccion_l                    'asdress of EEPROM
  I2cstart                           'repeated start
  I2cwbyte Cod_lec                        'slave address (read)
  I2crbyte Dato_memoria , Nack                 'read byte
  I2cstop                           'generate stop
Return


End
```

ANEXO 6: DIAGRAMA DE DETECCIÓN DEL CÓDIGO DEL MÓDULO ID-12



ANEXO 7: DIAGRAMA DE LA DETECCIÓN DE DATOS SERIALES



ANEXO 8: PROGRAMA FUENTE DE VISIAL BASIC

```
Imports System
Imports System.IO

Public Class Principal
    Dim texin As String
    Dim TexinAux As String
    Dim texout As String
    Dim Leer_hora_aux1 As String
    Dim Envio_US As String 'guarda el valor del usuario a grabar o eliminar del micro
    Dim Envio_LT As String 'string que pide al micro el numero de usuario que ha
ingresado
    Dim Am_Pm As String
    Dim BANDERA_INGRESOS As Byte

    Private Structure T_direciones
        Public Num_user As Byte 'guarda el numero de usuarios registrados
    End Structure

    Private Structure T_usuarios
        Public Nun_users_delete As Byte
        Public User_delete As Byte
    End Structure

    Private Structure T_nomina
        Public Numero As Byte
        <VBFixedString(30)> Public Codigo As String '* 30
        <VBFixedString(30)> Public Nombre As String '* 30
        <VBFixedString(30)> Public Cargo  As String '* 30
    End Structure

    Dim Num_tag As T_direciones
    Dim Num_user_del As T_usuarios
    Dim Nomina As T_nomina
    'variables usadas cuando se detecta un nuevo usuario
    Dim New_detec As String 'PERMITE CONOCER SI HA DETECTADO UNA NUEVA TAG
    Dim Archivo As Integer 'determina que numero de archivo abrir
    Dim Buscar_fin As Byte
    Dim Buscar_inicio As Byte

    Dim Codigo_NEW_TAG As String
    Dim Usuario_asignado_num As Byte 'determina que numero sera el asihnado al usuario
    Dim TOTAL_Users_Delete As Byte

    Dim User_num_dec As String
    Dim Band_Ing_Usuario As Byte
    Dim NumPuerta As Byte
    Dim users_borrados As Byte
    Dim total_user_registed As Byte
    '-----------------------------------------------
    Dim UsuariosRegistrados As Byte
    Dim ContFOR As Byte
    Dim Valor_dato As String
    'variables para lectura de usuarios que han ingresado
    Dim posicion_ini As Integer
    Dim num_user_reg As String
    Dim user_msb As Byte
    Dim user_lsb As Byte
    Dim T_user_ing As Byte
    Dim Cont_user_read As Byte 'contador que determina cual usuario se tiene que leer
    Dim in_user As String
```

```
    Dim in_hora As String
    Dim in_min As String
    Dim in_sec As String
    Dim in_am_pm As String
    Dim in_dia As String
    Dim in_mes As String
    Dim in_year As String
    Dim Num_txt As Byte 'detrmina en cual txt escribir segun la puerta detectada
    '---------------------variables para guardar usuarios en total
    Dim total_user_save As Byte
    Dim save_user As Byte
    Dim Band_save_user As Byte
    Dim tablafil As Integer
    Dim tablacol As Integer

    Private Sub Form1_Load(sender As System.Object, e As System.EventArgs) Handles
MyBase.Load

        'HABILITAR DESPUES
         cmdLeerP1.Enabled = False
         cmdLeerP2.Enabled = False
         cmdLeerP3.Enabled = False
         CmdConexion.Enabled = False

         Timer1.Enabled = False
         Timer2.Enabled = False
         Timer3.Enabled = False

         frmBorrar.Enabled = False
         frm_Puertas.Enabled = False
         frmInit.Enabled = False
         frmLeer.Enabled = False
         cmdSaveUsers.Enabled = False
         cmd_HF.Enabled = False

         Band_Ing_Usuario = 0
         texout = ""
         txt_Nombre.Text = ""
         txt_Cargo.Text = ""

         Tabla.RowCount = 255

         INICIALIZAR_TABLA() 'ÉSCRIBE LOS DATOS DE ENCABEZADO DE LA TABLA

         Init_tabla_datos() 'inicializa los datos de las tablas

'Dim objEscritor As IO.StreamWriter
'objEscritor = IO.File.CreateText("C:\ACCESO\CONTROL DE ACCESO\NOMINA_USER.dat")
'objEscritor.Close()
'objEscritor = IO.File.CreateText("C:\ACCESO\CONTROL DE ACCESO\USER_REGISTRADOS.dat")
'objEscritor.Close()
'objEscritor = IO.File.CreateText("C:\ACCESO\CONTROL DE ACCESO\USUARIOS_DELETE.dat")
'objEscritor.Close()

    End Sub

    Private Sub btnDeterminarConexion_Click(sender As System.Object, e As
System.EventArgs) Handles btnDeterminarConexion.Click

         cboPuertos.Items.Clear()

    For Each PuertoDisponible As String In My.Computer.Ports.SerialPortNames
            cboPuertos.Items.Add(PuertoDisponible)
        Next
```

```
        If cboPuertos.Items.Count > 0 Then

            cboPuertos.Text = cboPuertos.Items(0)
            MessageBox.Show("SELECCIONAR EL PUERTO A TRABAJAR")
            btnConectar.Enabled = True

        Else
            MessageBox.Show("NINGUN PUERTO ENCONTRADO")
            btnConectar.Enabled = False
            cboPuertos.Items.Clear()

            cboPuertos.Text = ("                    ")
        End If
    End Sub

    Private Sub btnConectar_Click(sender As System.Object, e As System.EventArgs)
Handles btnConectar.Click
        If btnConectar.Text = "CONECTAR" Then

            frmBorrar.Enabled = True
            frm_Puertas.Enabled = True
            frmInit.Enabled = True
            frmLeer.Enabled = True
            cmdSaveUsers.Enabled = True
            cmd_HF.Enabled = True
            CmdConexion.Enabled = True
            Timer1.Enabled = True

            Puerto.PortName = cboPuertos.Text
            Puerto.Open()
            btnConectar.Text = "DESCONECTAR"

        ElseIf btnConectar.Text = "DESCONECTAR" Then

            Timer1.Enabled = False
            frmBorrar.Enabled = False
            frm_Puertas.Enabled = False
            frmInit.Enabled = False
            frmLeer.Enabled = False
            cmdSaveUsers.Enabled = False
            cmd_HF.Enabled = False
            CmdConexion.Enabled = False
            Puerto.Close()
            btnConectar.Text = "CONECTAR"
        End If
    End Sub

    Private Sub cmd_HF_Click(sender As System.Object, e As System.EventArgs) Handles
cmd_HF.Click
        Dim Envio_HF As String
        Dim Leer_hora As String
        Dim Leer_hora_aux2 As String
        Dim Leer_fecha As String
        Dim Posicion As Byte

        Leer_hora = " "
        Envio_HF = "                  " 'LLENADO CON ESPACIOS PARA QUE NO DE ERROR

        Leer_hora_aux1 = Envio_HF
        Leer_hora_aux2 = Envio_HF

        '22:04:31 LA PC DEVUELVE ESTOS VALORES CON TIME
        '22/22/2011  CON DATE

        Leer_hora_aux1 = TimeString
```

```
        Leer_fecha = System.DateTime.Now

        Posicion = InStr(1, Leer_hora_aux1, ":") '22:  Posicion=3      2:  Posicion=2

        If Posicion = 2 Then 'menos de las 10
            Mid(Leer_hora_aux2, 1, 1) = "0"
            Mid(Leer_hora_aux2, 2) = Leer_hora_aux1
            Leer_hora = Leer_hora_aux2
            Am_Pm = "A"
        End If

        If Posicion = 3 Then 'mas de las 10
            Valor_dato = Mid(Leer_hora_aux1, 1, 2)
            Call Verificar_hora(Valor_dato)
            Leer_hora = Leer_hora_aux1
            'enviar a una subrutina que determine si es am o pm
        End If

        Valor_dato = Mid("HF", 1, 2) : Mid(Envio_HF, 1, 2) = Valor_dato
        Valor_dato = Mid(Leer_hora, 1, 2) : Mid(Envio_HF, 3, 2) = Valor_dato
        Valor_dato = Mid(Leer_hora, 4, 2) : Mid(Envio_HF, 5, 2) = Valor_dato
        Mid(Envio_HF, 7, 1) = Am_Pm

        Valor_dato = Mid(Leer_fecha, 1, 2) : Mid(Envio_HF, 8, 2) = Valor_dato
        Valor_dato = Mid(Leer_fecha, 4, 2) : Mid(Envio_HF, 10, 2) = Valor_dato
        Valor_dato = Mid(Leer_fecha, 9, 2) : Mid(Envio_HF, 12, 2) = Valor_dato
        Valor_dato = Mid("FIN", 1, 3) : Mid(Envio_HF, 14, 3) = Valor_dato

        texout = Envio_HF

        Puerto.DiscardOutBuffer()
        Puerto.Write(texout)
    End Sub

    Private Sub CmdConexion_Click(sender As System.Object, e As System.EventArgs)
Handles CmdConexion.Click
        texout = " CX CX CX "
        Puerto.DiscardOutBuffer()
        Puerto.Write(texout)
    End Sub

    Private Sub Timer1_Tick(sender As System.Object, e As System.EventArgs) Handles
Timer1.Tick
        texin = Puerto.ReadExisting

        If texin <> "" Then
            TexinAux = texin

        End If


        'LLENA LA STRING New_detec CON LOS TRES PRIMEROS CARACTERES QUE ENVIA EL MICRO
        'Y SEGUN ESTOS CARACTERES DETERMINO QUE OPERACION HA REALIZADO EL MICRO
        New_detec = Mid(texin, 1, 3)

        '--------------------------------------------------------------------------------
----------------------
        If New_detec = "NEW" Then 'detecta si ingreso un nuevo usuario

            Buscar_fin = InStr(texin, "FIN")

            If Buscar_fin > 0 Then
                Codigo_NEW_TAG = Mid(texin, 4, 8)
                lbl_Codigo.Text = Codigo_NEW_TAG 'leo el codigo del tag
```

```
                'lee si existe alguna direccion borrada

                Archivo = FreeFile()
                FileOpen(Archivo, "C:\ACCESO\CONTROL DE ACCESO\USUARIOS_DELETE.dat",
OpenMode.Random)
                FileGet(Archivo, Num_user_del, 1)
                FileClose(Archivo)

                users_borrados = Num_user_del.Nun_users_delete

                If users_borrados > 0 Then
                    'leo la direccion mas superior que contiene el numero de la
direccion disponible
                    Archivo = FreeFile() ' Número de archivo libre
                    FileOpen(Archivo, "C:\ACCESO\CONTROL DE ACCESO\USUARIOS_DELETE.dat",
OpenMode.Random)
                    FileGet(Archivo, Num_user_del, users_borrados + 1)
                    Usuario_asignado_num = Num_user_del.User_delete
                    FileClose(Archivo)
                    lbl_NumUsuarioNew.Text = Usuario_asignado_num

                Else 'aumento el numero de usuarios

                    Archivo = FreeFile() ' Número de archivo libre
                    FileOpen(Archivo, "C:\ACCESO\CONTROL DE
ACCESO\USER_REGISTRADOS.dat", OpenMode.Random)
                    FileGet(Archivo, Num_tag, 1)
                    FileClose(Archivo)

                    lbl_NumUsuarioNew.Text = (Num_tag.Num_user) + 1

                End If

                Band_Ing_Usuario = 1
                Beep()
                MsgBox("NUEVO USUARIO - ENCONTRADO")

            Else
                Beep()
                MsgBox("ERROR DE LECTURA DE CODIGO - INTENTE DENUEVO")
                Band_Ing_Usuario = 0
            End If

        End If

        '-------------------------------------------------------------------------------
----------------------
        If New_detec = "EEE" Then 'DETECTA ERRORES
            Beep()
            MsgBox("MECANISMO REMORO ERROR - ENVIA MENSAJE NUEVAMENTE")
            Band_Ing_Usuario = 0
        End If
        '-------------------------------------------------------------------------------
----------------------
        If New_detec = "HHH" Then 'DETECTA SI SE GURDO LA HORA
            Beep()
            MsgBox("HORA GUARDADA EXITOSAMENTE")
            Band_Ing_Usuario = 0
        End If
        '-------------------------------------------------------------------------------
----------------------
        If New_detec = "XXX" Then 'DETERMINA SI ESTA CONECTADO EL MICRO A AL PUERTA 1
            Beep()
            MsgBox("PUERTA 1 HABILITADA")
        End If
```

```
        '-------------------------------------------------------------------------------
----------------------
        If New_detec = "YYY" Then 'DETERMINA SI ESTA CONECTADO EL MICRO A AL PUERTA 1
            Beep()
            MsgBox("PUERTA 2 HABILITADA")
        End If

        '-------------------------------------------------------------------------------
----------------------
        If New_detec = "ZZZ" Then 'DETERMINA SI ESTA CONECTADO EL MICRO A AL PUERTA 1
            Beep()
            MsgBox("PUERTA 3 HABILITADA")
        End If

        '-------------------------------------------------------------------------------
----------------------

        If New_detec = "111" Then 'DETERMINA SI ESTA CONECTADO EL MICRO A AL PUERTA 1
            Beep()
            MsgBox("PUERTA 1 HABILITADA")

            Usuarios_q_ingresaron()

            If T_user_ing > 0 Then
                cmdLeerP1.Enabled = True
                cmdLeerP2.Enabled = False
                cmdLeerP3.Enabled = False
                Timer1.Enabled = False

            Else
                MsgBox("Ningun Usuario a Ingresado")
            End If

            Band_Ing_Usuario = 0
        End If
        '-------------------------------------------------------------------------------
----------------------
        If New_detec = "222" Then 'DETERMINA SI ESTA CONECTADO EL MICRO A AL PUERTA 2
            Beep()
            MsgBox("PUERTA 2 HABILITADA")
            Usuarios_q_ingresaron()

            If T_user_ing > 0 Then
                cmdLeerP1.Enabled = False
                cmdLeerP2.Enabled = True
                cmdLeerP3.Enabled = False
                Timer1.Enabled = False

            Else
                MsgBox("Ningun Usuario a ingresado")
            End If

            Band_Ing_Usuario = 0
        End If
        '-------------------------------------------------------------------------------
----------------------
        If New_detec = "333" Then 'DETERMINA SI ESTA CONECTADO EL MICRO A AL PUERTA 3
            Beep()
            MsgBox("PUERTA 3 HABILITADA")
            Usuarios_q_ingresaron()

            If T_user_ing > 0 Then
                cmdLeerP1.Enabled = False
                cmdLeerP2.Enabled = False
```

```
                cmdLeerP3.Enabled = True
                Timer1.Enabled = False

            Else
                MsgBox("Ningun Usuario a ingresado")
            End If

            Band_Ing_Usuario = 0
        End If
        '-------------------------------------------------------------------------------
----------------------


        If New_detec = "UUU" Then 'DETECTA SI SE GUARDO UN USUARIO

            'lee si existe alguna direccion borrada
            Archivo = FreeFile()
            FileOpen(Archivo, "C:\ACCESO\CONTROL DE ACCESO\USUARIOS_DELETE.dat",
OpenMode.Random)
            FileGet(Archivo, Num_user_del, 1)
            FileClose(Archivo)

            users_borrados = Num_user_del.Nun_users_delete

            If users_borrados > 0 Then

                'leo la direccion mas superior que contiene el numero de la direccion
disponible
                Archivo = FreeFile()
                FileOpen(Archivo, "C:\ACCESO\CONTROL DE ACCESO\USUARIOS_DELETE.dat",
OpenMode.Random)
                FileGet(Archivo, Num_user_del, 1 + users_borrados)
                Usuario_asignado_num = Num_user_del.User_delete
                FileClose(Archivo)

                'disminuyo en una direccion el numero de usuarios borrados y escribo

                Num_user_del.User_delete = 0
                Num_user_del.Nun_users_delete = users_borrados - 1

                Archivo = FreeFile()
                FileOpen(Archivo, "C:\ACCESO\CONTROL DE ACCESO\USUARIOS_DELETE.dat",
OpenMode.Random)
                FilePut(Archivo, Num_user_del, users_borrados)
                FileClose(Archivo)

            End If

            Beep()
            MsgBox("USUARIO REGISTRADO EXITOSAMENTE")
            Init_tabla_datos() 'inicializa los datos de las tablas
            Band_Ing_Usuario = 0
        End If
        '-------------------------------------------------------------------------------
----------------------
        If New_detec = "MMM" Then 'DETECTA SI SE INICIALIZO LA CUENTA DE LOS USUARIOS
            Beep()
            MsgBox("CUENTA DE USUARIOS INICIALIZADA")
            Band_Ing_Usuario = 0

            'VACIA EL CONTENIDO DE TODAS LAS TABLAS (PUERTAS)DE INGRESO DE USUARIOS
            Dim objEscritor As IO.StreamWriter
            objEscritor = IO.File.CreateText("C:\ACCESO\CONTROL DE
ACCESO\Puerta1_DATOS.txt")
            objEscritor.Close()
```

```
            objEscritor = IO.File.CreateText("C:\ACCESO\CONTROL DE
ACCESO\Puerta2_DATOS.txt")
            objEscritor.Close()
            objEscritor = IO.File.CreateText("C:\ACCESO\CONTROL DE
ACCESO\Puerta3_DATOS.txt")
            objEscritor.Close()

        End If

        '-------------------------------------------------------------------------------
----------------------

        If New_detec = "DDD" Then 'DETECTA SI SE ELIMINO TODOS LOS USUARIOS
            Beep()
            MsgBox("ELIMINADOS TODOS LOS REGISTROS DE LOS USUARIOS")
            Band_Ing_Usuario = 0

            'hace que el numero de usuarios registrados vuelva a ser cer0
            Archivo = FreeFile() ' Número de archivo libre
            Num_tag.Num_user = 0
            FileOpen(Archivo, "C:\ACCESO\CONTROL DE ACCESO\USER_REGISTRADOS.dat",
OpenMode.Random)
            FilePut(Archivo, Num_tag, 1)
            FileClose(Archivo)


            Num_user_del.Nun_users_delete = 0
            Num_user_del.User_delete = 0
            Archivo = FreeFile()
            FileOpen(Archivo, "C:\ACCESO\CONTROL DE ACCESO\USUARIOS_DELETE.dat",
OpenMode.Random)
            FilePut(Archivo, Num_user_del, 1)
            FileClose(Archivo)

            Tabla.Rows.Clear()
            Tabla.RowCount = 255
            INICIALIZAR_TABLA()


            'VACIA EL CONTENIDO DE TODAS LAS TABLAS (PUERTAS)DE INGRESO DE USUARIOS
            Dim objEscritor As IO.StreamWriter
            objEscritor = IO.File.CreateText("C:\ACCESO\CONTROL DE
ACCESO\Puerta1_DATOS.txt")
            objEscritor.Close()
            objEscritor = IO.File.CreateText("C:\ACCESO\CONTROL DE
ACCESO\Puerta2_DATOS.txt")
            objEscritor.Close()
            objEscritor = IO.File.CreateText("C:\ACCESO\CONTROL DE
ACCESO\Puerta3_DATOS.txt")
            objEscritor.Close()

        End If

        '-------------------------------------------------------------------------------
----------------------

        If New_detec = "BBB" Then 'DETECTA SI SE BORRO UN USUARIO
            Beep()
            MsgBox("USUARIO BORRADO EXITOSAMENTE")
            Band_Ing_Usuario = 0

            Dim valor_int As Integer
            'Open "NOMINA_USER.dat" For Random As #Archivo Len = Len(Nomina)
            'Put #1, txtNumero.Text, Nomina
            'Close #Archivo
```

```
            valor_int = CInt(txtNumero.Text)

            Nomina.Numero = 0
            Nomina.Cargo = "** USER **"
            Nomina.Nombre = "BORRADO"
            Nomina.Codigo = "00000000"

            Archivo = FreeFile()
            FileOpen(Archivo, "C:\ACCESO\CONTROL DE ACCESO\NOMINA_USER.dat",
OpenMode.Random)
            FilePut(Archivo, Nomina, valor_int)
            FileClose(Archivo)

            'lee cuantos usuario han sido borrados
            Archivo = FreeFile()
            FileOpen(Archivo, "C:\ACCESO\CONTROL DE ACCESO\USUARIOS_DELETE.dat",
OpenMode.Random)
            FileGet(Archivo, Num_user_del, 1)
            FileClose(Archivo)

            TOTAL_Users_Delete = Num_user_del.Nun_users_delete
            TOTAL_Users_Delete = TOTAL_Users_Delete + 1

            'escribo el numero total de usuarios borrados

            Num_user_del.User_delete = 0
            Num_user_del.Nun_users_delete = TOTAL_Users_Delete

            Archivo = FreeFile()
            FileOpen(Archivo, "C:\ACCESO\CONTROL DE ACCESO\USUARIOS_DELETE.dat",
OpenMode.Random)
            FilePut(Archivo, Num_user_del, 1)
            FileClose(Archivo)

            'escribo el usuario que ha sido borrado

            Num_user_del.User_delete = CInt(txtNumero.Text)
            Num_user_del.Nun_users_delete = 0

            Archivo = FreeFile()
            FileOpen(Archivo, "C:\ACCESO\CONTROL DE ACCESO\USUARIOS_DELETE.dat",
OpenMode.Random)
            FilePut(Archivo, Num_user_del, 1 + TOTAL_Users_Delete)
            FileClose(Archivo)

            Init_tabla_datos() 'inicializa los datos de las tablas
            txtNumero.Text = ""

        End If

    End Sub

    Sub Verificar_hora(hora As String)

        If hora = "10" Then
            Mid(Leer_hora_aux1, 1, 2) = "10" : Am_Pm = "A"
        End If

        If hora = "11" Then
            Mid(Leer_hora_aux1, 1, 2) = "11" : Am_Pm = "A"
        End If

        If hora = "12" Then
            Mid(Leer_hora_aux1, 1, 2) = "12" : Am_Pm = "P"
        End If
```

```
    If hora = "13" Then
        Mid(Leer_hora_aux1, 1, 2) = "01" : Am_Pm = "P"
    End If

    If hora = "14" Then
        Mid(Leer_hora_aux1, 1, 2) = "02" : Am_Pm = "P"
    End If
    If hora = "15" Then
        Mid(Leer_hora_aux1, 1, 2) = "03" : Am_Pm = "P"
    End If

    If hora = "16" Then
        Mid(Leer_hora_aux1, 1, 2) = "04" : Am_Pm = "P"
    End If

    If hora = "17" Then
        Mid(Leer_hora_aux1, 1, 2) = "05" : Am_Pm = "P"
    End If

    If hora = "18" Then
        Mid(Leer_hora_aux1, 1, 2) = "06" : Am_Pm = "P"
    End If

    If hora = "19" Then
        Mid(Leer_hora_aux1, 1, 2) = "07" : Am_Pm = "P"
    End If

    If hora = "20" Then
        Mid(Leer_hora_aux1, 1, 2) = "08" : Am_Pm = "P"
    End If

    If hora = "21" Then
        Mid(Leer_hora_aux1, 1, 2) = "09" : Am_Pm = "P"
    End If

    If hora = "22" Then
        Mid(Leer_hora_aux1, 1, 2) = "10" : Am_Pm = "P"
    End If

    If hora = "23" Then
        Mid(Leer_hora_aux1, 1, 2) = "11" : Am_Pm = "P"
    End If

End Sub


Private Sub cmdGuardarMicro_Click(sender As System.Object, e As System.EventArgs)

End Sub

Public Sub INICIALIZAR_TABLA()

    Dim ContFOR1 As Byte

    For ContFOR1 = 0 To 254
        Tabla.Item(0, ContFOR1).Value = ContFOR1 + 1
    Next ContFOR1

End Sub

Public Sub Init_tabla_datos()

    Dim ContFOR2 As Byte
```

```
        lbl_Codigo.Text = ""
        lbl_NumUsuarioNew.Text = ""
        txt_Nombre.Text = ""
        txt_Cargo.Text = ""
        Envio_US = "                    "
        opt_puerta1.Checked = False
        opt_puerta2.Checked = False
        opt_puerta3.Checked = False

        Archivo = FreeFile()
        FileOpen(Archivo, "C:\ACCESO\CONTROL DE ACCESO\USER_REGISTRADOS.dat",
OpenMode.Random)
        FileGet(Archivo, Num_tag, 1)
        FileClose(Archivo)
        UsuariosRegistrados = Num_tag.Num_user

        Archivo = FreeFile()
        FileOpen(Archivo, "C:\ACCESO\CONTROL DE ACCESO\NOMINA_USER.dat",
OpenMode.Random)

        For ContFOR2 = 1 To UsuariosRegistrados
            FileGet(Archivo, Nomina, ContFOR2)

            tablacol = CStr(1) : tablafil = CStr(ContFOR2 - 1) : Tabla.Item(tablacol,
tablafil).Value = CStr(Nomina.Cargo)
            tablacol = CStr(2) : tablafil = CStr(ContFOR2 - 1) : Tabla.Item(tablacol,
tablafil).Value = CStr(Nomina.Nombre)
            tablacol = CStr(3) : tablafil = CStr(ContFOR2 - 1) : Tabla.Item(tablacol,
tablafil).Value = CStr(Nomina.Numero)
            tablacol = CStr(4) : tablafil = CStr(ContFOR2 - 1) : Tabla.Item(tablacol,
tablafil).Value = CStr(Nomina.Codigo)
        Next ContFOR2

        FileClose(Archivo)

    End Sub


    Private Sub SalirToolStripMenuItem_Click(sender As System.Object, e As
System.EventArgs) Handles SalirToolStripMenuItem.Click
        End
    End Sub

    Private Sub Puerta1ToolStripMenuItem_Click(sender As System.Object, e As
System.EventArgs) Handles Puerta1ToolStripMenuItem.Click
        'Puerta_1.ShowDialog()
        Puerta_1.Show()
    End Sub

    Private Sub Puerta2ToolStripMenuItem_Click(sender As System.Object, e As
System.EventArgs) Handles Puerta2ToolStripMenuItem.Click
        Puerta_2.Show()
    End Sub

    Private Sub Puerta3ToolStripMenuItem_Click(sender As System.Object, e As
System.EventArgs) Handles Puerta3ToolStripMenuItem.Click
        Puerta_3.Show()
    End Sub

    Private Sub cmdInitMicro_Click(sender As System.Object, e As System.EventArgs)
Handles cmdInitMicro.Click


        'PERMITE QUE LA NOMINA DE INGRESO DE USUARIOS SE INICIALIZE
        'BORRA EL REGISTRO DE USUARIOS QUE HAN INGRESADO
```

```
        'TANTO EN LA PC COMO EN EL MICROCONTROLADOR
        texout = " BM BM BM "
        Puerto.DiscardOutBuffer()
        Puerto.Write(texout)

    End Sub

    Private Sub cmdInitMiUser_Click(sender As System.Object, e As System.EventArgs)
Handles cmdInitMiUser.Click
        'BORRA EL REGISTRO DE USUARIOS Y TAMBIEN LOS VALORES DE  LOS TXT DE INGRESO
        'TANTO EN LA PC COMO EN EL MICROCONTROLADOR
        Dim A As MsgBoxResult

        A = MsgBox("DESEA ELIMINAR TODOS LOS USUARIOS", MsgBoxStyle.YesNo, "USUARIOS")

        If A = MsgBoxResult.Yes Then
            texout = " DM DM DM "
            Puerto.DiscardOutBuffer()
            Puerto.Write(texout)
        End If

        'If A = MsgBoxResult.No Then
        'MsgBox("You clicked no")
        'End If
    End Sub

    Private Sub cmdIdentificar_Click(sender As System.Object, e As System.EventArgs)
Handles cmdIdentificar.Click
        Dim INDEX As Integer
        Dim ESPERA As Integer

        Timer1.Enabled = True
        Timer2.Enabled = False

        texout = " ID ID ID "
        Puerto.DiscardOutBuffer()

        Puerto.Write(texout)

        ESPERA = 0
        For INDEX = 1 To 40000
            ESPERA = ESPERA + 1
        Next

        Puerto.Write(texout)
    End Sub


    Private Sub cmdGuardarMicro_Click_1(sender As System.Object, e As System.EventArgs)
Handles cmdGuardarMicro.Click

        If opt_puerta1.Checked = False And opt_puerta2.Checked = False And
opt_puerta3.Checked = False Then
            Beep()
            MsgBox("SELECCIONE UNA PUERTA")
            GoTo Ninguna_accion
        End If

        If txt_Nombre.Text = "" Then
            Beep()
            MsgBox("INGRESE NOMBRE")
            GoTo Ninguna_accion
        End If

        If txt_Cargo.Text = "" Then
```

```
            Beep()
            MsgBox("INGRESE CARGO")
            GoTo Ninguna_accion
        End If

        If Band_Ing_Usuario = 1 Then

            Envio_US = "                    " 'LLENADO CON ESPACIOS PARA QUE NO DE ERROR
            User_num_dec = "  "

            'leo la cantidad de usuarios registrados
            Archivo = FreeFile() ' Número de archivo libre
            FileOpen(Archivo, "C:\ACCESO\CONTROL DE ACCESO\USER_REGISTRADOS.dat",
OpenMode.Random)
            FileGet(Archivo, Num_tag, 1)
            FileClose(Archivo)

            total_user_registed = Num_tag.Num_user
            'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
            Valor_dato = Mid("US", 1, 2) 'cargo Us en valor
            Mid(Envio_US, 1, 2) = Valor_dato 'Pongo lo de valor en la strin principal
            'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
            Archivo = FreeFile() ' Número de archivo libre

            If Band_Ing_Usuario = 1 Then
                'detrmina si incrementar el numero de usuario y en caso de que haya uno
                'o varios borrados deve de asignar una direccion borrada
                'para que los datos de el usuario se guarden en esta direccion
                'lee si existe alguna direccion borrada

                Archivo = FreeFile()
                FileOpen(Archivo, "C:\ACCESO\CONTROL DE ACCESO\USUARIOS_DELETE.dat",
OpenMode.Random)
                FileGet(Archivo, Num_user_del, 1)
                FileClose(Archivo)

                users_borrados = Num_user_del.Nun_users_delete

                'determino la direccion donde se escribira el dato del usuario
                If users_borrados > 0 Then
                    'leo la direccion mas superior que contiene el numero de la
direccion disponible


                    Archivo = FreeFile()
                    FileOpen(Archivo, "C:\ACCESO\CONTROL DE ACCESO\USUARIOS_DELETE.dat",
OpenMode.Random)
                    FileGet(Archivo, Num_user_del, 1 + users_borrados)
                    Usuario_asignado_num = Num_user_del.User_delete
                    FileClose(Archivo)

                Else 'aumento el numero de usuarios

                    Archivo = FreeFile() ' Número de archivo libre
                    FileOpen(Archivo, "C:\ACCESO\CONTROL DE
ACCESO\USER_REGISTRADOS.dat", OpenMode.Random)
                    FileGet(Archivo, Num_tag, 1)
                    FileClose(Archivo)

                    Usuario_asignado_num = (Num_tag.Num_user) + 1
                    total_user_registed = Usuario_asignado_num
                End If
            End If
            'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
            If total_user_registed < 10 Then 'determino la cantidad total de usuarios
```

```
                Decimal_user(total_user_registed) 'transformo el dato a string
hexadecimal
            Else
                User_num_dec = total_user_registed
            End If
            Mid(Envio_US, 3, 2) = User_num_dec
            'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
            If Usuario_asignado_num < 10 Then 'determino la direccion donde se guardara
el usuario en el microcontrolador
                Decimal_user(Usuario_asignado_num)
            Else
                User_num_dec = Usuario_asignado_num
            End If
            Mid(Envio_US, 5, 2) = User_num_dec 'direccion donde se escribira el codigo
del usuario
            'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
            If opt_puerta1.Checked = True Then 'determino el numero de puerta a incluir
en la string
                Mid(Envio_US, 7, 2) = "01"
                NumPuerta = 1
            End If

            If opt_puerta2.Checked = True Then
                Mid(Envio_US, 7, 2) = "02"
                NumPuerta = 2
            End If

            If opt_puerta3.Checked = True Then
                Mid(Envio_US, 7, 2) = "03"
                NumPuerta = 3
            End If
            'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
            Mid(Envio_US, 9, 8) = Codigo_NEW_TAG 'escribo el codigo del tag
            Mid(Envio_US, 17, 3) = "FIN"

            'GUARDO EL NUMERO TOTAL DE USUARIOS
            Archivo = FreeFile()
            Num_tag.Num_user = total_user_registed
            FileOpen(Archivo, "C:\ACCESO\CONTROL DE ACCESO\USER_REGISTRADOS.dat",
OpenMode.Random)
            FilePut(Archivo, Num_tag, 1)
            FileClose(Archivo)
            'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
            'GUARDO LA INFORMACION DEL USUARIO

            Nomina.Numero = NumPuerta
            Nomina.Cargo = txt_Cargo.Text
            Nomina.Nombre = txt_Nombre.Text
            Nomina.Codigo = Codigo_NEW_TAG

            Archivo = FreeFile()
            FileOpen(Archivo, "C:\ACCESO\CONTROL DE ACCESO\NOMINA_USER.dat",
OpenMode.Random)
            FilePut(Archivo, Nomina, Usuario_asignado_num)
            FileClose(Archivo)
            'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
            Band_Ing_Usuario = 0
        End If

        texout = ""
        texout = Envio_US
        Puerto.DiscardOutBuffer()
        Puerto.Write(texout)

Ninguna_accion:
```

```
End Sub

Sub Decimal_user(User As Byte)

    If User = 0 Then
        User_num_dec = "00"
    End If

    If User = 1 Then
        User_num_dec = "01"
    End If

    If User = 2 Then
        User_num_dec = "02"
    End If

    If User = 3 Then
        User_num_dec = "03"
    End If

    If User = 4 Then
        User_num_dec = "04"
    End If

    If User = 5 Then
        User_num_dec = "05"
    End If

    If User = 6 Then
        User_num_dec = "06"
    End If

    If User = 7 Then
        User_num_dec = "07"
    End If

    If User = 8 Then
        User_num_dec = "08"
    End If

    If User = 9 Then
        User_num_dec = "09"
    End If

End Sub

Public Sub Usuarios_q_ingresaron()

    Dim string_msb As String
    Dim string_lsb As String

    user_lsb = 0
    user_msb = 0
    num_user_reg = "  "
    'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx
    posicion_ini = InStr(TexinAux, "AA")
    num_user_reg = Mid(TexinAux, posicion_ini + 2, 2)

    string_msb = Mid(num_user_reg, 1, 1)
    string_lsb = Mid(num_user_reg, 2, 1)

    If string_msb = "0" Then user_msb = 0
    If string_msb = "1" Then user_msb = 1
    If string_msb = "2" Then user_msb = 2
```

```
        If string_msb = "3" Then user_msb = 3
        If string_msb = "4" Then user_msb = 4
        If string_msb = "5" Then user_msb = 5
        If string_msb = "6" Then user_msb = 6
        If string_msb = "7" Then user_msb = 7
        If string_msb = "8" Then user_msb = 8
        If string_msb = "9" Then user_msb = 9

        If string_lsb = "0" Then user_lsb = 0
        If string_lsb = "1" Then user_lsb = 1
        If string_lsb = "2" Then user_lsb = 2
        If string_lsb = "3" Then user_lsb = 3
        If string_lsb = "4" Then user_lsb = 4
        If string_lsb = "5" Then user_lsb = 5
        If string_lsb = "6" Then user_lsb = 6
        If string_lsb = "7" Then user_lsb = 7
        If string_lsb = "8" Then user_lsb = 8
        If string_lsb = "9" Then user_lsb = 9

        T_user_ing = user_msb * 10
        T_user_ing = T_user_ing + user_lsb

        'xxxx en caso de error de lectura xxxxx
        'todo esto en vez de lo anterior
        'posicion_ini = InStr(texin, "AA")
        'num_user_reg = Mid(texin, posicion_ini + 2, 2)
        'T_user_ing = num_user_reg
        'xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx

    End Sub

    Private Sub cmdLeerP1_Click(sender As System.Object, e As System.EventArgs) Handles
cmdLeerP1.Click
        BANDERA_INGRESOS = 1
        Cont_user_read = 0
        Num_txt = 1
        Leer_Num_Ingresos()
         Timer2.Enabled = True

    End Sub

    Private Sub cmdLeerP2_Click(sender As System.Object, e As System.EventArgs) Handles
cmdLeerP2.Click
        BANDERA_INGRESOS = 1
        Cont_user_read = 0
        Num_txt = 2
        Leer_Num_Ingresos()
        Timer2.Enabled = True
    End Sub

    Private Sub cmdLeerP3_Click(sender As System.Object, e As System.EventArgs) Handles
cmdLeerP3.Click
        BANDERA_INGRESOS = 1
        Cont_user_read = 0
        Num_txt = 3
        Leer_Num_Ingresos()
        Timer2.Enabled = True
    End Sub

    Private Sub Timer2_Tick(sender As System.Object, e As System.EventArgs) Handles
Timer2.Tick
        texin = Puerto.ReadExisting

        Buscar_inicio = InStr(texin, "US")
        Buscar_fin = InStr(texin, "XW")
```

```
        'Si encuentro US o XW en la string de entrada determino que el mensaje llego
correctamente

        If Buscar_inicio > 0 And Buscar_fin > 0 Then 'detecta si ingreso un nuevo
usuario

            Cont_user_read = Cont_user_read + 1
            Leer_Num_Ingresos()

            'una ves que ha llegado la string correctamente extaigo la informacion
            in_user = Mid(texin, Buscar_inicio + 2, 2)
            in_hora = Mid(texin, Buscar_inicio + 4, 2)
            in_min = Mid(texin, Buscar_inicio + 6, 2)
            in_sec = Mid(texin, Buscar_inicio + 8, 2)
            in_am_pm = Mid(texin, Buscar_inicio + 10, 2)
            in_dia = Mid(texin, Buscar_inicio + 12, 2)
            in_mes = Mid(texin, Buscar_inicio + 14, 2)
            in_year = Mid(texin, Buscar_inicio + 16, 2)

            texin = ""
            'A continuacion dependiendo del tipo de puerta escribo en el txt
correspondiente
            If Num_txt = 1 Then

                Dim objEscritor As IO.StreamWriter
                'abrir el archivo
                objEscritor = IO.File.AppendText("C:\ACCESO\CONTROL DE
ACCESO\Puerta1_DATOS.txt")
                'escribes el dato
                objEscritor.WriteLine("El usuario   " & in_user & "   paso a la hora   "
& in_hora & ":" & in_min & ":" & in_sec & " " & in_am_pm & "   del   " & in_dia & "/" &
in_mes & "/" & in_year)
                'cierra el archico
                objEscritor.Close()

            End If

            If Num_txt = 2 Then
                Dim objEscritor As IO.StreamWriter
                objEscritor = IO.File.AppendText("C:\ACCESO\CONTROL DE
ACCESO\Puerta2_DATOS.txt")
                objEscritor.WriteLine("El usuario   " & in_user & "   paso a la hora   "
& in_hora & ":" & in_min & ":" & in_sec & " " & in_am_pm & "   del   " & in_dia & "/" &
in_mes & "/" & in_year)
                objEscritor.Close()
            End If

            If Num_txt = 3 Then
                Dim objEscritor As IO.StreamWriter
                objEscritor = IO.File.AppendText("C:\ACCESO\CONTROL DE
ACCESO\Puerta3_DATOS.txt")
                objEscritor.WriteLine("El usuario   " & in_user & "   paso a la hora   "
& in_hora & ":" & in_min & ":" & in_sec & " " & in_am_pm & "   del   " & in_dia & "/" &
in_mes & "/" & in_year)
                objEscritor.Close()
            End If

            If Cont_user_read > T_user_ing - 1 Then

                Beep()

                cmdLeerP1.Enabled = False
                cmdLeerP2.Enabled = False
                cmdLeerP3.Enabled = False
```

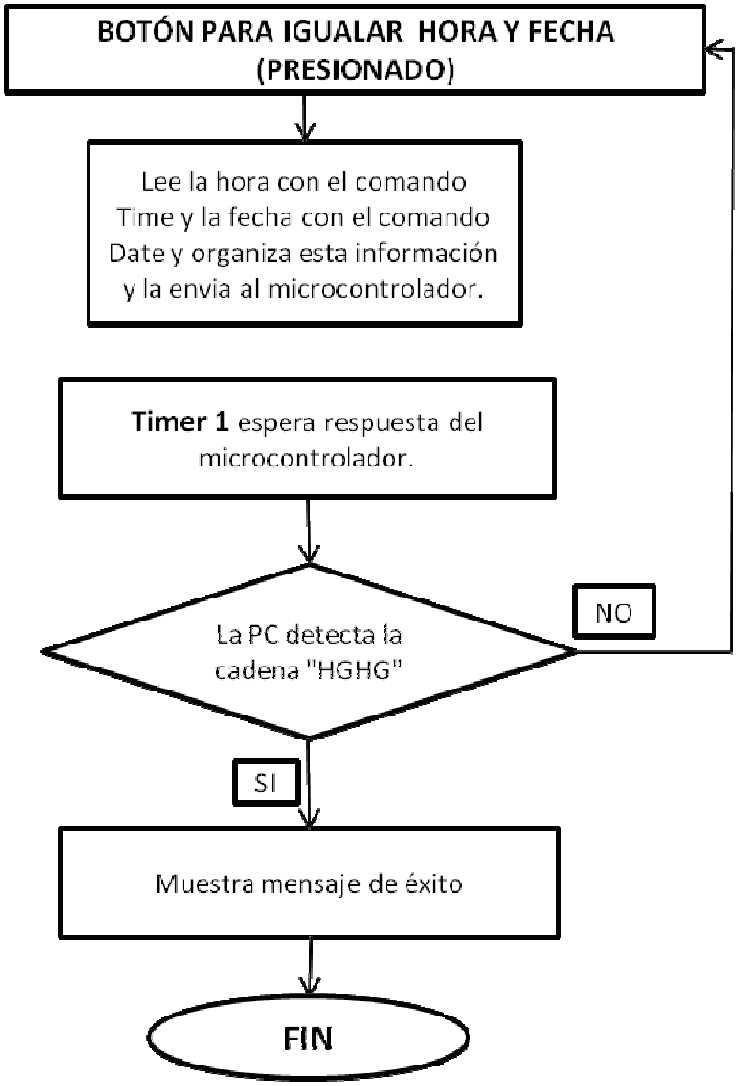
```
                Timer2.Enabled = False
                Timer1.Enabled = True

                MsgBox("LECTURA FINALIZADA")
                Puerto.DiscardOutBuffer()
                texout = ""
                Envio_LT = ""

            End If

        Else

            If Timer2.Enabled = True Then
                texout = Envio_LT
                Puerto.Write(texout)
            End If



        End If
    End Sub

    Public Sub Leer_Num_Ingresos() 'determino el numero de ingreso de usuario que se
pedira al microcontrolador

        Envio_LT = "        "
        Valor_dato = Mid("LT", 1, 2) : Mid(Envio_LT, 1, 2) = Valor_dato

        If Cont_user_read < 10 Then 'transformo el dato a string decimal
            Decimal_user(Cont_user_read)
        Else
            User_num_dec = Cont_user_read
        End If

        Valor_dato = Mid(User_num_dec, 1, 2) : Mid(Envio_LT, 3, 2) = Valor_dato
        Mid(Envio_LT, 5, 3) = "FIN"

        texout = Envio_LT
        Puerto.Write(texout)

        If BANDERA_INGRESOS = 1 Then
            MsgBox("OBTENIENDO INFORMACION DE INGRESOS")
            BANDERA_INGRESOS = 0
        End If

    End Sub


    Private Sub cmdBorrar_Click(sender As System.Object, e As System.EventArgs) Handles
cmdBorrar.Click

        If txtNumero.Text = "" Then
            Beep()
            MsgBox("SELECCIONE UN USUARIO")
        Else

            Envio_US = "                      " 'LLENADO CON ESPACIOS PARA QUE NO DE
ERROR
            User_num_dec = "   "

            Valor_dato = Mid("US", 1, 2) : Mid(Envio_US, 1, 2) = Valor_dato


            Archivo = FreeFile() ' Número de archivo libre
```

```
            FileOpen(Archivo, "C:\ACCESO\CONTROL DE ACCESO\USER_REGISTRADOS.dat",
OpenMode.Random)
            FileGet(Archivo, Num_tag, 1)
            FileClose(Archivo)

            total_user_registed = Num_tag.Num_user

            If total_user_registed < 10 Then 'transformo el dato a string decimal
                Decimal_user(total_user_registed)
            Else
                User_num_dec = total_user_registed
            End If
            Mid(Envio_US, 3, 2) = User_num_dec

            Usuario_asignado_num = txtNumero.Text
            If Usuario_asignado_num < 10 Then 'transformo el dato a string decimal
                Decimal_user(Usuario_asignado_num)
            Else
                User_num_dec = Usuario_asignado_num
            End If
            Mid(Envio_US, 5, 2) = User_num_dec

            Mid(Envio_US, 7, 2) = "00"
            Mid(Envio_US, 9, 8) = "00000000"
            Mid(Envio_US, 17, 3) = "FIN"

            texout = ""
            texout = Envio_US
            Puerto.DiscardOutBuffer()
            Puerto.Write(texout)

        End If

    End Sub

    Private Sub cmdSaveUsers_Click(sender As System.Object, e As System.EventArgs)
Handles cmdSaveUsers.Click
        Timer1.Enabled = False
        Timer3.Enabled = True
        Band_save_user = 1
        save_user = 1
    End Sub

    Private Sub Timer3_Tick(sender As System.Object, e As System.EventArgs) Handles
Timer3.Tick

        texin = Puerto.ReadExisting

        New_detec = Mid(texin, 1, 3)

        If New_detec = "SSS" Then 'detecta si ingreso un nuevo usuario

            save_user = save_user + 1
            Band_save_user = 1

            If save_user > total_user_save Then
                Beep()
                Timer1.Enabled = True
                Timer3.Enabled = False
                MsgBox("TODOS LOS USUARIOS GUARDADOS")
                GoTo NO_GUARDAR
            End If

        End If
```

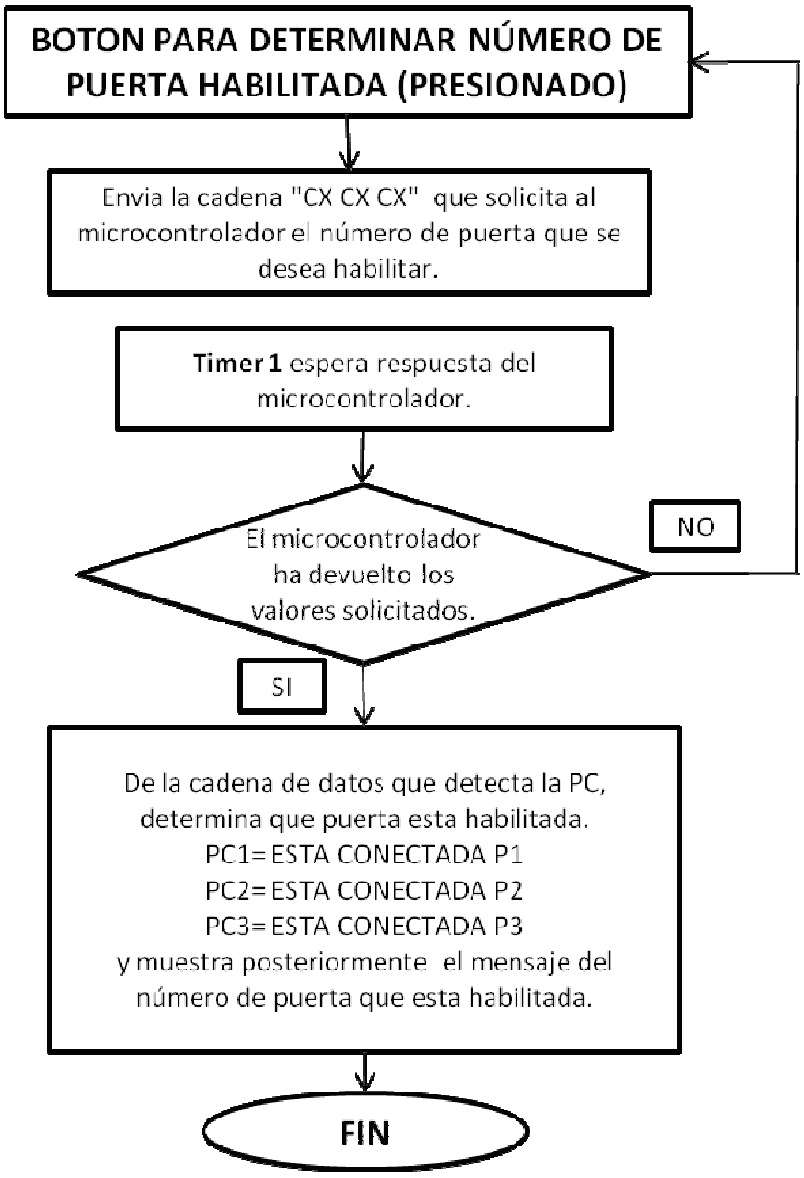
```
        If New_detec = "EEE" Then 'detecta si ingreso un nuevo usuario
            Band_save_user = 1
        End If

        If Band_save_user = 1 Then
            Envio_US = "                    " 'LLENADO CON ESPACIOS PARA QUE NO DE ERROR
            User_num_dec = "  "


            Archivo = FreeFile() ' Número de archivo libre
            FileOpen(Archivo, "C:\ACCESO\CONTROL DE ACCESO\USER_REGISTRADOS.dat",
OpenMode.Random)
            FileGet(Archivo, Num_tag, 1)
            FileClose(Archivo)


            total_user_save = Num_tag.Num_user
            Valor_dato = Mid("SV", 1, 2) : Mid(Envio_US, 1, 2) = Valor_dato

            If total_user_save < 10 Then 'transformo el dato a string hexadecimal
                Decimal_user(total_user_save)
            Else
                User_num_dec = total_user_save
            End If

            Mid(Envio_US, 3, 2) = User_num_dec 'DIRECCION QUE LLEVA EL VALOR TOTAL DE
USUARIO


            Archivo = FreeFile()
            FileOpen(Archivo, "C:\ACCESO\CONTROL DE ACCESO\NOMINA_USER.dat",
OpenMode.Random)
            FileGet(Archivo, Nomina, save_user)
            FileClose(Archivo)


            If Nomina.Numero < 10 Then 'transformo el dato a string hexadecimal
                Decimal_user(save_user)
            Else
                User_num_dec = save_user
            End If

            Mid(Envio_US, 5, 2) = User_num_dec 'direccion donde se escribira el codigo
del usuario

            If Nomina.Numero = 0 Then
                Mid(Envio_US, 7, 2) = "00"
            End If

            If Nomina.Numero = 1 Then
                Mid(Envio_US, 7, 2) = "01"
            End If

            If Nomina.Numero = 2 Then
                Mid(Envio_US, 7, 2) = "02"
            End If

            If Nomina.Numero = 3 Then
                Mid(Envio_US, 7, 2) = "03"
            End If

            Mid(Envio_US, 9, 8) = Nomina.Codigo
            Mid(Envio_US, 17, 3) = "FIN"

            texout = ""
```

```
            texout = Envio_US
            Puerto.DiscardOutBuffer()
            Puerto.Write(texout)

            Band_save_user = 0
        End If

NO_GUARDAR:
    End Sub

    Private Sub AcercaToolStripMenuItem_Click(sender As System.Object, e As
System.EventArgs) Handles AcercaToolStripMenuItem.Click
        'Puerta_1.ShowDialog()

        Acerca.ShowDialog()

        'Puerta_1.Show()
    End Sub
End Class
```

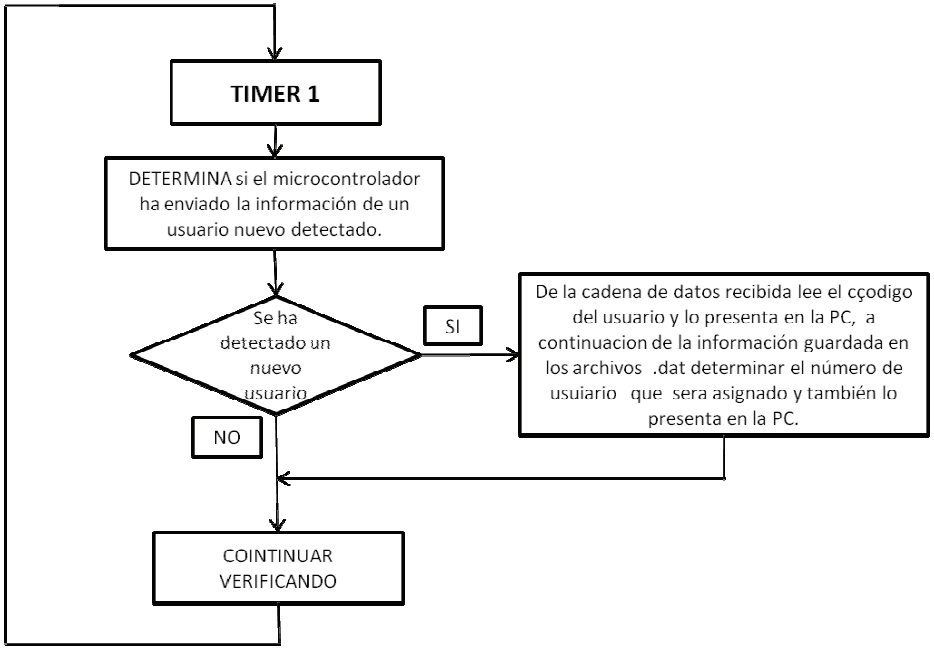
ANEXO 9: DIAGRAMA PARA GUARDAR HORARIO

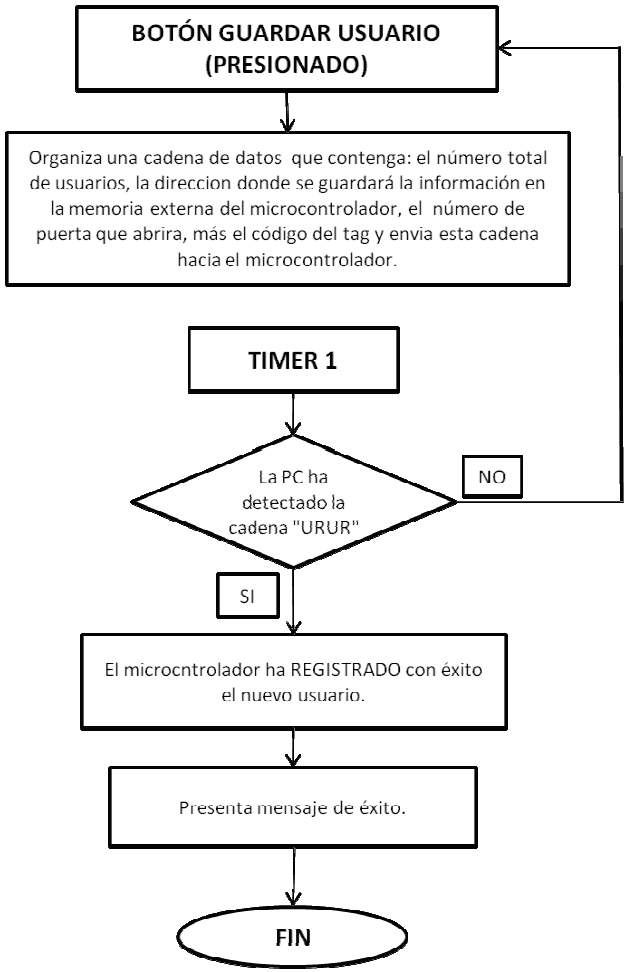


ANEXO 10: DIAGRAMA PARA DETERMINAR LA PUERTA HABILITADA

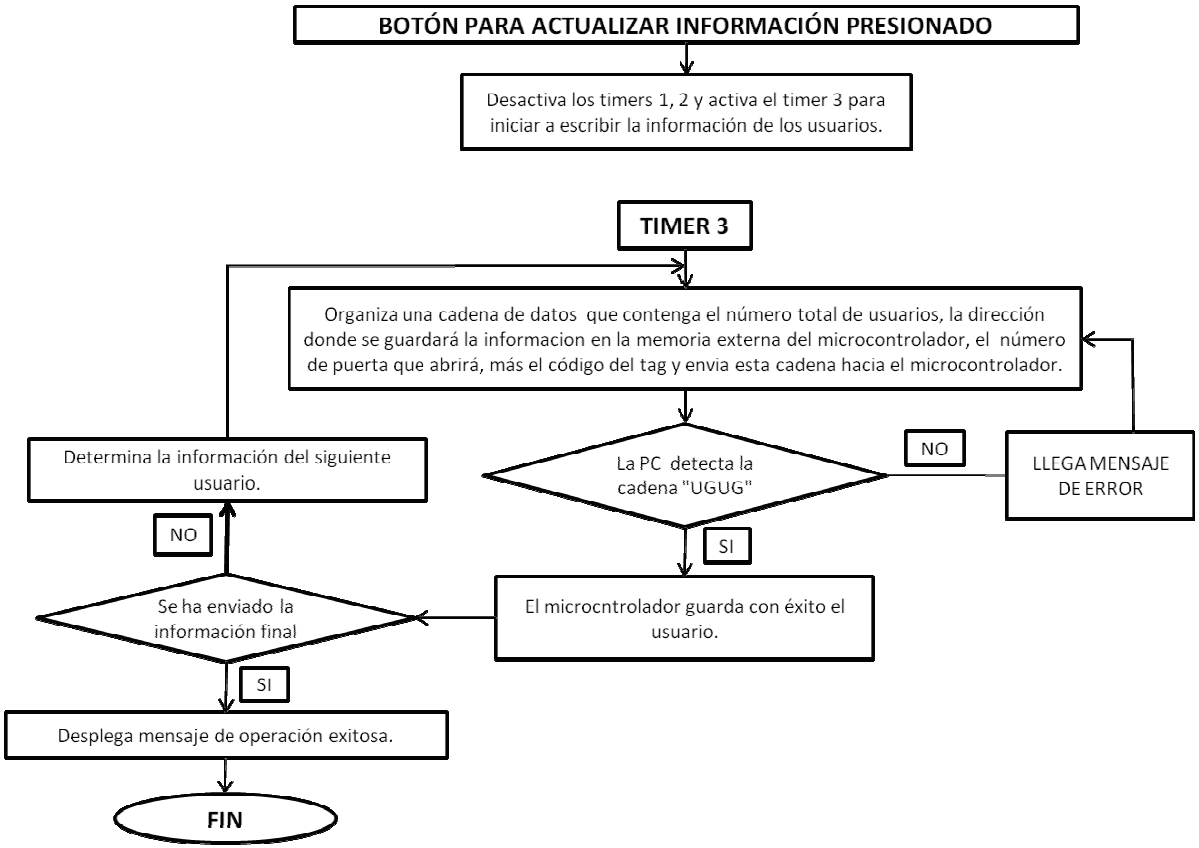


ANEXO 11: DIAGRAMA PARA REGISTRAR UN NUEVO USUARIO





ANEXO 12: DIAGRAMA PARA ACTUALIZAR INFORMACIÓN



ANEXO 13: DIAGRAMA DE LECTURA DE INGRESOS