ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

IMPLEMENTACIÓN DEL PROTOTIPO DE UN SISTEMA DE CONTROL DE ACCESO ELECTRÓNICO DE PERSONAL PARA UNA ORGANIZACIÓN.

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

ROMMEL DANILO BERMEO SANTAMARÍA
rberym@hotmail.com
HILDA MARIZOL MORENO ANDRADE
marizolconz@hotmail.com

DIRECTOR: ING. ALCÍVAR COSTALES
eduardo.costales@epn.edu.ec

QUITO, JUNIO 2011

DECLARACIÓN

Nosotros, Rommel Danilo Bermeo Santamaría e Hilda Marizol Moreno Andrade, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos nuestros derechos 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.

Rommel Bermeo S. **Hilda Moreno A.**

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Rommel Danilo Bermeo Santamaría e Hilda Marizol Moreno Andrade, bajo mi supervisión.

Ing. Alcívar Costales G.

DIRECTOR DE PROYECTO

AGRADECIMIENTO

Agradezco en primer lugar a Dios, a mi familia por todo el apoyo, comprensión brindada durante este ciclo importante en mi vida, de manera especial a mis padres por estar siempre junto a mí en todos los momentos, a todos mis hermanos por darme el ejemplo de vida, superación, solidaridad, y amor, a todos mis amigos por el cariño que se ha creado con el paso del tiempo.

Marisol

AGRADECIMIENTO

Agradezco a Dios por ser mi guía y fortaleza.

A mi abuelito Carlos, mis padres Alicia y Riquelme, mis hermanos Liliana, Mariela, Sofía, Sixto, Alisson por ser mi apoyo siempre en los momentos buenos y malos que he tenido que pasar.

Gracias a mis compañeros y amigos por compartir sus anhelos y sueños, de manera especial a Marizol por ser una amiga incondicional.

Al personal docente y administrativo de la Escuela Politécnica Nacional por haberme formado como persona y profesional.

A todas estas personas muchas gracias por haber sido una parte de esta meta.

Rommel

DEDICATORIA

Dedico este logro a mis tíos abuelitos Humberto Andrade Luna, Mariana Andrade Luna y en especial a mi abuelito Isael Andrade Luna por la bondad y la belleza de sus corazones, que aunque no estén presentes hoy siguen estando por siempre en mi mente y corazón.

Marisol

DEDICATORIA

Dedico este logro a mi abuelito Carlos Alonso Santamaría por siempre guiarme desde el principio de mi formación por el sendero del bien y haberme impulsado a sobresalir y llegar a ser alguien mejor en esta vida, a mi mamá Alicia Santamaría por su apoyo incondicional en la realización de este proyecto.

Rommel

TABLA DE CONTENIDO

RESUMEN

En el primer capítulo se define de manera clara y sencilla el concepto general de los microcontroladores, luego las características y el funcionamiento de un AVR, para después puntualizar el AVR especifico que se utilizó en este proyecto. Además se hace una descripción de los elementos que se utilizaron, se hace mención del Protocolo RS 232, que nos permitirá interactuar el circuito con un computador.

En el segundo capítulo se describe el funcionamiento del sistema, diagrama de flujo, bloques y todos los circuitos esquemáticos que conforman el proyecto.

En el tercer capítulo se detalla la implementación del circuito, los dos programas creados para el sistema del control de acceso, realizados en lenguaje BASCOM AVR para la programación del microcontrolador y Visual Basic para la comunicación del sistema con el computador. Se presenta fotografías del desarrollo y funcionamiento, las diferentes pruebas efectuadas en vivo y simulaciones del circuito.

Finalmente concluimos con un breve análisis técnico y económico, donde se determina el valor aproximado del proyecto, se detalla la lista de precios y se hace una comparación del costo total del sistema de control de acceso desarrollado por nosotros con sistemas que ya existen en el mercado.

PRESENTACIÓN

En la actualidad existen organizaciones que no poseen un sistema que controle el acceso de personal, estas podrían convertirse en un blanco potencial de robo o acceso de personas no autorizadas, varias de estas organizaciones cuentan con equipos e información importante para sus empresas u organizaciones, razón por la cual se plantea la elaboración de un sistema electrónico, de acceso que controle el ingreso solo a personas autorizadas, el mismo que en un futuro impedirá que personal no autorizado acceda a estas.

Las unidades de control están diseñadas para brindar un alto grado de seguridad, conjugando al mismo tiempo todas las ventajas del AVR Atmega 164P que es el cerebro del sistema de este proyecto, ya que controla y ordena según el programa incluido en la memoria todas la acciones. Su capacidad de almacenamiento de datos va a depender del alcance de capacidad de la memoria del microcontrolador, en este caso necesitamos de dos memorias externas para mayor alcance en el almacenamiento de información. En lo referente a seguridad, la entrada solamente será permitida al personal autorizado, mediante el nombre del Usuario y un código único e irreproducible, también cuenta con un puerto de entrada para sensores magnéticos, de manera que determine si la puerta o ventanas fueron violentadas, inmediatamente se activara un dispositivo sonoro, Así mismo un relé de alarma accionará una sirena cuando se haya detectado 3 errores al momento de ingresar la clave, la alarma sonará un tiempo determinado y se desactivara automáticamente; el equipo asentará el evento detectado, en memoria del microcontrolador AVR y será registrado en un PC que a la vez tiene una base de datos de las personas que se encuentran registradas con un ID y una contraseña. Por medio del Software de Control de Acceso se obtiene el registro de la fecha y hora de entrada de las veces que cada usuario registrado en el sistema ingreso y solamente una persona en este caso el administrador será la única persona autorizada para desactivar y bloquear claves así mismo será el encargado de emitir las claves en caso de ampliar el número de registro de personas.

CAPÍTULO 1: INFORMACIÓN TEÓRICA

1.1 MICROCONTROLADOR[1]

Un microcontrolador figura 1.1 es un circuito integrado o chip que incluye en su interior las tres unidades funcionales de una computadora: CPU, Memoria y Unidades de E/S, es decir, se trata de una computadora completa en un solo circuito integrado programable y se destina a gobernar una sola tarea con el programa que reside en su memoria. Sus líneas de entrada/salida soportan el conexionado de los sensores y actuadores del dispositivo a controlar.

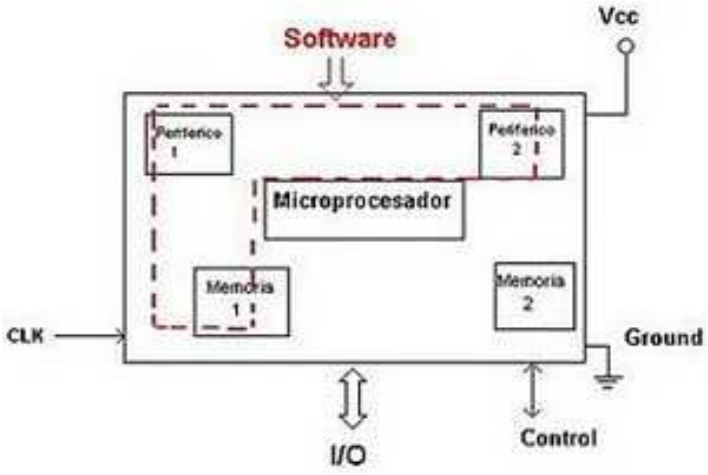


Figura 1. 1 Esquema de un microcontrolador

1.1.1 DIFERENCIAS ENTRE EL MICROPROCESADOR Y MICROCONTROLADOR.

El microprocesador es un circuito integrado que contiene la Unidad Central de Proceso (UCP), también llamada procesador, de un computador. La UCP está formada por la Unidad de Control, que interpreta las instrucciones, y el camino de datos, que las ejecuta. Los pines de un microprocesador sacan al exterior las líneas de sus buses de direcciones, datos y control, para permitir conectarle con la Memoria y los Módulos de E/S y configurar un computador implementado por varios circuitos integrados. Se dice que un microprocesador es un sistema abierto

[1] http://losmicrocontroladores.blogspot.com/

porque su configuración es variable de acuerdo con la aplicación a la que se destine.

Son diseñados para disminuir el costo económico y el consumo de energía de un sistema en particular. Por eso el tamaño de la CPU, la cantidad de memoria y los periféricos incluidos dependerán de la aplicación.

Un microcontrolador difiere de una CPU normal, debido a que es más fácil convertirla en una computadora en funcionamiento, con un mínimo de chips externos de apoyo. La idea es que el chip se coloque en el dispositivo, enganchado a la fuente de energía y de información que necesite, y eso es todo. Un microprocesador tradicional no le permitirá hacer esto, ya que espera que todas estas tareas sean manejadas por otros chips. Hay que agregarle los módulos de entrada/salida (puertos) y la memoria para almacenamiento de información.

Los microcontroladores negocian la velocidad y la flexibilidad para facilitar su uso. Debido a que se utiliza bastante sitio en el chip para incluir funcionalidad, como los dispositivos de entrada/salida o la memoria que incluye el microcontrolador, se ha de prescindir de cualquier otra circuitería.

El microprocesador es un circuito integrado que contiene la Unidad Central de Proceso (UCP), también llamada procesador, de un computador. La UCP está formada por la Unidad de Control, que interpreta las instrucciones, y el Camino de Datos, que las ejecuta.[2]

Las patitas de un microprocesador sacan al exterior las líneas de sus buses de direcciones, datos y control, para permitir conectarle con la Memoria y los Módulos de E/S y configurar un computador implementado por varios circuitos integrados. Se dice que un microprocesador es un sistema abierto porque su configuración es variable de acuerdo con la aplicación a la que se destine.

[2] http://www.monografias.com/trabajos12/microco/microco.shtml

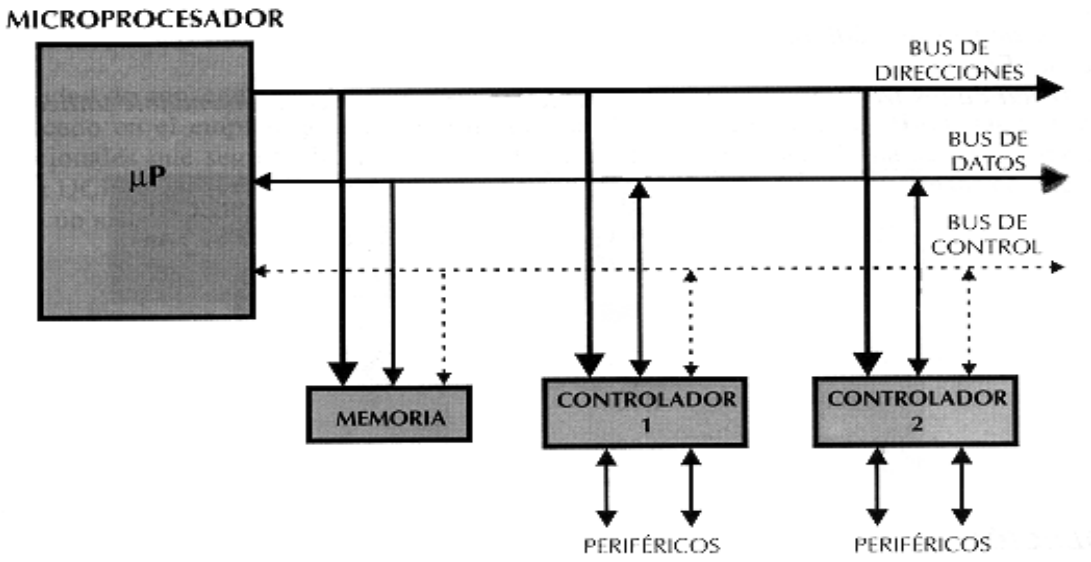


Figura 1. 2 Estructura de un sistema abierto basado en un microprocesador. La disponibilidad de los buses en el exterior permite que se configure a la medida de la aplicación.

Si sólo se dispusiese de un modelo de microcontrolador, éste debería tener muy potenciados todos sus recursos para poderse adaptar a las exigencias de las diferentes aplicaciones. Esta potenciación supondría en muchos casos un despilfarro. En la práctica cada fabricante de microcontroladores oferta un elevado número de modelos diferentes, desde los más sencillos hasta los más poderosos. Es posible seleccionar la capacidad de las memorias, el número de líneas de E/S, la cantidad y potencia de los elementos auxiliares, la velocidad de funcionamiento, etc. Por todo ello, un aspecto muy destacado del diseño es la selección del microcontrolador a utilizar.

El microcontrolador es un sistema cerrado. Todas las partes del computador están contenidas en su interior y sólo salen al exterior las líneas que gobiernan los periféricos. (Figura 1.3)

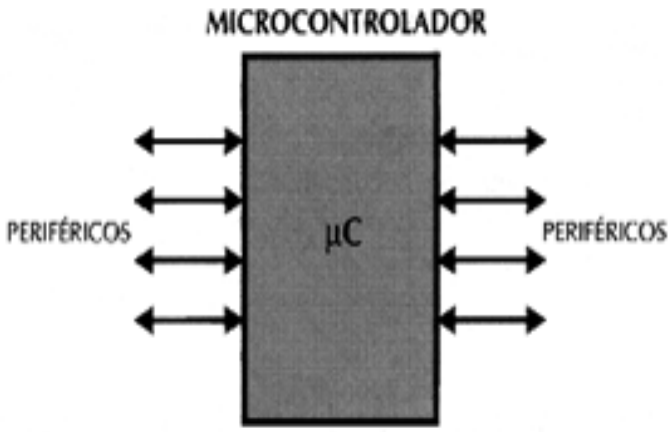


Figura 1.3 Microcontrolador y periféricos

1.1.2 RECURSOS COMUNES A TODOS LOS MICROCONTROLADORES

Al estar todos los microcontroladores integrados en un chip, su estructura fundamental y sus características básicas son muy parecidas. Todos deben disponer de los bloques esenciales Procesador, memoria de datos y de instrucciones, líneas de E/S, oscilador de reloj y módulos controladores de periféricos. Sin embargo, cada fabricante intenta enfatizar los recursos más idóneos para las aplicaciones a las que se destinan preferentemente.

1.1.3 NÚCLEO DE UN MICROCONTROLADOR

Aun cuando el microcontrolador es una computadora embebida dentro de un circuito integrado, se compone de un núcleo y un conjunto de circuitos adicionales. Dentro del núcleo se encuentran el procesador y la memoria, todo ello estructurado de forma tal que conforme una arquitectura de computadora.

1.1.4 TIPOS DE ARQUITECTURAS DE MICROCONTROLADORES

1.1.4.1 Arquitectura Von Neumann

La arquitectura tradicional de computadoras y microprocesadores está basada en la arquitectura Von Neumann figura 1. 4, en la cual la unidad central de proceso (CPU), está conectada a una memoria única donde se guardan las instrucciones del programa y los datos.

El tamaño de la unidad de datos o instrucciones está fijado por el ancho del bus que comunica la memoria con la CPU. Así un microprocesador de 8 bits con un

bus de 8 bits, tendrá que manejar datos e instrucciones de una o más unidades de 8 bits (bytes) de longitud. Si tiene que acceder a una instrucción o dato de más de un byte de longitud, tendrá que realizar más de un acceso a la memoria.

Y el tener un único bus hace que el microprocesador sea más lento en su respuesta, ya que no puede buscar en memoria una nueva instrucción mientras no finalicen las transferencias de datos de la instrucción anterior.

Resumiendo todo lo anterior, las principales limitaciones que nos encontramos con la arquitectura Von Neumann son:

- La limitación de la longitud de las instrucciones por el bus de datos, que hace que el microprocesador tenga que realizar varios accesos a memoria para buscar instrucciones complejas.
- La limitación de la velocidad de operación a causa del bus único para datos e instrucciones que no deja acceder simultáneamente a unos y otras, lo cual impide superponer ambos tiempos de acceso.

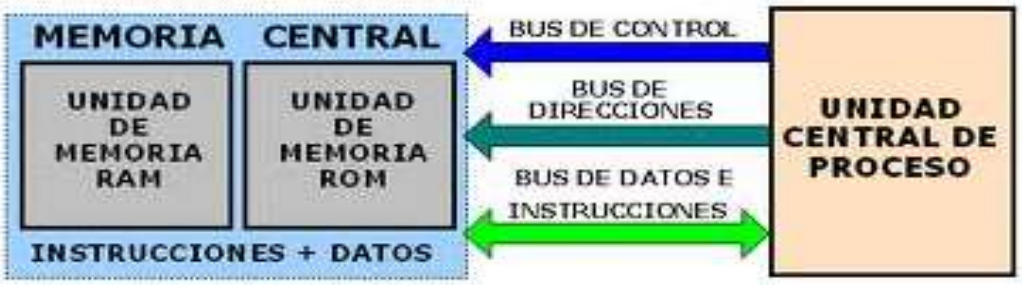


Figura 1.4 Arquitectura Von Neumann

1.1.4.2 Arquitectura Harvard.

La Arquitectura Harvard figura 1. 5 tiene la Unidad Central de Proceso (CPU) conectada a dos memorias (una con las instrucciones y otra con los datos) por medio de dos buses diferentes.

Una de las memorias contiene solamente las instrucciones del programa (Memoria de Programa), y la otra, sólo almacena datos (Memoria de Datos).

Ambos buses son totalmente independientes y pueden ser de distintos anchos. Para un procesador de Set de Instrucciones Reducido, o RISC (Reduced Instrucción Set Computer), el set de instrucciones y el bus de memoria de programa pueden diseñarse de tal manera que todas las instrucciones tengan una sola posición de memoria de programa de longitud.

Además, al ser los buses independientes, la CPU puede acceder a los datos para completar la ejecución de una instrucción, y al mismo tiempo leer la siguiente instrucción a ejecutar.

Las ventajas de esta arquitectura son:

- El tamaño de las instrucciones no está relacionado con el de los datos, y por lo tanto puede ser optimizado para que cualquier instrucción ocupe una sola posición de memoria de programa, logrando así mayor velocidad y menor longitud de programa.
- El tiempo de acceso a las instrucciones puede superponerse con el de los datos, logrando una mayor velocidad en cada operación.

Una pequeña desventaja de los procesadores con arquitectura Harvard, es que deben poseer instrucciones especiales para acceder a tablas de valores constantes que pueda ser necesario incluir en los programas, ya que estas tablas se encontraran físicamente en la memoria de programa (por ejemplo en la EPROM de un microprocesador).

Figura 1. 5 Arquitectura Harvard

1.1.5 EL PROCESADOR O UCP[3]

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.

- 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.

- 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.

- 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).

[3] http://www.monografias.com/trabajos12/microco/microco.shtml#DIFER

1.1.6 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. Hay dos peculiaridades que diferencian a los microcontroladores de los computadores personales:

- No existen sistemas de almacenamiento masivo como disco duro o disquetes.
- Como el microcontrolador sólo se destina a una tarea en la memoria ROM, sólo hay que almacenar un único programa de trabajo.

La RAM en estos dispositivos es de poca capacidad pues sólo debe contener las variables y los cambios de información que se produzcan en el transcurso del programa. Por otra parte, como sólo existe un programa activo, no se requiere guardar una copia del mismo en la RAM pues se ejecuta directamente desde la ROM.

Los usuarios de computadores personales están habituados a manejar Megabytes de memoria, pero, los diseñadores con microcontroladores trabajan con capacidades de ROM comprendidas entre 512 bytes y 8 k bytes y de RAM comprendidas entre 20 y 512 bytes.

Según el tipo de memoria ROM que dispongan los microcontroladores, la aplicación y utilización de los mismos es diferente. Se describen las cinco versiones de memoria no volátil que se pueden encontrar en los microcontroladores del mercado.

- ROM CON MÁSCARA: Es una memoria no volátil de sólo lectura cuyo contenido se graba durante la fabricación del chip. El elevado coste 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.

- OTP: El microcontrolador contiene una memoria no volátil de sólo lectura "programable una sola vez" por el usuario. OTP (One Time Programmable). Es el usuario quien puede escribir el programa en el chip mediante un sencillo grabador controlado por un programa desde un PC. La versión OTP es recomendable cuando es muy corto el ciclo de diseño del producto, o bien, en la construcción de prototipos y series muy pequeñas. Tanto en este tipo de memoria como en la EPROM, se suele usar la encriptación mediante fusibles para proteger el código contenido.
- EPROM: Los microcontroladores que disponen de memoria EPROM (Erasable Programmable Read Only Memory) pueden borrarse y grabarse muchas veces. La grabación se realiza, como en el caso de los OTP, con un grabador gobernado desde un PC. Si, posteriormente, se desea borrar el contenido, disponen de una ventana de cristal en su superficie por la que se somete a la EPROM a rayos ultravioleta durante varios minutos. Las cápsulas son de material cerámico y son más caros que los microcontroladores con memoria OTP que están hechos con material plástico.
- EEPROM: Se trata de memorias de sólo lectura, programables y borrables eléctricamente EEPROM (Electrical Erasable Programmable Read Only Memory). Tanto la programación como el borrado, se realizan eléctricamente desde el propio grabador y bajo el control programado de un PC. Es muy cómoda y rápida la operación de grabado y la de borrado. No disponen de ventana de cristal en la superficie. Los microcontroladores dotados de memoria EEPROM una vez instalados en el circuito, pueden grabarse y borrarse cuantas veces se quiera sin ser retirados de dicho circuito. Para ello se usan "grabadores en circuito" que confieren una gran flexibilidad y rapidez a la hora de realizar modificaciones en el programa de trabajo.

  El número de veces que puede grabarse y borrarse una memoria EEPROM es finito, por lo que no es recomendable una reprogramación continúa. Son muy idóneos para la enseñanza y la Ingeniería de diseño.

Se va extendiendo en los fabricantes la tendencia de incluir una pequeña zona de memoria EEPROM en los circuitos programables para guardar y modificar cómodamente una serie de parámetros que adecuan el dispositivo a las condiciones del entorno. Este tipo de memoria es relativamente lenta.

- FLASH: Se trata de una memoria no volátil, de bajo consumo, que se puede escribir y borrar. Funciona como una ROM y una RAM pero consume menos y es más pequeña.

  A diferencia de la ROM, la memoria FLASH es programable en el circuito. Es más rápida y de mayor densidad que la EEPROM.

  La alternativa FLASH está recomendada frente a la EEPROM cuando se precisa gran cantidad de memoria de programa no volátil. Es más veloz y tolera más ciclos de escritura/borrado.

Las memorias EEPROM y FLASH son muy útiles al permitir que los microcontroladores que las incorporan puedan ser reprogramados "en circuito", es decir, sin tener que sacar el circuito integrado de la tarjeta. Así, un dispositivo con este tipo de memoria incorporado al control del motor de un automóvil permite que pueda modificarse el programa durante la rutina de mantenimiento periódico, compensando los desgastes y otros factores tales como la compresión, la instalación de nuevas piezas, etc. La reprogramación del microcontrolador puede convertirse en una labor rutinaria dentro de la puesta a punto.

1.1.7 PUERTAS DE ENTRADA Y SALIDA

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

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.

1.1.8 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 junto a elementos pasivos o bien un resonador cerámico o una red R-C.

Aumentar la frecuencia de reloj supone disminuir el tiempo en que se ejecutan las instrucciones pero lleva aparejado un incremento del consumo de energía.

1.1.9 RECURSOS ESPECIALES

Cada fabricante oferta numerosas versiones de una arquitectura básica de microcontrolador. En algunas amplía las capacidades de las memorias, en otras incorpora nuevos recursos, en otras reduce las prestaciones al mínimo para aplicaciones muy simples, etc. La labor del diseñador es encontrar el modelo mínimo que satisfaga todos los requerimientos de su aplicación. De esta forma, minimizará el coste, el hardware y el software. Los principales recursos específicos que incorporan los microcontroladores son:

• Temporizadores o "Timers".

• Perro guardián o "Watchdog".

• Protección ante fallo de alimentación o "Brownout".

• Estado de reposo o de bajo consumo.

• Conversor A/D.

• Conversor D/A.

• Comparador analógico.

• Modulador de anchura de impulsos o PWM.

• Puertas de E/S digitales.

• Puertas de comunicación.

1.1.9.1 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 disminuyendo al ritmo de los impulsos de reloj o algún múltiplo hasta que se desborde y llegue a 0, momento en el que se produce un aviso.

Cuando se desean contar acontecimientos que se materializan por cambios de nivel o flancos en alguna de las patitas del microcontrolador, el mencionado registro se va incrementando o decrementando al ritmo de dichos impulsos.

1.1.9.2 Perro Guardián o “Watchdog”

Cuando el computador personal se bloquea por un fallo del software u otra causa, se pulsa el botón del reset y se reinicializa el sistema. Pero un microcontrolador funciona sin el control de un supervisor y de forma continuada las 24 horas del día. 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.

1.1.9.3 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.

1.1.9.4 Estado de reposo ó de bajo consumo

Son abundantes las situaciones reales de trabajo en que el microcontrolador debe esperar, sin hacer nada, a que se produzca algún acontecimiento externo que le ponga de nuevo en funcionamiento. 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.

1.1.9.5 Conversor A/D (CAD)

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.

1.1.9.6 Conversor D/A (CDA)

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. Existen muchos efectores que trabajan con señales analógicas.

1.1.9.7 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 una de las patitas de la cápsula. La salida del comparador proporciona un nivel lógico 1 ó 0 según una señal sea mayor o menor que la otra.

También hay modelos de microcontroladores con un módulo de tensión de referencia que proporciona diversas tensiones de referencia que se pueden aplicar en los comparadores.

1.1.9.8 Modulador de anchura de impulsos o PWM

Son circuitos que proporcionan en su salida impulsos de anchura variable, que se ofrecen al exterior a través de las patitas del encapsulado.

1.1.9.9 Puertos de E/S digitales

Todos los microcontroladores destinan algunas de sus patitas a soportar líneas de E/S digitales. Por lo general, estas líneas se agrupan de ocho en ocho formando Puertos.

Las líneas digitales de los Puertos pueden configurarse como Entrada o como Salida cargando un 1 ó un 0 en el bit correspondiente de un registro destinado a su configuración.

1.1.9.10 Puertos de comunicación

Con objeto de dotar al microcontrolador de la posibilidad de comunicarse con otros dispositivos externos, otros buses de microprocesadores, buses de sistemas, buses de redes y poder adaptarlos con otros elementos bajo otras normas y protocolos. Algunos modelos disponen de recursos que permiten directamente esta tarea, entre los que destacan:

- UART, adaptador de comunicación serie asíncrona.
- USART, adaptador de comunicación serie síncrona y asíncrona
- Puerta paralela esclava para poder conectarse con los buses de otros microprocesadores.
- USB (Universal Serial Bus), que es un moderno bus serie para los PC.
- Bus I2C, que es un interfaz serie de dos hilos desarrollado por Philips.
- 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. En EE.UU. se usa el J185O.

1.1.10 PRINCIPALES FABRICANTES

Por lo general los fabricantes de microprocesadores lo son de microcontroladores. Los fabricantes de microcontroladores son más de 50, podemos mencionar como los principales a:

- Atmel
- Motorola
- Intel
- Microchip
- NEC
- Hitachi
- Mitsubishi
- Philips
- Matsushita
- Toshiba
- AT&T
- Zilog
- Siemens
- National Semiconductor

Para nuestro proyecto se utilizará un microcontrolador AVR, razón por la cual se mencionará a ATMEL que es el fabricante.

1.1.10.1 Atmel

ATMEL es una compañía de semiconductores, fundada en 1984. Su línea de productos incluye microcontroladores (incluyendo derivados del 8051, el AT91SAM basados en ARM, y sus arquitecturas propias AVR y AVR32), dispositivos de radiofrecuencia, memorias EEPROM y Flash, ASICs, WiMAX, y muchas otras. También tiene capacidad de ofrecer soluciones del tipo system on chip (SoC).[4]

1.2 MICROCONTROLADOR AVR.

Los AVR son una familia de microcontroladores RISC de Atmel. La arquitectura de los AVR fue concebida por dos estudiantes en el Norwegian Institute of Technology, y posteriormente refinada y desarrollada en Atmel Norway, la empresa subsidiaria de Atmel, fundada por los dos arquitectos del chip.

1.2.1 DEFINICIÓNES DE AVR

- [5]En inglés, abreviatura para:
  - ✓ Automatic Voltage Regulator (Regulador automático de voltaje): dispositivo de hardware empleado para mantener un voltaje específico en dispositivos electrónicos.
  - ✓ Automatic Voice Recognition (Reconocimiento automático de voz): posibilidad de los dispositivos electrónicos/informáticos para reconocer y entender la voz humana.
  - ✓ Automatic Voice Response (Respuesta automática de voz): es una respuesta en voz por parte de una computadora, ya sea una respuesta pregrabada o generada artificialmente.
- [6]Audio, Video, Radio: Microcontroladores pensados para aplicaciones en audio, video y radio.

[4] http://es.wikipedia.org/wiki/Atmel
[5] http://www.alegsa.com.ar/Dic/avr.php
[6] http://www.clubse.com.ar/DIEGO/NOTAS/2/nota18.htm

- Advanced Virtual Risc.
- Corresponde a las iniciales de sus inventores: Alf Egil Bogen and Vegard Wollan (AlfVegardRisc).

1.2.2 CARACTERÍSTICAS DE LA FAMILIA ATMEL[7]

La familia de Microcontroladores AVR, pertenecen a ATMEL, los cuales poseen una arquitectura moderna. Estos Microcontroladores están divididos en tres grupos: TinyAVR, AVR Clásico y MegaAVR.

Se muestran en la tabla figura 1.6 los dispositivos Microcontroladores de la serie AVR. Todos ellos se fabrican en el mismo proceso y los mismos niveles de implantación. Los dispositivos varían en densidad de memoria figura 1.6

Con 1K byte Flash	Con 2K byte Flash	Con 4K byte Flash	Con 8K byte Flash	Con 12K byte Flash	Con 16K byte Flash	Con 32K/40K byte Flash	Con 64K byte Flash	Con 128K byte Flash	Con 256K byte Flash
Tiny13	Tiny14	Mega48	Mega8	90VC8544	Mega16	Mega32	Mega64	Mega128	Mega2560
	Tiny25	Tiny45	Mega8515		Mega162	Mega325	Mega645	Mega1280	Mega2561
	Tiny26		Mega8535		Mega169	Mega329	Mega649	Mega1281	
	Tiny2313		Mega88		Mega165	Mega406	Mega644		
					Mega168				

Figura 1. 6 Dispositivos Microcontroladores de la serie AVR.

1.2.3 DESCRIPCIÓN DEL AVR

[8]El AVR es una CPU de Arquitectura Harvard. Tiene 32 registros de 8 bits. Algunas instrucciones sólo operan en un subconjunto de estos registros. La concatenación de los 32 registros, los registros de entrada/salida y la memoria de datos conforman un espacio de direcciones unificado, al cual se accede a través de operaciones de carga/almacenamiento. A diferencia de los microcontroladores PIC, el stack se ubica en este espacio de memoria unificado, y no está limitado a un tamaño fijo.

El AVR fue diseñado desde un comienzo para la ejecución eficiente de código C compilado. Como este lenguaje utiliza profusamente punteros para el manejo de variables en memoria, los tres últimos pares de registros internos del procesador,

[7] http://www.scribd.com/doc/7828553/Capitulo1-Introduccion-del-ATmega32-espanol
[8] http://es.wikipedia.org/wiki/AVR

son usados como punteros de 16 bit al espacio de memoria externa, bajo los nombres X, Y Z. Esto es un compromiso que se hace en arquitecturas de ocho bit desde los tiempos de Intel 8008, ya que su tamaño de palabra nativo de 8 bit (256 localidades accedidas) es pobre para direccionar. Por otro lado, hacer que todo el banco superior de 16 registros de 8 bit tenga un comportamiento alterno como un banco de 8 registros de 16 bit, complicaría mucho el diseño, violando la premisa original de su simplicidad. Además, algunas instrucciones tales como 'suma inmediata' ('add immediate' en inglés) faltan, ya que la instrucción 'resta inmediata' ('substract immediate' en inglés) con el complemento dos puede ser usada como alternativa.

El set de instrucciones AVR está implementado físicamente y disponible en el mercado en diferentes dispositivos, que comparten el mismo núcleo AVR pero tienen distintos periféricos y cantidades de RAM y ROM: desde el microcontrolador de la familia *Tiny AVR* ATtiny11 con 1KB de memoria flash y sin RAM (sólo los 32 registros), y 8 pines, hasta el microcontrolador de la familia *Mega AVR*ATmega2560 con 256KB de memoria flash, 8KB de memoria RAM, 4KB de memoria EEPROM, conversor análogo digital de 10 bits y 16 canales, temporizadores, comparador analógico, JTAG, etc. La compatibilidad entre los distintos modelos es preservada en un grado razonable.

Los microcontroladores AVR tienen una cañería ('pipeline' en inglés) con dos etapas (cargar y ejecutar), que les permite ejecutar la mayoría en un ciclo de reloj, lo que los hace relativamente rápidos entre los microcontroladores de 8-bit.

El set de instrucciones de los AVR es más regular que la de la mayoría de los microcontroladores de 8 bits (por ejemplo, los PIC). Sin embargo, no es completamente ortogonal:

- Los registros punteros X, Y, Z tienen capacidades de direccionamiento diferentes entre sí (ver más arriba por qué)
- Los registros 0 al 15 tienen diferentes capacidades de direccionamiento que los registros 16 al 31.

- Las registros de I/O 0 al 31 tienen distintas características que las posiciones 32 al 63.
- La instrucción CLR afecta los 'flag', mientras que la instrucción SER no lo hace, a pesar de que parecen ser instrucciones complementarias (dejar todos los bits en 1, y dejar todos los bits en 0 respectivamente).
- Los códigos de operación 0x95C8 y 0x9004 hacen exactamente lo mismo (LPM).

1.2.4 DIAGRAMA DE BLOQUES

En la figura 1.7 se puede observar el diagrama en bloques de la serie ATtiny 25/45/85 de ATmel. El núcleo del AVR combina un conjunto rico de instrucciones con 32 registros de trabajo de propósito general. Los 32 registros están directamente conectados a la Unidad Aritmético-Lógica (ALU), permitiendo que 2 registros independientes se accedan en una sola instrucción ejecutada en un ciclo de reloj. La arquitectura resultante es más eficiente en lo que respecta a código, en tanto que logra un rendimiento hasta 10 veces superior que los microcontroladores convencionales CISC. De las características enunciadas anteriormente, sobre el ATtiny 25/45/85 podemos destacar las siguientes: 2/4/8kB de Memoria Flash Programable en el Sistema, 128/256/512 bytes de EEPROM, 128/256/256 bytes de SRAM, 6 líneas de entrada/salida de propósito general, 32 registros de trabajo de propósito general, un Temporizador/Contador de 8 bits con modos de comparación, un Temporizador/Contador de alta velocidad de 8 bits, una Interfaz Serie Universal, Interrupciones Internas y Externas, un ADC de 4 canales de 10 bits, un Temporizador Programable de Vigilancia con Oscilador Interno y 3 modos de ahorro de potencia seleccionables por software. El modo de Descanso detiene la CPU en tanto que permite que la SRAM, el Temporizador/Contador, el ADC, el Comparador Analógico, y el sistema de Interrupción sigan trabajando. El modo de Reducción de Potencia guarda el contenido de los registros, inhabilitando todas las funciones del integrado hasta la siguiente Interrupción o re inicialización. El modo de Reducción de Ruido del ADC detiene la CPU y todos los módulos de E/S excepto el ADC, a fin de minimizar el ruido de conmutación durante las conversiones del ADC.

El dispositivo se fabrica usando la tecnología de memoria no-volátil y alta densidad de ATMEL.

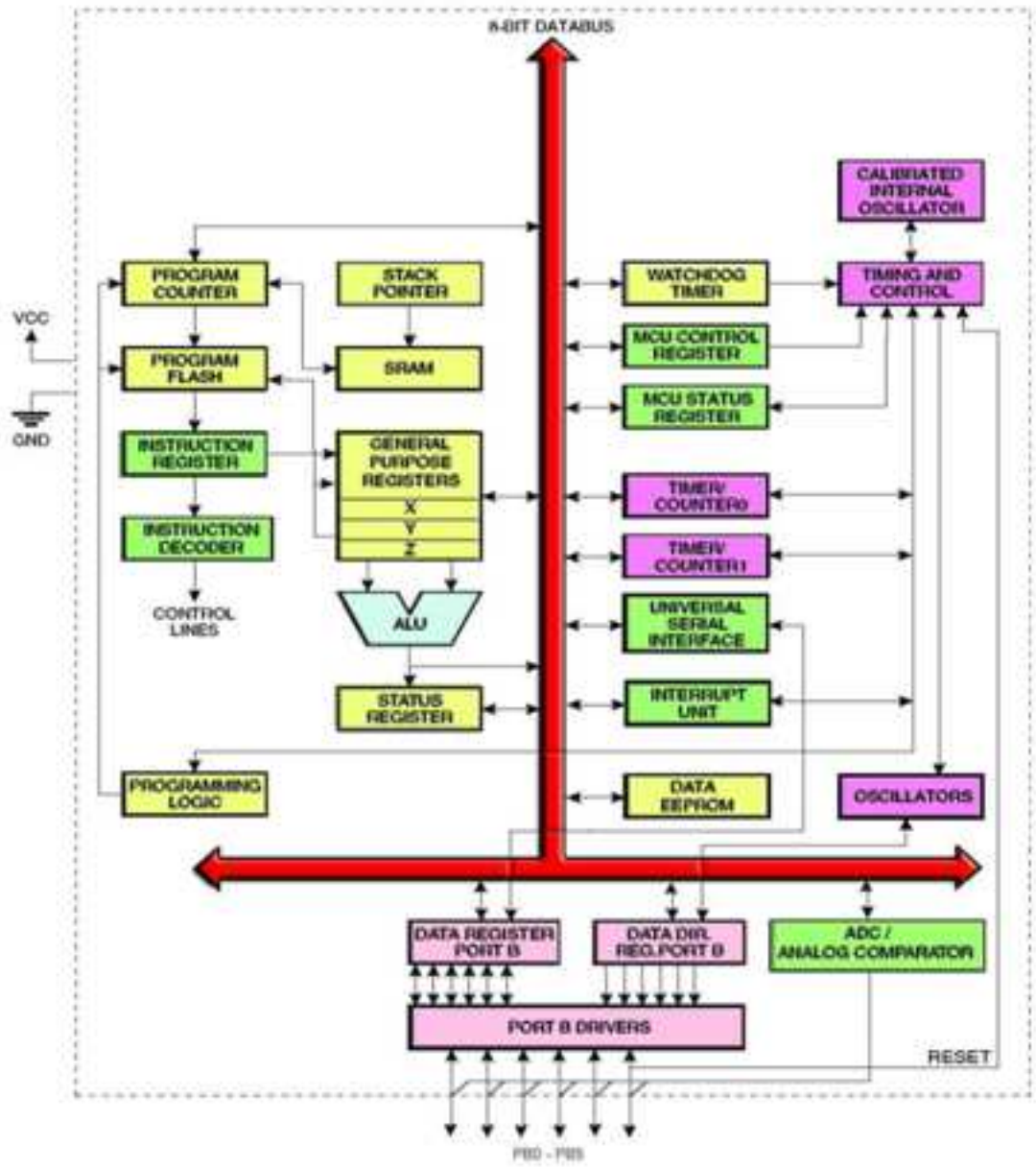


Figura 1. 7 diagrama en bloques de la serie Atiny 25/45/85 de Atmel.

1.2.5 LA CPU DEL AVR

Trataremos la arquitectura del núcleo del AVR figura 1.8 en general. La función principal del núcleo de la CPU es asegurar una correcta ejecución del programa. La CPU, por lo tanto, debe acceder a memorias, realizar cálculos, controlar

periféricos, y manejar interrupciones. A fin de maximizar el desempeño y el paralelismo, el AVR usa una arquitectura Harvard, con memorias y buses separados para el programa y los datos. Las instrucciones que están en la memoria de Programa se ejecutan con un solo nivel de transmisión por conductos. Mientras que se ejecuta una instrucción, se extrae la siguiente instrucción de la memoria de Programa. Este concepto permite que las instrucciones se ejecuten en cada ciclo de reloj. La memoria de programa es la memoria flash reprogramable en el Sistema. El Archivo de Registros de acceso rápido contiene 32 registros de trabajo de propósito general de 8 bits con un tiempo de acceso de un solo ciclo de reloj. Esto permite la operación de la Unidad Aritmético Lógica (ALU) en un sólo ciclo. En una típica operación de la ALU, se toman 2 operandos del Archivo de Registros, se ejecuta la operación, y el resultado se almacena nuevamente en el Archivo de Registros en un ciclo de reloj.

La ALU soporta operaciones aritméticas y lógicas entre registros o entre una constante y un registro. Las operaciones de un solo registro también se pueden ejecutar en la ALU. Luego de una operación aritmética, el Registro de Estado se actualiza para reflejar la información sobre el resultado de la operación. El Programa tiene instrucciones de salto condicional e incondicional e instrucciones de llamada, capaces de direccionar en forma directa todo el espacio de direcciones. La mayoría de las instrucciones del AVR tienen un solo formato de palabra de 16 bits. Cada dirección de memoria de Programa contiene una instrucción de 16 o de 32 bits.

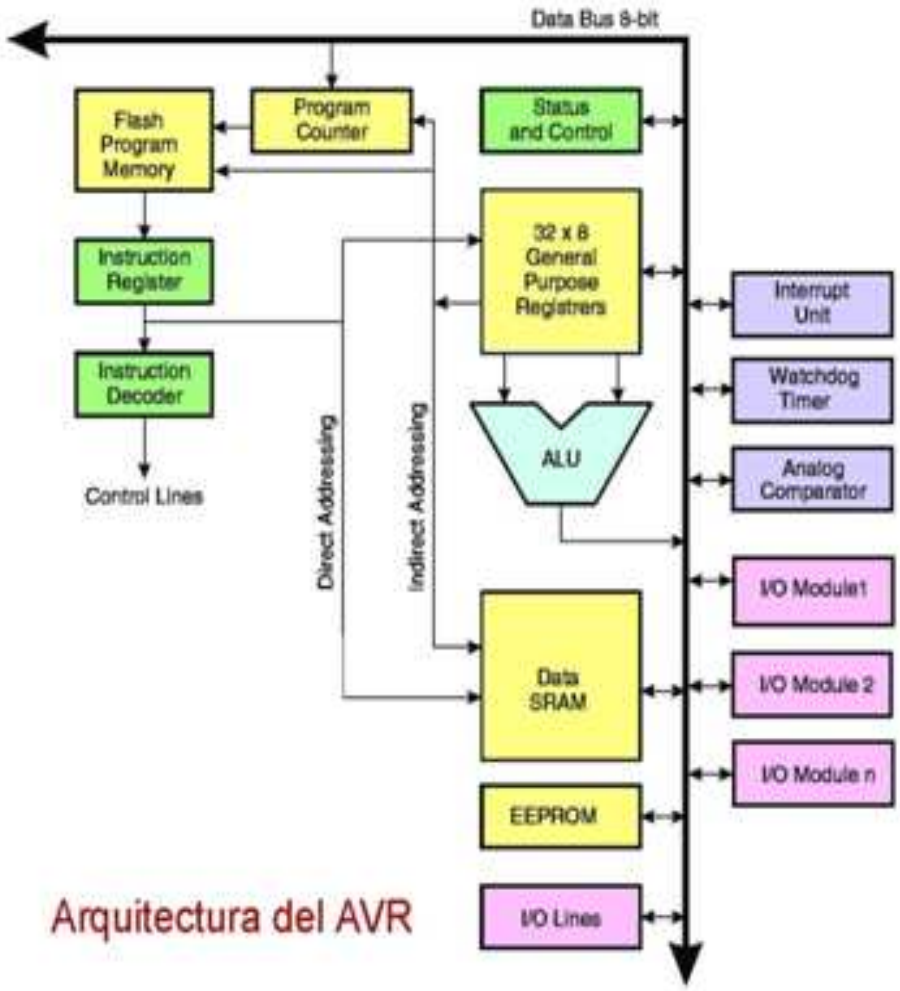


Figura 1. 8 Arquitectura del AVR

1.2.6 EL REGISTRO DE ESTADO

El Registro de Estado figura 1.9 contiene información sobre el resultado de la instrucción más recientemente ejecutada. Esta información se puede usar para alterar el flujo del programa a fin de ejecutar operaciones condicionales. Notemos que el Registro de Estado se actualiza después de todas las operaciones de la ALU. Esto, en muchos casos, evita la necesidad de usar instrucciones de comparación especiales, resultando un código más compacto y más rápido. El Registro de Estado no se almacena automáticamente cuando se ingresa a una rutina de interrupción y se vuelve a almacenar cuando se regresa de una interrupción. Esto se maneja mediante el software. El Registro de Estado del AVR (SREG) posee una estructura como la mostrada en la figura 9. Bit 7-I: Habilitación Global de Interrupción. Este bit debe ponerse en 1 para que se habiliten las interrupciones. El control individual de habilitación de interrupción se ejecuta luego

en registros de control separados. Si se pone en 0, no se habilita ninguna interrupción, independientemente de cómo estén las posiciones individuales de habilitación de interrupción. El bit I se pone en 0 mediante hardware después que haya ocurrido una interrupción, y se pone en 1 mediante la instrucción RETI para permitir interrupciones subsiguientes. El bit I también se puede poner en 1 y en 0 mediante las instrucciones SEI y CLI.

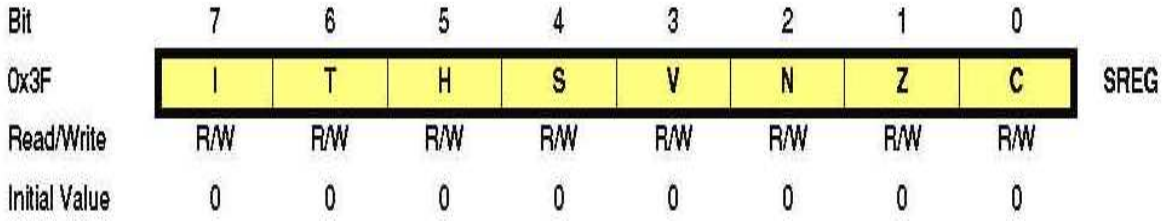


Figura 1. 9 Registro de Estado

1.2.7 REGISTROS DE PROPÓSITO GENERAL

El Archivo de Registros se optimiza para el conjunto de instrucciones RISC mejorado del AVR. A fin de lograr el desempeño y la flexibilidad requeridos, el Archivo de Registros soporta los siguientes esquemas de E/S:

• Un operando de salida de 8 bits y una entrada de resultados de 8 bits.
• Dos operandos de salida de 8 bits y una entrada de resultados de 8 bits.
• Dos operandos de salida de 8 bits y una entrada de resultados de 16 bits.
• Un operando de salida de 16 bits y una entrada de resultados de 16 bits.

La figura 1.10 se muestra la estructura de los 32 registros de trabajo de propósito general de la CPU. La mayoría de las instrucciones que operan en el Archivo de Registros tienen acceso directo a todos los registros, y la mayoría de ellas son instrucciones de un solo ciclo. Como se ve en la figura, a cada registro se le asigna una dirección de memoria de Datos, las cuales se mapean directamente en las primeras 32 posiciones del Espacio de Datos del usuario. Aunque no se implementan físicamente como posiciones de memoria de SRAM, esta organización de memoria proporciona una gran flexibilidad en el acceso de los registros, ya que los registros apuntadores X, Y Z pueden apuntar a cualquier registro del archivo. Los Registros X, Y Z

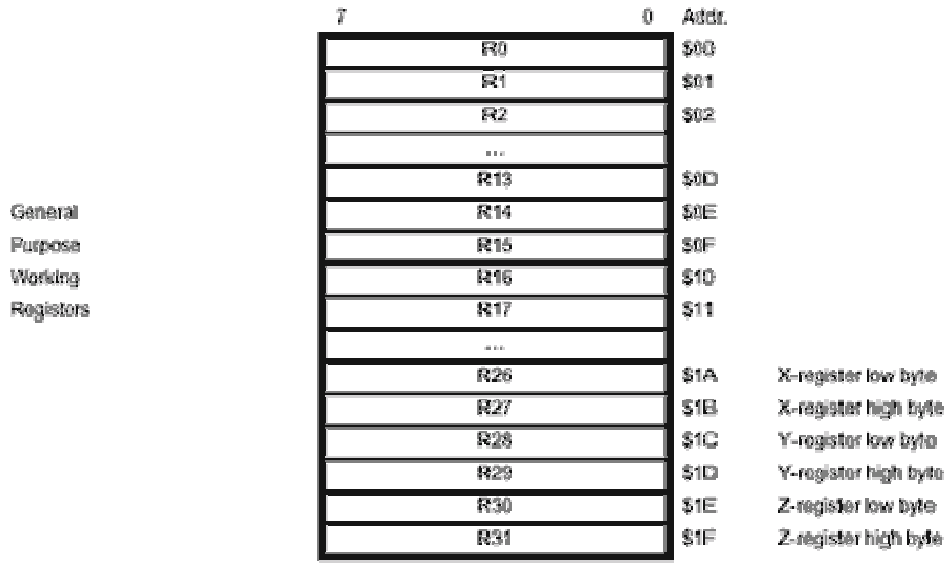


Figura 1. 10 Registro de trabajo de propósitos generales del AVR.

1.2.8 FAMILIAS BÁSICAS[9]

AVRs se clasifica generalmente en cuatro amplios grupos:

1.2.8.1 TinyAVRs

- 1-8 memoria del programa del KB
- paquete 8-32-pin
- Sistema limitado el periférico

1.2.8.2 MegaAVRs

- 4-256 memoria del programa del KB
- paquete 28-100-pin
- Sistema de instrucción extendida (multiplique las instrucciones y las instrucciones para manejar memorias más grandes del programa)
- Sistema extenso el periférico

1.2.8.3 XMEGA

- 16-256 memoria del programa del KB
- paquete 44-100-pin
- El funcionamiento extendido ofrece, por ejemplo el acceso directo de memoria, el “sistema del acontecimiento”, y la ayuda de la criptografía.
- Periférico extenso fijado con DACs

[9] http://www.worldlingo.com/ma/enwiki/es/Atmel_AVR

1.2.8.4 Específico AVRs del uso

- MegaAVRs con las características especiales no encontradas en los otros miembros de la familia del AVR, tales como regulador del LCD, regulador del USB, PWM avanzado, PODER etc.
- FPSLIC (Circuito integrado programable del nivel de sistema del campo), una base del AVR en-muere con FPGA. El FPSLIC utiliza SRAM para el código del programa del AVR, desemejante del resto del AVRs. En parte debido a la diferencia relativa de la velocidad entre SRAM y el flash, la base del AVR en el FPSLIC puede funcionar en hasta 50MHz.

1.2.9 DIFERENCIAS ENTRE ATMEL Y OTROS MICROCONTROLADORES

[10]ATMEL fabrica los microcontroladores de la familia AVR, esta nueva tecnología proporciona todos los beneficios habituales de arquitectura RISC y memoria flash reprogramable eléctricamente. La característica que los identifica a estos microcontroladores de ATMEL es la memoria flash y EEPROM que incorpora. AVR compite con varias familias de microcontroladores bien establecidas en el mercado, tales como 8051 de Intel, 68HC11 de Motorola y la familia PIC de Microchip. La firma también produce y vende varios subproductos de la popular familia 8051 con la diferencia de que están basados en la memoria flash.

El diseño AVR de ATMEL difiere de los demás microcontroladores de 8 bits por tener mayor cantidad de registros (32) y un conjunto ortogonal de instrucciones. AVR es mucho más moderna que su competencia. Por ejemplo, los 8051, 6805 y los PIC, se los arreglan con un único acumulador, los 658HC11 y 68HC12 tienen simplemente 2. Esto hace que la arquitectura AVR sea más fácil de programar a nivel de lenguaje ensamblador y que sea fácil de optimizar con un compilador. El gran conjunto de registros disminuye la dependencia respecto a la memoria, lo cual mejora la velocidad y disminuye las necesidades de almacenamiento de datos. Además casi todas las instrucciones se ejecutan en 1 ó 2 ciclos de reloj versus 5-10 ciclos de reloj para los chips 8051, 6805, 68HC11 y PIC.

Una vez ensamblado y depurado el código fuente del programa, se transferirá el código máquina a la memoria flash del microcontrolador para esto se debe disponer de otro entorno de desarrollo para programar en forma serial o paralelo la memoria flash.

[10] http://www.lulu.com/items/volume_38/588000/588200/1/print/SESION_1_ATMEGA8.pdf

[11]A diferencia de los microcontroladores PIC, el stack se ubica en este espacio de memoria unificado, y no está limitado a un tamaño fijo.

1.3 MICROCONTROLADOR ATMEGA 164P

1.3.1 CARACTERÍSTICAS:

Microcontrolador AVR de 8 bits de alto rendimiento y bajo consumo figura 1.11.

1.3.1.1 Arquitectura Avanzada RISC

- 131 instrucciones. La mayoría de un solo ciclo de reloj de ejecución.
- 32 registros de trabajo de 8 bits para propósito general.
- Funcionamiento estático total.
- Capacidad de procesamiento de unos 20 MIPS a 20 MHz
- Multiplicador por hardware de 2 ciclos

1.3.1.2 Memorias de programa y de datos no volátiles de alta duración

- 16/32/44 K bytes de FLASH auto programable en sistema
- 512B/1K/2K bytes de EEPROM
- 1/2/4K bytes de SRAM Interna
- Ciclos de escritura/borrado: 10.000 en Flash / 100.000 en EEPROM
- Retención de Datos: 20 años a 85ºC / 100 años a 25ºC
- Sección opcional de código Boot con bits de bloqueo independientes. Programación en sistema del programa Boot que se encuentra dentro del mismo chip. Operación de lectura durante la escritura.
- Bloqueo programable para la seguridad del software.

1.3.1.3 Interfase JTAG

- Capacidades de Boundary Scan de acuerdo con el estándar JTAG
- Soporte Extendido Debug dentro del chip.

[11] http://es.wikipedia.org/wiki/AVR

- Programación de FLASH, EEPROM, fusibles y bits de bloqueo a través fase JTAG.

1.3.1.4 Características de los periféricos

- Dos Timer/Contadores de 8 bits con prescalamiento separado y modo comparación.
- Un Timer/Contador de 16 bits con prescalamiento separado, modo comparación y modo de captura.
- Contador en Tiempo Real con Oscilador separado
- 6 Canales para PWM
- ADC de 10 bits y 8 canales
- Modo Diferencial con ganancia seleccionable a x1, x10 o x200.
- Interface serie de dos hilos con byte orientado.
- Dos puertos Seriales USART Programables
- Interfaz Serial SPI maestro-esclavo
- Watchdog Timer programable con oscilador independiente, dentro del mismo chip.
- Comparador Analógico dentro del mismo Chip
- Interrupt and Wake-up on Pin Change

1.3.1.5 Características especiales del microcontrolador

- Power-on Reset (en el encendido) y detección de Brown-out (pérdida de polarización) programable.
- Oscilador RC interno calibrado.
- Fuentes de interrupción externas e internas.

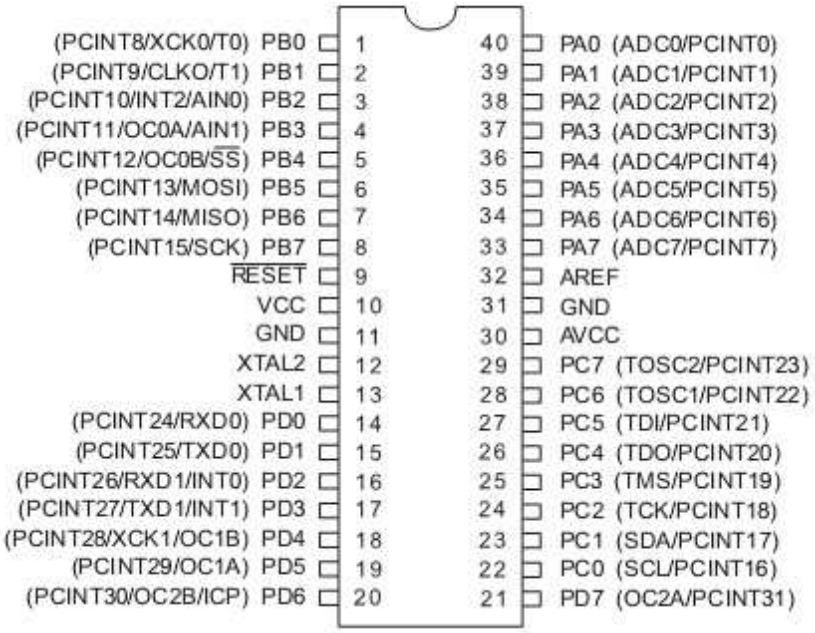


Figura 1. 11 Encapsulado

- 6 modos de descanso: Idle, Reducción de Ruido ADC, Power-save, Power-down, Standby y Standby extendido.

1.3.1.6 Encapsulados para Entradas/Salidas (E/S)

- 32 líneas de E/S programables.
- PDIP de 40 pines, TQFP y QFN/MLF de 44 pines.

1.3.1.7 Voltajes de Operación

- 1.8– 5.5V para el ATMEGA 164P/324P/644PV
- 2.7– 5.5V para el ATMEGA 164P/324P/644P

1.3.1.8 Velocidad de Funcionamiento

- ATMEGA 164P/324P/644PV: 0– 4MHz a 1.8– 5.5V - 10MHz a 2.7– 5.5V
- ATMEGA 164P/324P/644PV: 0– 10MHz a 2.7– 5.5V - 20MHz a 4.5– 5.5V

1.3.1.9 Consumo de energía a 1MHz, 1.8V, 25ºC para el ATMEGA 164P/324P/644P

- Activo: 0.4Ma
- Modo Power-down: 0.1uA
- Modo Power-Save: 0.6uA (Incluye RTC de 32 Khz)

1.3.1.10 Configuración de Pines

En la Figura se muestra los pines de salida del ATmega164P/324P/644P

TQFP/QFN/MLF figura 1.12.

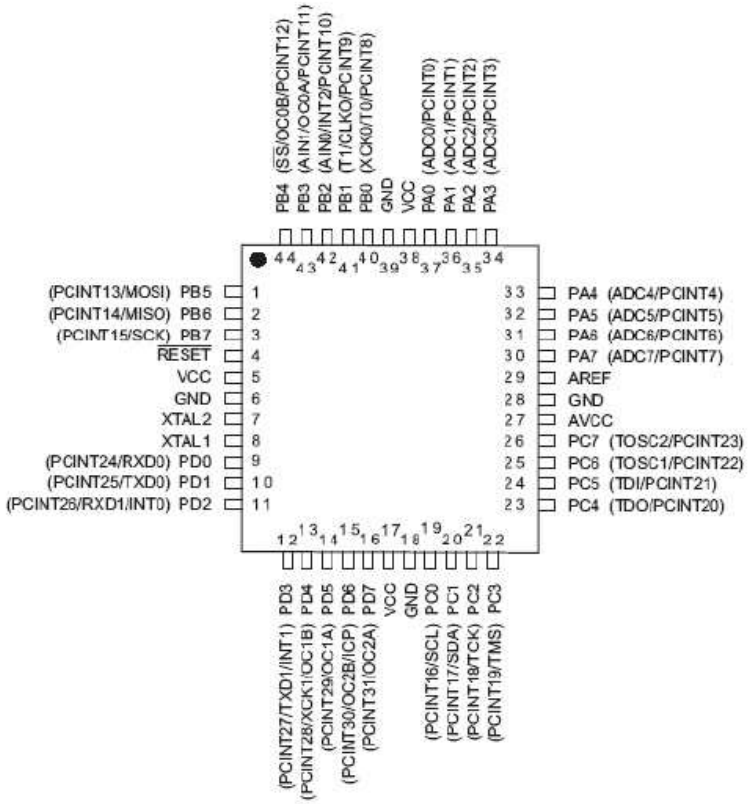


Figura 1. 12 Configuración de los Pines

1.3.1.11 Revisión Global

El ATmega164P/324P/644P es un microcontrolador CMOS de 8 bits de bajo consumo basado en la arquitectura RISC mejorada. Sus instrucciones se ejecutan en un ciclo de máquina, el ATmega164P / 324P / 644P consigue transferencia de información alrededor de 1 MIPS por MHz admitido por el sistema, permitiendo al diseñador del sistema optimizar el consumo de energía versus la velocidad de procesamiento.

1.3.1.12 Descripción del Microcontrolador ATMEGA164 P

Este modulo tiene la capacidad de cambiar la configuración de los diferentes sensores, agregar, eliminar o modificar claves de acceso y activar o desactivar al Sistema de Seguridad. Para lo anterior cuenta con un programa figura 1.13.

Figura 1.13 ATmega164P

1.3.1.13 Encapsulados atmega164

Estos encapsulados presentan lo siguiente

- Terminales de polarización
- Voltajes de funcionamiento
  - ✓ 2.7 - 5.5V (ATmega164P)
  - ✓ 1.8 - 5.5V (ATmega164PV)
  - ✓ 200 mA es la corriente máxima en los terminales VCC y GND
- Entrada para el reset
- Reset en el encendido y externo
  - ✓ Para que se active cuando se polariza
  - ✓ O en cualquier instante
- Terminales Para El Cristal

  Rangos de velocidad

  - ✓ 0 – 20 MHz (ATmega164P)
  - ✓ 0 – 10 MHz (ATmega164PV)
- Pórticos de Entrada y Salida Paralela
  - ✓ I/O
  - ✓ 32 líneas de E/S programables
    - Pórtico A (8 bits)
    - Pórtico B (8 bits)
    - Pórtico C (8 bits)
    - Pórtico D (8 bits)

- Conversor de Analógico a Digital
  - ✓ ADC de 10 bits 8 canales
    - 8 canales de un solo terminal
    - 2 canales diferenciales con ganancia programable de x1, x10 y x200
    - 7 canales diferenciales sólo en el encapsulado TQFP
- Comparador Analógico
  - ✓ Incorporado en el mismo chip
    - La entrada positiva es AIN0
    - La negativa es AIN1
    - Se puede reemplazar AIN1 por las entradas analógicas ADC0, ADC7
- Interface JTAG para sistema de depuración
  - ✓ JTAG (IEEE std1149.1)
    - En la depuración se tiene acceso a todos periféricos
    - Programación de la Flash, EEPROM, Fusibles y Bits de seguridad
    - Depuración soportada por el AVR Studio
- Interface a Periféricos Seriales
  - ✓ SPI
    - Full dúplex
    - Tres líneas para comunicaciones sincrónicas
    - Operación maestro / esclavo
    - Siete velocidades programables
    - Bandera de fin de la transmisión
- Interrupciones Externas
  - ✓ INT0, INT1 e INT2
    - Pueden activarse por flanco de subida o de bajada, o por nivel de cero lógico
    - También se puede generar por software, si son configurados los terminales como salidas
- Temporizadores / Contadores
  - ✓ Timer0 y Timer2 de 8 bits Timer1 de 16 bits

- Dispone de unidades comparadoras
- Sirven como Generadores de Frecuencias
- Poseen relojes pre escalables de 10 bits
- Permiten implementar Moduladores por Ancho del Pulso

- Interface Serial con dos Líneas
  - ✓ TWI
    - Operación maestro / esclavo
    - Puede trabajar como transmisor o como receptor
    - Velocidad de transferencia hasta 400 KHz.
    - Longitud de la dirección de 7 bits para 127 esclavos
- Receptores / Transmisores Universales Sincrónicos y Asincrónicos
  - ✓ USART0 y USART1
    - Full dúplex
    - Velocidad de alta resolución
    - Tramas de 5, 6, 7, 8 o 9 bits, con 1 o 2 bits de parada
    - Detector de errores de velocidad y en la trama
    - Operación de maestro o esclavo en comunicaciones sincrónicas
- Salida del Reloj
  - ✓ Clock
    - Habilitación de la señal programando el fusible
    - Incluye como fuente al oscilador interno RC
    - Se puede utilizar el sistema pre escalable para realizar la división del reloj
- Interrupciones por Cambio de Estado
  - ✓ Interrupciones Externas Adicionales
    - Cambios entre PCINT0 y PCINT7 se registra en PCI0
    - Cambios entre PCINT8 y PCINT15 se registra en PCI1
    - Cambios entre PCINT16 y PCINT23 se registra en PCI2
    - Cambios entre PCINT24 y PCINT31 se registra en PCI3
- Memorias en ATmega164P

Descripción de la Memoria figura 1.14.

DEVICE	FLASH	EEPROM	RAM
ATmega164P	16Kbytes	16Kbytes	1Kbytes

Figura 1. 14 Descripción de la Memoria

1.3.1.14 Diagrama de Bloque

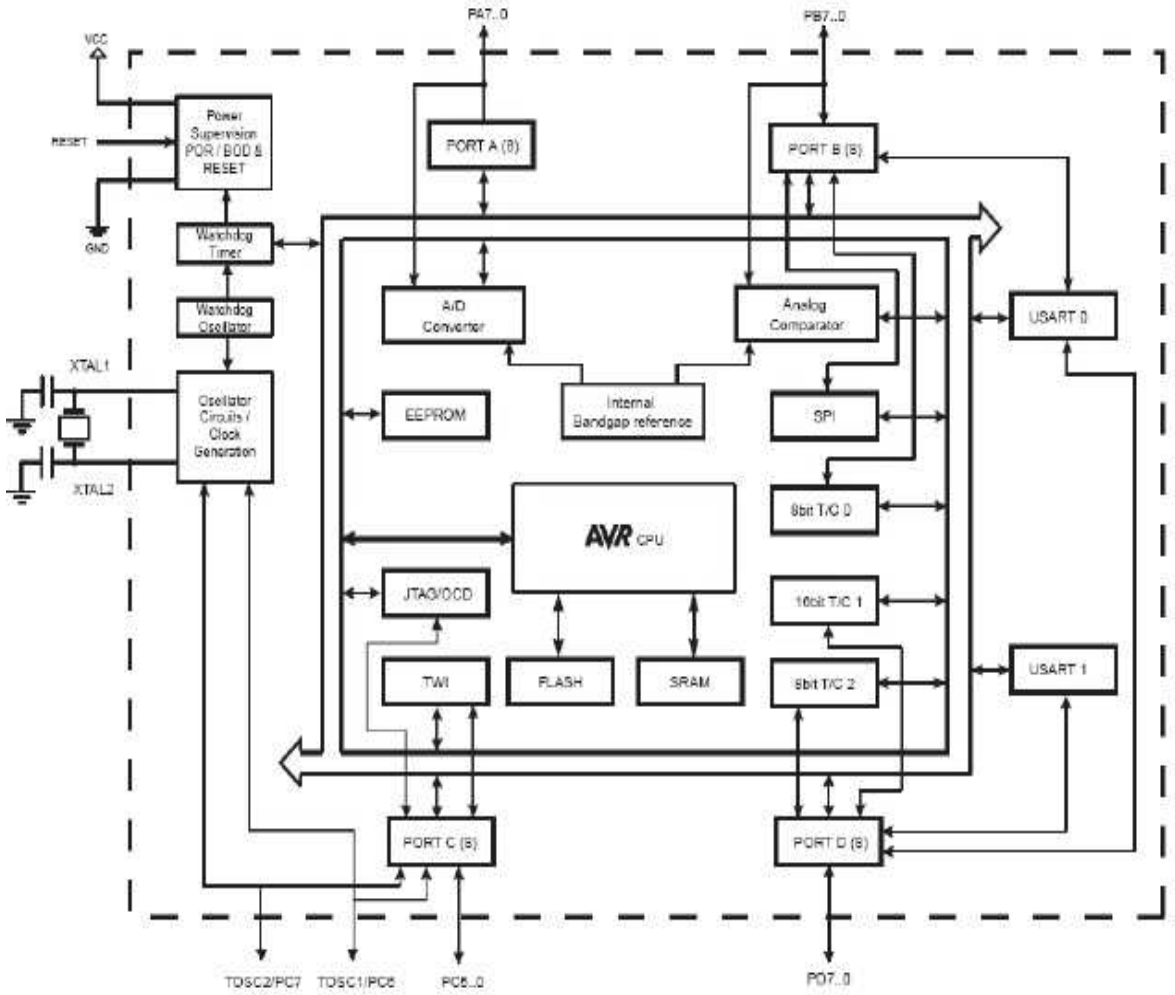


Figura 1. 15 Diagrama de Bloques del AVR 164P

El core (núcleo) AVR combina un conjunto de instrucciones RISC con 32 registros para uso de propósito general. Todos los 32 registros están directamente relacionados con la Unidad Aritmética Lógica (ALU), admitiendo dos registros independientes al ejecutarse una instrucción en un ciclo de máquina. El resultado de esta arquitectura es más eficiente, se consigue un caudal de flujo y

transferencia hasta diez veces más rápido que microcontroladores CISC convencionales.

El ATmega164P / 324P / 644P provee las siguientes características: 16/32 / 64K bytes en el sistema de Flash Programable con capacidad de lectura y escritura de 512B/1K/2K bytes en la EEPROM, 1/2/4K bytes en la SRAM, 32 pines de E/S para propósito general, 32 registros de propósito general, Contador en Tiempo real (RTC), tres Timer/Contadores.

1.4 SENSORES

Un sensor es un dispositivo que detecta manifestaciones de fenómenos fiscos; está diseñado para recibir información del exterior y transformarla en una magnitud, normalmente eléctrica, que sea factible cuantificar y manipular por un sistema eléctrico.

Pueden ser de indicación directa o estar conectados a un indicador (posiblemente a través de un convertidor analógico a digital, una computadora y un display de modo que los valores muestreados puedan ser leídos por un humano.

Los sensores típicos para un Sistema de Seguridad son:

1.4.1 DE PRESENCIA.

Este tipo de sensores tienen como finalidad determinar la presencia de un objeto en un intervalo de distancia especificado. Algunos de los tipos más importantes son: inductivos, de efecto Hall, capacitivos, ultrasónicos e infrarrojos. Su salida puede ser normalmente abierta (NA) o normalmente cerrada (NC). El principio de funcionamiento está basado en la diferencia (moderada) entre la temperatura del ambiente y la temperatura del cuerpo humano. Cuando la temperatura del cuerpo humano difiere de la temperatura del ambiente, la radiación generada por el intruso es empleada para activar una alarma.

1.4.2 DE HUMO

Estos sensores detectan el humo dentro de un área en específico y cambian su estado ante la presencia de este. Entre sus principios de funcionamiento existen dos tipos fundamentales, los de cámara de ionización y los fotoeléctricos. El primero detecta la descompensación de conductividad producida por el humo

entre dos células: la de medida y la de referencia. El segundo detecta radiaciones provenientes de las llamas.

1.4.3 DE APERTURA DE PUERTAS O VENTANAS.

Este tipo de sensores consisten en un contacto magnético. El contacto magnético consta de un imán y normalmente el imán cierra el contacto con el sensor. Si se abre una puerta o una ventana, el contacto se abrirá y enviará una señal de alteración en el estado del sensor.

1.4.4 DE RUPTURA DE CRISTALES.

Existen dos clases de sensores para la ruptura de cristales. El primero detecta la vibración en el vidrio y el segundo el sonido que produce un cristal al romperse; este tipo es recomendable porque hay tónicas especiales para quebrar un vidrio sin producir impacto.

1.5 PANTALLA LCD

[12]Independientemente de la marca de display LCD, todos traen internamente un microprocesador HITACHI 44780.

El módulo LCD se comunica con el microcontrolador a través de comandos de 8 bits. Estos comandos son controlados a través de tres líneas de control, que le indican al módulo si los datos que se le envían son comandos o caracteres y si se lee o escribe el módulo.

Poseen una línea de habilitación lo que permite utilizar la misma conexión paralela para otros dispositivos. Si transmitimos en 4 bits se utilizan las líneas de datos D7 a D4 y las líneas D3 a D0 no se utilizan. La comunicación se hace transmitiendo primero los 4 bits de mayor peso y luego los 4 bits de menor peso.

1.5.1 LCD 20X4

[13] Interfaz para aplicaciones con microcontroladores, que permite visualizar 40 caracteres alfanuméricos en cuatro líneas de 20 caracteres. Posee incorporado una tabla completa de caracteres ASCII y además puede operar con caracteres definidos por el usuario. La tarjeta cuenta con una entrada de datos de tipo serial

[12] http://www.webelectronica.com.ar/news32/nota06.htm
[13] http://www.caveo.com.ar/lcd_20x4_sinc.htm

sincrónica con comando por 3 líneas: clock/data/strobe. La información al display se envía carácter a carácter y no requieren ser refrescados periódicamente. Es decir, una vez enviado un carácter al display, este permanece estático hasta no recibir una nueva orden. El display posee un gran número de funciones que permiten desplazar los caracteres en ambos sentidos, hacer parpadeos, posicionar el cursor un lugar específico, etc. Otra característica destacable es la posibilidad de encender una luz interna del display para poder ser utilizado sin iluminación ambiente. Todas estas características hacen de este interfaz el dispositivo óptimo para representación de información para un sistema microprocesado. Es importante destacar que de las tres líneas de comando dos de ellas (clock y data) pueden ser compartidas por otros dispositivos conectados al microcontrolador. Solo la línea de strobe es exclusiva del interface. Este interfaz utiliza la librería en 'C' <lcdser.h>

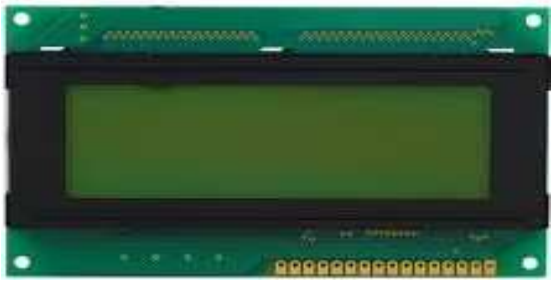


Figura 1. 16 LCD 20x4

1.6 TECLADO MATRICIAL 4X4[14]

Dispositivo de entrada de datos que consta de 16 teclas o pulsadores, dispuestos e interconectados en filas y columnas. Dispone de un conector SIL (Single In Line) macho de 8 pines que se corresponden con las 4 filas y las cuatro columnas de las que dispone figura 1.17.

[14] http://www.x-robotics.com/rutinas.htm

Figura 1. 17Teclado Matricial

1.6.1 FUNCIONAMIENTO:

En la siguiente figura1.18 vemos el esquema de conexionado interno del teclado matricial y sus correspondientes pines de salida numerados de izquierda a derecha mirando el teclado tal y como se ve en la foto anterior. Cuando se presiona un pulsador se conecta una fila con una columna, teniendo en cuenta este hecho es muy fácil averiguar que tecla fue pulsada. También podemos ver el conexionado típico con el puerto B del µC PIC. Las resistencias de 2k2 son necesarias para poder compartir el puerto del pic independientemente del teclado y por ejemplo poder conectar un LCD o una barra de leds al mismo tiempo. Durante la fase de lectura del teclado la mitad de la puerta B es configurada como entrada y la otra mitad como lectura y durante la escritura en el LCD u otro sistema, la puerta B es configurada como salidas. Entonces se podrían cortocircuitar accidentalmente las salidas de los puertos provocando su destrucción, si pulsásemos alguna tecla es ese momento con lo que al poner estas resistencias evitamos este hecho y así si se produjera el cortocircuito tan solo circularía una pequeña corriente y el puerto del µC no correría ningún riesgo.

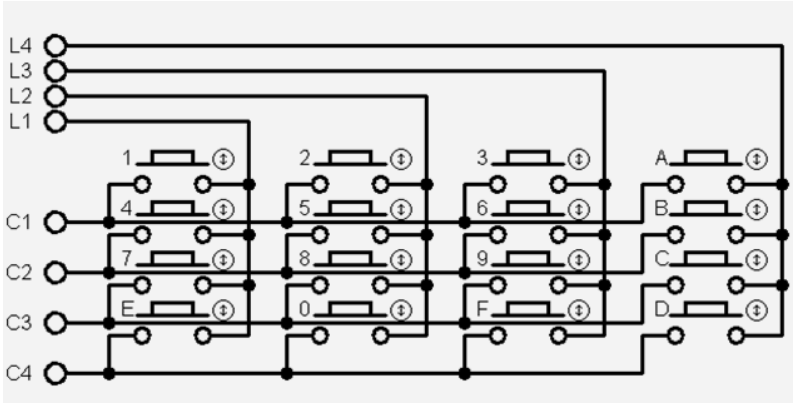


Figura 1. 18 Conexión del teclado

1.7 CERRADURA

Una cerradura es un mecanismo de metal que se incorpora a puertas y cajones de armarios, cofres, arcones, etc., para impedir que se puedan abrir sin la llave y así proteger su contenido.

Este mecanismo se puede accionar mediante una llave de metal, normalmente acero. En la actualidad, aparte de las cerraduras mecánicas, existen otras como el electro-mecánico o las electrónicas, en las que la llave puede ser una tarjeta de plástico o PVC. La llave encaja en la cerradura por el llamado "ojo", que es un agujero situado normalmente en la parte central del cilindro de la cerradura.

1.7.1 CERRADURA ELECTRÓNICA

Cerradura electrónica (más exacto una cerradura eléctrica) es a fijación del dispositivo cuál funciona por medio de corriente eléctrica. Las cerraduras eléctricas son a veces independientes con una asamblea de control electrónico montada directamente a la cerradura. Las cerraduras más a menudo eléctricas están conectadas con control de acceso sistema. Las ventajas de una cerradura eléctrica conectaron con un sistema de control de acceso incluyen: control dominante, donde las llaves pueden ser agregadas y ser quitadas sin la reintroducción del cilindro de la cerradura; control de acceso fino, donde están factores el tiempo y el lugar; y transacción que registra, donde se registra la actividad.

1.7.2 CERRADURA ELECTROMAGNÉTICA.

Su estructura se basa en un electroimán que se encuentra vinculado a una placa que al mismo tiempo logra hermanarse en el cierre.

Lo que hace dicha placa es fijarse a la puerta en cuestión, mientras que paralelamente, el cuerpo principal se va instalando del lado interior de la misma y encima del marco. Al mismo tiempo el sistema posee un control de acceso el cual provee energía de aproximadamente unos 12 VCC, lo que provoca la atracción de la placa que se encuentra firmemente fijada en la puerta, logrando que la misma pueda resistir empujes de hasta 500 kg, según el modelo de cerradura electromagnética del que estemos hablando. Ahora bien, muchas personas se preguntan qué tan seguras suelen ser las cerraduras electromagnéticas y para contestar esta pregunta debemos decir que para lograr abrir esta cerradura únicamente deberemos quitar el suministro eléctrico

1.8 RELÉ[15]

El relé o relevador, es un dispositivo electromecánico. Funciona como un interruptor controlado por un circuito eléctrico en el que, por medio de una bobina y un electroimán, se acciona un juego de uno o varios contactos que permiten abrir o cerrar otros circuitos eléctricos independientes. Fue inventado por Joseph Henry en 1835.

Dado que el relé es capaz de controlar un circuito de salida de mayor potencia que el de entrada, puede considerarse, en un amplio sentido, como un amplificador eléctrico. Como tal se emplearon en telegrafía, haciendo la función de repetidores que generaban una nueva señal con corriente procedente de pilas locales a partir de la señal débil recibida por la línea. Se les llamaba "relevadores". De ahí "relé".

1.9 EL RTC DALLAS DS1307

[16]El semiconductor Maxim/Dallas DS1307 como muestra la figura 1.19 es un reloj de tiempo real exacto, el cual automáticamente, mantiene el tiempo y la fecha

[15] http://es.wikipedia.org/wiki/Rel%C3%A9

actual, incluyendo compensación para meses con menos de 31 días y saltos de año. El DS1307 es un dispositivo de 8 pines al que se le conecta un cristal de cuarzo estándar, de bajo costo, a 32.768kHz entre los pines 1 y 2 para proveer tiempo base exacto.

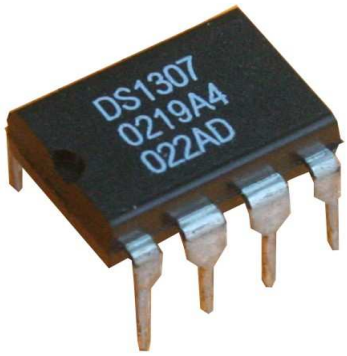


Figura 1. 19 DS1307

Opcionalmente se le puede conectar al pin3, baterías de respaldo de 3 volt, asegurando que se mantendrá el tiempo a la fecha aunque esté desconectada la fuente de tensión del circuito principal. El circuito integrado automáticamente detecta que se ha removido la energía en el circuito principal y se conectan las baterías de respaldo cuando es requerido.

La batería de respaldo puede durar hasta 10 años y se coloca en la misma base de circuito impreso, tal como muestra la figura 1.20

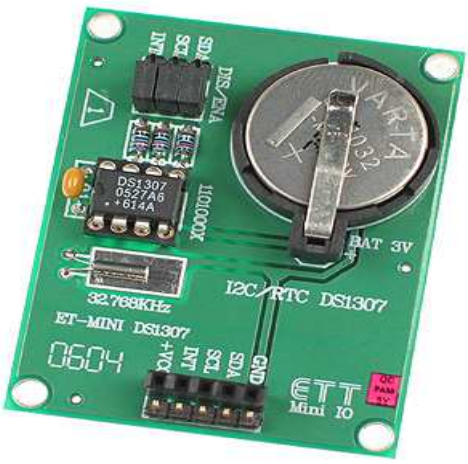


Figura 1. 20 Batería de respaldo del Dallas DS1307

[16] http://www.clubse.com.ar/download/pdf/notasrevistas08/nota01.htm

Adicionalmente el circuito integrado DS1307 tiene dos características interesantes. El pin 7 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. El circuito integrado también tiene 56 bytes de memoria RAM para propósito general, el cual puede ser usado como memoria extra.

1.10 DISPOSITIVO SONORO

Sirena es un instrumento acústico. La versión original produciría sonidos bajo el agua, lo que sugiere un vínculo con las sirenas de la mitología griega. Otras sirenas más modernas son las de defensa civil o ataque aéreo, sirenas de tornado o las sirenas de emergencia en los vehículos de servicio, tales como ambulancias, coches de policía y camiones de bomberos. Hay dos tipos generales, neumáticas y electrónicas.

Una sirena es un aparato generador de sonidos mediante las interrupciones periódicas de una corriente de aire o vapor, por uno o más discos con agujeros situados formando un círculo. La sirena emite un sonido de frecuencia igual al producto del número de orificios por el de revoluciones.

Encontramos los siguientes tipos:

1.10.1 SIRENA ELECTRÓNICA.

La sirena electrónica se compone de una unidad de control que ha almacenado en el interior de la secuencia de tonos, y uno o dos altavoces conectados a esta unidad. El uso de sirenas electrónica está muy extendido, siendo especialmente adecuadas para su funcionamiento continuo, también tienen un bajo consumo eléctrico y no requieren mantenimiento.

La sirena electrónica de última generación utiliza altavoces muy potentes (neodimio), que permiten una mayor audibilidad y, por tanto, una mayor eficacia. Algunos incluso han llegado a 200 vatios de potencia cada uno, y se pueden integrar en el techo del vehículo o en el motor.

1.10.2 SIRENA ELECTRONEUMÁTICA.

La sirena electro-neumático tiene dos o más trompetas de membrana, alimentadas por un compresor equipado con un dispositivo rotativo que gestiona la secuencia de tonos. Este tipo de sirena requiere mantenimiento periódico, ya que el compresor requiere ser lubricado con aceite y se debe comprobar el nivel, para evitar un sobrecalentamiento de la unidad. Las versiones actuales están equipadas con un compresor para servicio continuo, a diferencia de los modelos más antiguos que tienden a recalentarse después de varios minutos de funcionamiento ininterrumpido.

1.10.3 SIRENA MECÁNICA.

La sirena mecánica es un cuerpo único, que incluye dentro un motor conectado a un ventilador, que genera un sonido largo y agudo (silbato), que se ajusta de forma automática a través de relé intermitente; a veces también incluye un botón de control manual.

1.11 FUENTES DE ALIMENTACIÓN

En la mayoría de las aplicaciones se requiere una tensión fija y estable de un valor determinado. Un regulador de voltaje proporciona un voltaje de salida de DC constante que es prácticamente independiente del voltaje de entrada, la corriente de carga de salida y la temperatura. El regulador de voltaje forma parte de una fuente de alimentación. Su voltaje de entrada proviene de la salida filtrada de un rectificador derivada de un voltaje de AC o de una batería en el caso de sistema portátiles[17]figura 1.21.

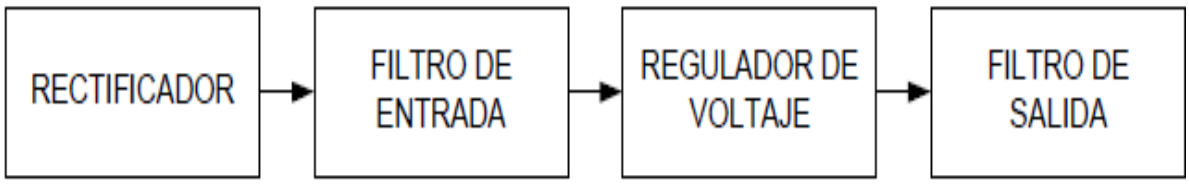


Figura 1. 21 Diagrama de bloques de una fuente de alimentación

[17] http://www.foxitsoftware.com

Existen muchos tipos de reguladores de circuitos integrados, los tipos más populares de reguladores lineales son el regulador de voltaje fijo de tres terminales y el regulador de voltaje ajustable de tres terminales.

Dentro de los reguladores de voltaje fijo están los LM78XX, cada uno de estos dispositivos posee sólo tres terminales, uno corresponde a la entrada de tensión no regulada, otra es la salida regulada y la restante es la masa común a ambas.

Resumiendo, y para comprender completamente la simplicidad de una fuente de alimentación de este tipo, solo basta observar el diseño de la figura 1.22

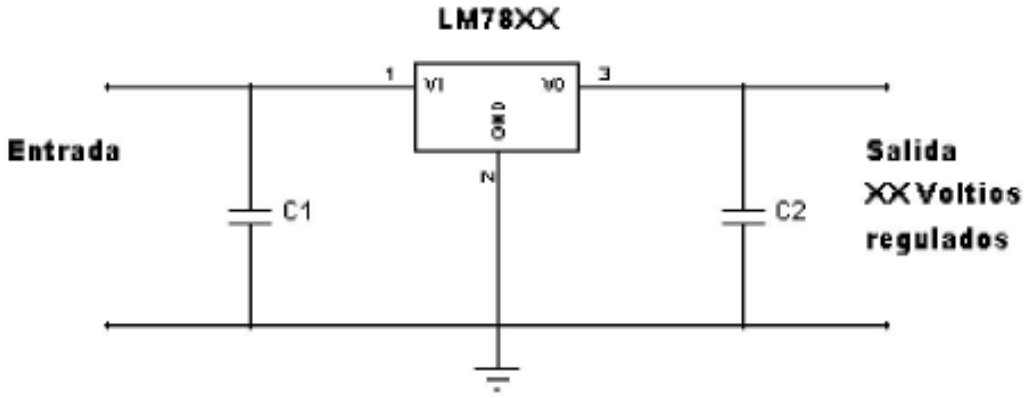


Figura 1. 22 Diseño una fuente de alimentación regulada con el LM78XX

Como se puede observar se requiere agregar dos capacitares al circuito integrado. La función de cada uno de ellos es: C1, que se halla a la entrada de regulador, filtra la tensión de posibles transitorios y picos indeseables, mientras que C2, que se encuentra a la salida, disminuye la tensión de rizado de salida, a la vez que evita oscilaciones.

Esta serie de reguladores son capaces de producir corriente de salida superior a 1A cuando se usa con un disipador de calor apropiado, en cuanto al voltaje de entrada debe ser por lo menos 2V mayor que el voltaje de salida a fin de mantener la regulación.

1.12 ICSP

Este es un protocolo para poder grabar los en el propio circuito sin tener que desoldar nada. ICSP significa “In Circuit Serial Programming“.[18]

[19]Los microcontroladores se graban mediante un método llamado ICSP (in circuit serial programing), por el cual se puede grabar la memoria de programa, la memoria de datos y la palabra de configuración.

Para realizar la grabación, el PIC debe estar en modo monitor. Existen varias maneras de entrar en este modo, que dependen del microcontrolador usado.

El método más general, que consisten en introducir una tensión de 12 voltios por la pata MCLR. (El otro método es el denominado de bajo voltaje. Hay que introducir 5 voltios por la pata RB3.

En esta figura 1.23 se muestra un ejemplo de un circuito para hacer que el microcontrolador entre en modo monitor. Hay que introducir 12v por la pata MCLR. Cada vez que se pulse (y suelte) el botón de reset, el microcontrolador entrará en modo monitor, por lo que se tendrá acceso a los servicios de grabación.

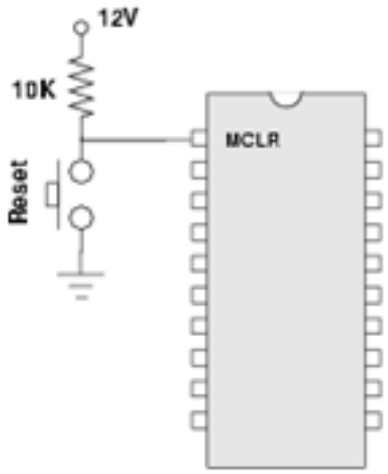


Figura 1. 23 Circuito para hacer que el microcontrolador entre en modo monitor

[18] http://elbaroncojo.stream18.com/microcontroladores/
[19] http://www.iearobotics.com/proyectos/cuadernos/ct4/ct4.html

1.12.1 ARQUITECTURA

Una vez en modo monitor, se tiene acceso a una serie de servicios, a través del protocolo ICSP. Este protocolo se describe a dos niveles como se muestra en la figura 1.24: nivel físico y nivel de comandos. En el nivel físico se especifica cómo se transmiten los bits (temporizaciones, sincronismo, etc.) y en el de comandos qué tramas hay que enviar para tener acceso a los diferentes servicios.

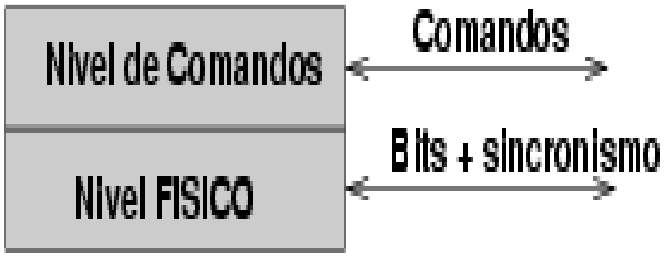


Figura 1. 24 Niveles del Protocolo

1.12.1.1 Nivel de Comandos

Los comandos son proporcionados por el programa con el que se está trabajando el microcontrolador.

1.12.1.2 Nivel físico

Para realizar la comunicación se utiliza un protocolo serie síncrono. Son necesarios dos hilos, del puerto B, uno para llevar los datos (conectado a RB7) y otro para el reloj (conectado a RB6) figura 1.25

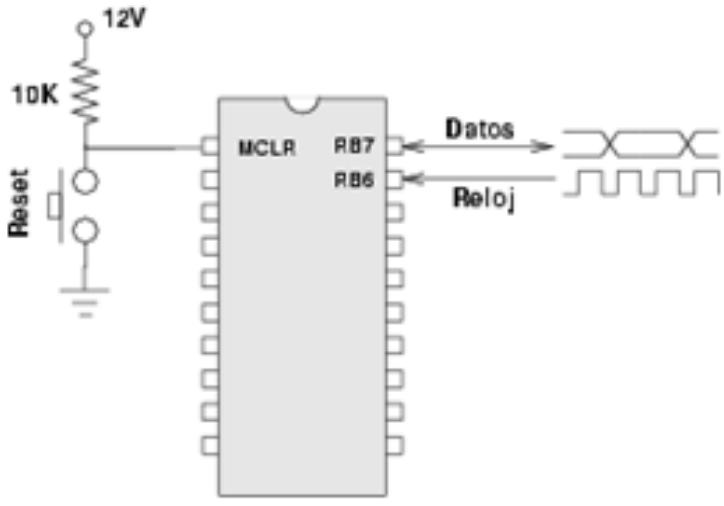


Figura 1. 25 Comunicación

1.12.2 DESCRIPCIÓN DEL ICSP

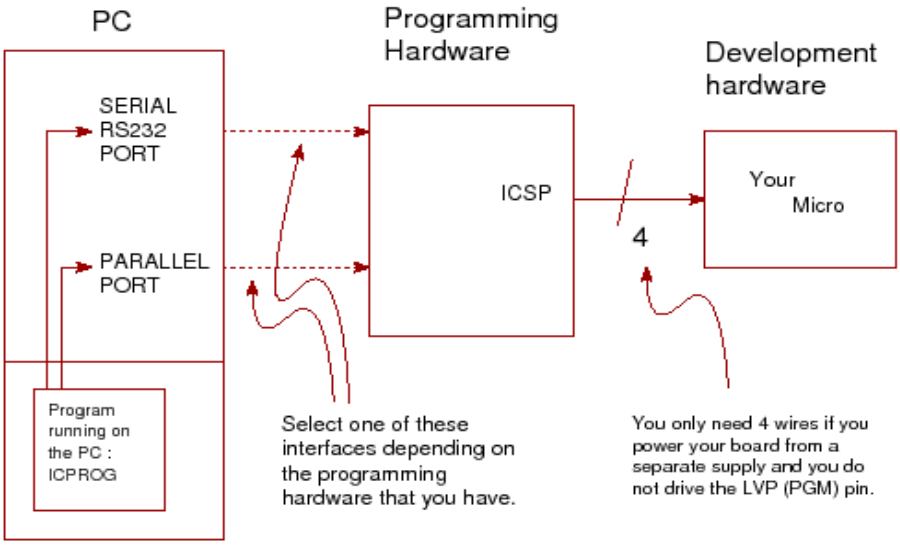


Figura 1. 26 Interface serial usada por el PIC para cargar el programa dentro de la memoria del programa del microcontrolador

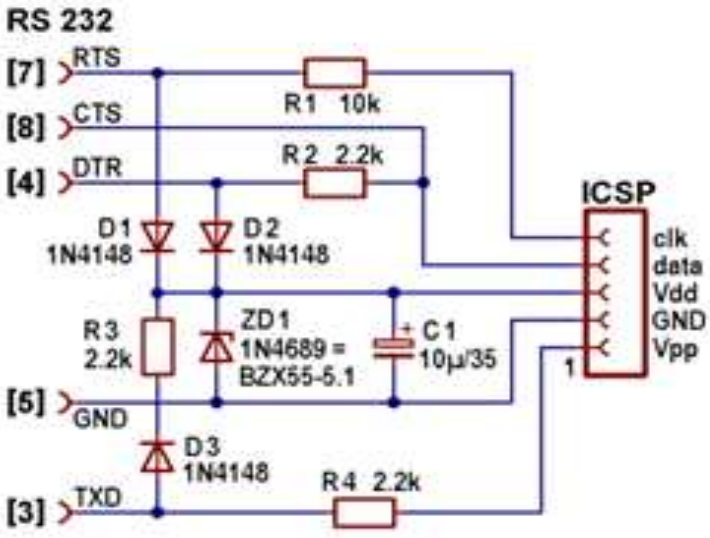


Figura 1. 27 Conexiones

ICSP figura 1.27 es un conector que disponen algunas placas mediante el cual podemos actualizar o reprogramar el chip sin sacarlo del zócalo de donde lo tenemos colocado en un proyecto, algunos programadores disponen de este

conector para unir el programador con la placa que contiene el chip a programar. El conector, tiene:

- VPP/MCLR: 13v o 13.5v (máximo 14v) terminal para poner el microcontrolador en modo programación (13v) y resetear el PIC (MCLR)
- VCC: 5v
- GND: 0v
- PGC /CLOCK: sincronización
- PGD/DATA: Datos a transferir
- PGM – LVP: señal de programación de bajo voltaje

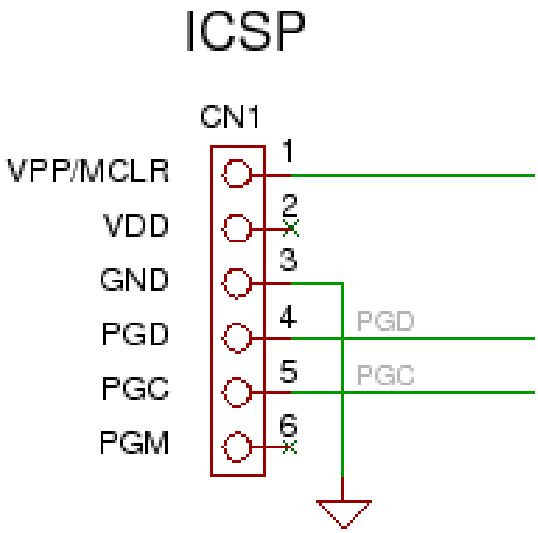


Figura 1. 28 Distribución de pines

1.12.3 CÓMO USAR ICSP PARA PROGRAMAR UN MICROCONTROLADOR EN EL CIRCUITO

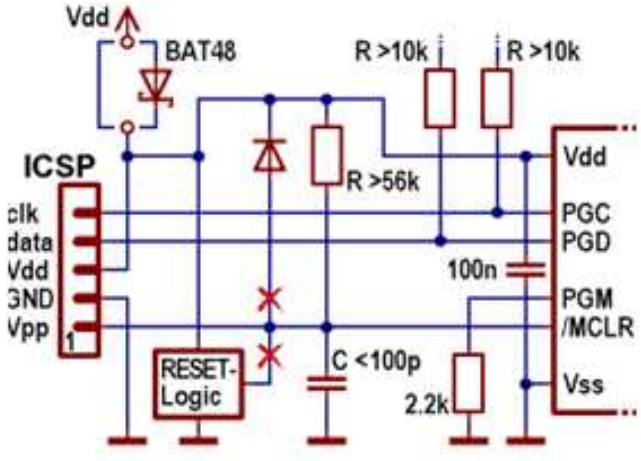


Figura 1. 29 Conexión con el Microprocesador

ICSP ofrece la posibilidad de programar un microcontrolador en un circuito de aplicación. Esto es especialmente ventajoso en ámbitos educativos o de desarrollo, donde es necesario pasar repetidamente de programar el PIC a testear el circuito, y viceversa. Además de ahorrar tiempo, ICSP evita riesgos al mover el micro entre zócalos distintos como torcer los pines o "atormentar" el PIC por descargas electroestáticas

Para poder usar ICSP, el circuito no debe distorsionar las señales de programación, ni las señales de programación deberían afectar el circuito. Las condiciones principales para programar un micro son:

- El voltaje de programación *Vpp* aplicado a /MCLR debe cambiar entre cero y 13 voltios en pocos microsegundos.
- Las señales de reloj y de datos deben alcanzar los niveles extremos (*Vdd* o *GND*) en menos de un microsegundo.
- Algunos PIC usan un pin (RB3 o RB4) como señal PGM para el modo LVP (*Low Voltage Programming*). Este pin debe permanecer en *Low* durante la programación.
- Algunos micros requieren que el voltaje de programación *Vpp* se aplique antes del voltaje operativo *Vdd*. En este caso *Vdd* debe ser controlado por el módulo de programación.

La manera más simple de satisfacer las dos primeras condiciones es no usar los pines correspondientes para el circuito de aplicación. Si se quiere usar el depurador, esto es incluso una necesidad.

1.13 MEMORIA 24LC512

Es una memoria serie del tipo 64K x 8 E^2PROM, es decir, una memoria eléctricamente borrable de 512 Kbits, capaz de operar en un amplio rango de tensiones (2.5 a 5.5v) y a una frecuencia máxima de reloj de 400KHz.

Este dispositivo permite realizar lecturas aleatorias y secuenciales de hasta 512 Kbits, es decir, el espacio total direccionable que puede ser ampliado hasta 4 Mbits mediante la utilización de tres líneas adicionales de direccionamiento que permiten conectar hasta ocho dispositivos en un mismo bus.

1.13.1 MODOS DE FUNCIONAMIENTO DE 24LC512

El 24LC512 puede operar de dos modos diferentes en cuanto a transmisión de datos se refiere. Un dispositivo que envía datos sobre el bus se define como transmisor mientras que un dispositivo que recibe datos a través del mismo se define como receptor. El 24LC512 puede funcionar como emisor o bien como receptor.

Típicamente el 24LC512 funcionará como parte de un sistema master-slave, donde el papel de maestro será desempeñado usualmente por un microprocesador y la memoria estudiada será el eslavo. El maestro controlará el bus, a través del cual memoria y microprocesador quedan comunicados, generando la señal de reloj (SCL), controlando el acceso al mismo y generando la señalización necesaria como las condiciones de comienzo y de parada.

1.14 COMUNICACIÓN SERIAL

La comunicación serial consiste en el envío de un bit de información de manera secuencial, esto es, un bit a la vez y a un ritmo acordado entre el emisor y el receptor.

La comunicación serial en computadores ha seguido los estándares definidos en 1969 por el RS-232 (Recommended Standard 232) que establece niveles de voltaje, velocidad de transmisión de los datos, etc. Por ejemplo, este protocolo establece un nivel de -12v como un uno lógico y un nivel de voltaje de +12v como un cero lógico (por su parte, los microcontroladores emplean por lo general 5v como un uno lógico y 0v como un cero lógico).

Existen en la actualidad diferentes ejemplos de puertos que comunican información de manera serial (un bit a la vez). El conocido como “puerto serial” ha sido gradualmente reemplazado por el puerto USB (Universal Serial Bus) que permite mayor versatilidad en la conexión de múltiples dispositivos. Aunque en naturaleza serial, no suele referenciarse de esta manera ya que sigue sus propios estándares y no los establecidos por el RS-232.

La mayoría de los microcontroladores poseen un puerto de comunicación serial. Para comunicarse con los computadores personales actuales que poseen únicamente puerto USB requieren de un dispositivo “traductor”. Arduino emplea el integrado FT232R, el cual es un convertidor USB-Serial. A través de este integrado el microcontrolador puede recibir y enviar datos a un computador de manera serial.

La parte física encargada de la comunicación serial es la UART (Universal Asynchronous Receiver and Transmitter). Los microcontroladores Atmega8/168/328, en los cuales está basado Arduino, disponen de un dispositivo compatible llamado USART (Universal Synchronous and Asynchronous serial Receiver and Transmitter) que permite tanto la comunicación asincrónica como sincrónica.

En la comunicación asincrónica figura 1.30, la velocidad de envío de los datos es acordada a priori entre el emisor y el receptor. En la comunicación sincrónica, el envío de los datos es sincronizado por el emisor a partir de un pulso constante de reloj (Clock), con cada pulso envía un nuevo dato.

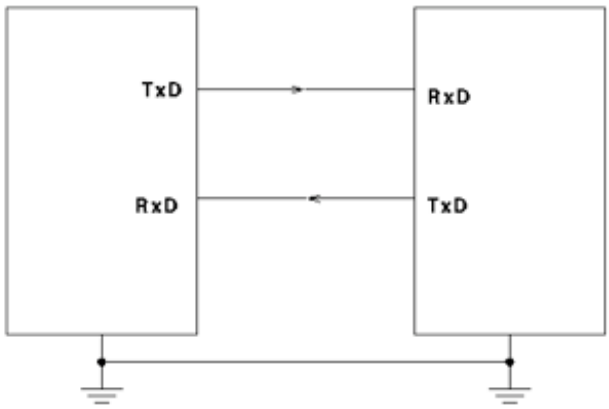


Figura 1. 30 Comunicación asincrónica

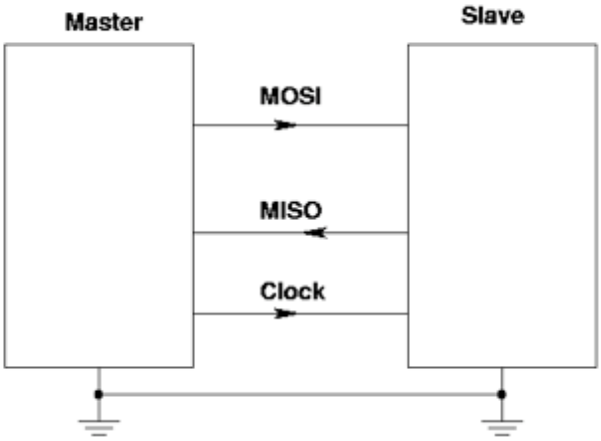


Figura 1. 31 Comunicación sincrónica

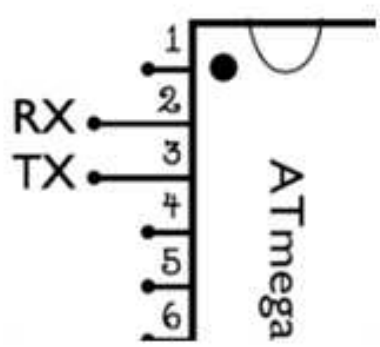


Figura 1. 32 Pines encargados de la comunicación serial en los microcontroladores Atmega

Para comunicación asincrónica requiere de sólo dos líneas de conexión que corresponden con los pines 2 y 3: Pin 2 (Rx) pin de recepción y pin 3 (Tx) pin de transmisión, y del establecimiento de un nivel de tierra común con el computador, esto es, ambas tierras deben estar conectadas, estableciendo el mismo nivel de voltaje de referencia.

Además de realizar las conexiones físicas entre el microcontrolador y el computador, para que pueda establecerse la comunicación serial debe existir un acuerdo previo en la manera que van a ser enviados los datos. Este acuerdo debe incluir los niveles de voltaje que serán usados, el tamaño y formato de cada uno de los mensajes (número de bits que constituirán el tamaño de la palabra, existirá o no un bit de inicio y/o de parada, se empleará o no un bit de paridad), el tipo de

lógica empleada (qué voltaje representará un cero o un uno), el orden en que serán enviados los datos (será enviado primero el bit de mayor peso o el de menor peso) y la velocidad de envío de datos.

1.14.1 NORMA RS232[20]

La Norma RS-232 fue definida para conectar un ordenador a un modem. Además de transmitirse los datos de una forma serie asíncrona son necesarias una serie de señales adicionales, que se definen en la norma. Las tensiones empleadas están comprendidas entre +15/-15 voltios figura 1.33.

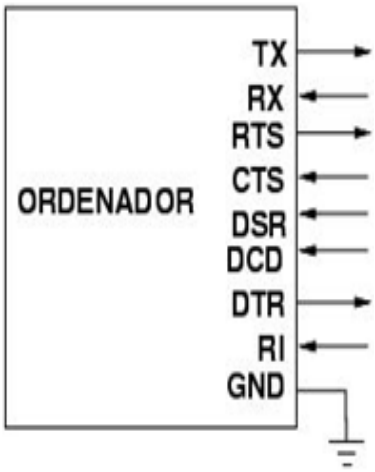


Figura 1. 33 Pines del Ordenador

1.14.2 CONEXIÓN DE UN MICROCONTROLADOR AL PUERTO SERIE DEL COMPUTADOR

Para conectar el PC a un microcontrolador por el puerto serie se utilizan las señales Tx, Rx y GND. El PC utiliza la norma RS232, por lo que los niveles de tensión de los pines están comprendidos entre +15 y -15 voltios. Los microcontroladores normalmente trabajan con niveles TTL (0-5v). Es necesario por tanto intercalar un circuito que adapte los niveles:

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

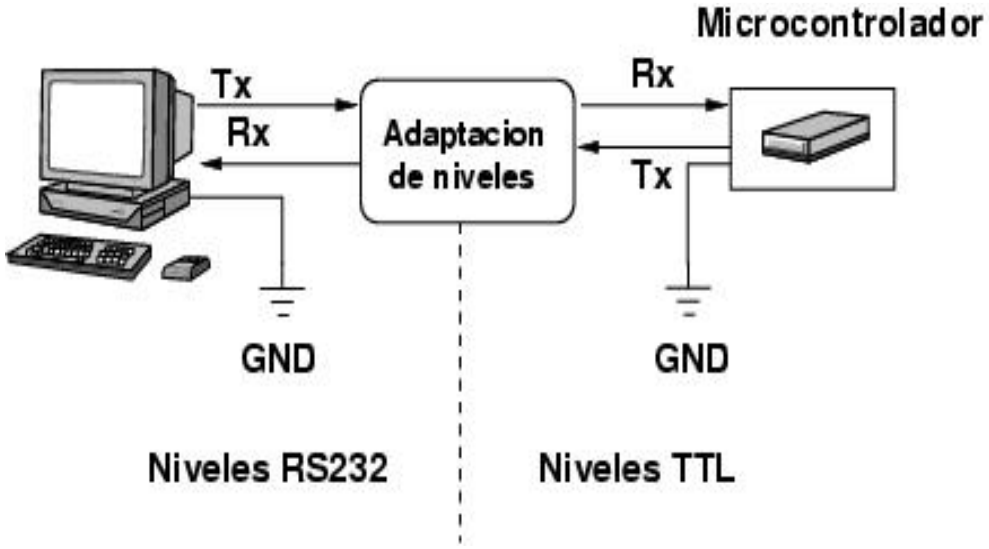


Figura 1. 34 Conexión del PC

Uno de estos circuitos, que se utiliza mucho, es el MAX232.

1.14.3 EL CONECTOR DB9 DEL PC

En los PCs hay conectores DB9 macho, de 9 pines, por el que se conectan los dispositivos al puerto serie. Los conectores hembra que se enchufan tienen una colocación de pines diferente, de manera que se conectan el pin 1 del macho con el pin 1 del hembra, el pin2 con el 2, etc.

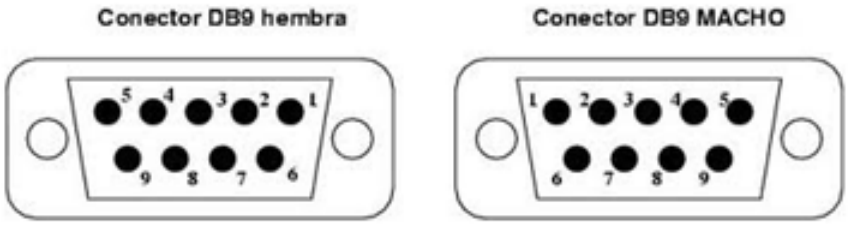


Figura 1. 35 Conector DB9

La información asociada a cada uno de los pines es la siguiente:

Pat.	Nombre	RS232	V.24	Dir	Descripción
1	CD	CF	109	←	Detección de portadora
2	RXD	BB	104	←	Recepción de datos
3	TXD	BA	103	→	Transmisión de datos
4	DTR	CD	108.2	→	Terminal de datos preparado
5	GND	AB	102	—	Tierra
6	DSR	CC	107	←	Dispositivo preparado
7	RTS	CA	105	→	Petición de envío
8	CTS	CB	106	←	Preparado para transmitir
9	RI	CE	125	←	Indicador de llamada entrante

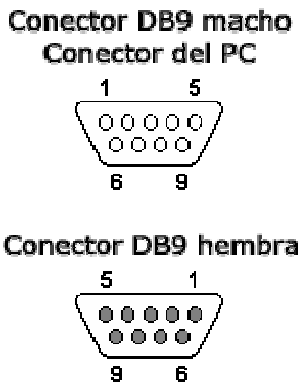


Figura 1. 36 Distribución de Pines del Conector DB9

1.14.4 EL CHIP MAX 232

Este chip permite adaptar los niveles RS232 y TTL, permitiendo conectar un PC con un microcontrolador. Sólo es necesario este chip y 4 condensadores electrolíticos de 22 micro - faradios. El esquema es el siguiente:

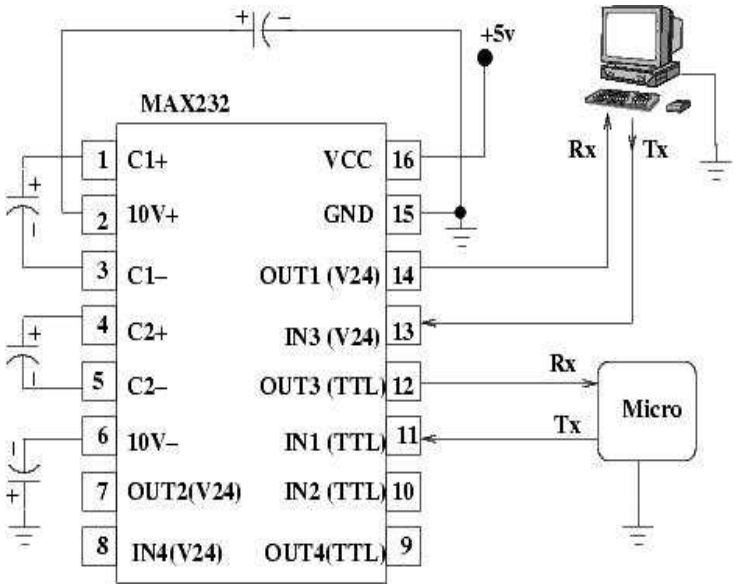


Figura 1.37 Conexión del PC con el Micro a través del MAX 232

1.14.4.1 CIRCUITO INTEGRADO PARA CONVERSIÓN DE NIVELES[21]

El MAX232 es un circuito integrado que convierte los niveles de las líneas de un puerto serie RS232 a niveles TTL y viceversa. Lo interesante es que sólo necesita una alimentación de 5V, ya que genera internamente algunas tensiones que son necesarias para el estándar RS232. Otros integrados que manejan las líneas RS232 requieren dos voltajes, +12V y -12V.

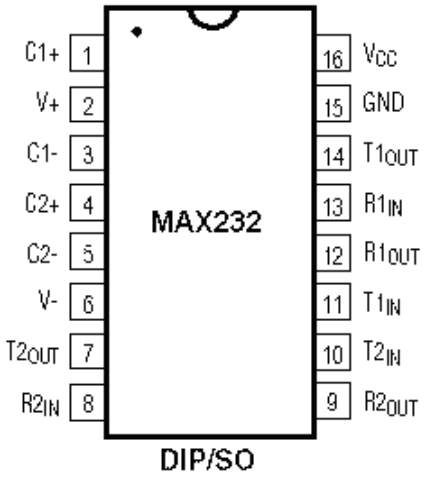


Figura 1. 38 Pines del chip

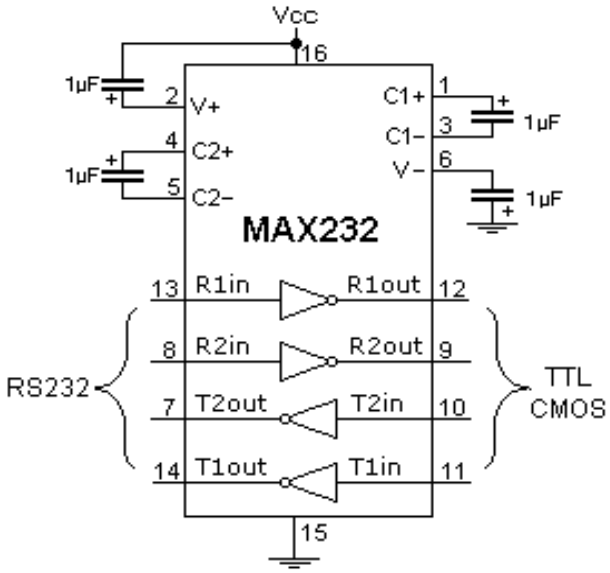


Figura 1. 39 Circuitos Internos del Encapsulado

El MAX232 soluciona la conexión necesaria para lograr comunicación entre el puerto serie de una PC y cualquier otro circuito con funcionamiento en base a

[21] http://axxon.com.ar/rob/Comunicacion_max232.htm

señales de nivel TTL/CMOS. El circuito integrado posee dos conversores de nivel TTL a RS232 y otros dos que, a la inversa, convierten de RS232 a TTL. Estos conversores son suficientes para manejar las cuatro 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, y RTS y CTS se utilizan para establecer el protocolo para el envío y recepción de los datos.

1.14.5 CABLE DE CONEXIÓN

Para realizar la conexión entre el PC y nuestro circuito podemos usar diferentes alternativas. Una manera es utilizar un cable serie macho-hembra no cruzado, y en el circuito un conector hembra db9 para circuito impreso:

Figura 1. 40 Cable serie macho-hembra no cruzado y conector hembra db9 para circuito impreso

Cuando conectamos un micro al PC normalmente sólo usamos los pines TX, RX y GND, sin embargo en este tipo de cables se llevan los 9 pines. Por ello puede resultar útil el utilizar otro tipo de cable.

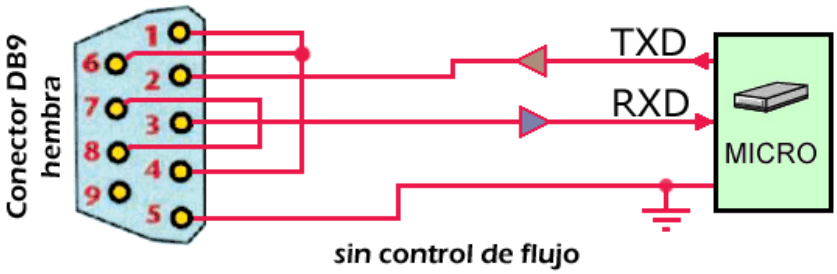


Figura 1. 41 En la placa de circuito impreso donde se encuentra el micro y donde se colocará el conector DB9

También se puede fabricar un cable serie utilizando cable plano de bus, conectando un conector db9 hembra para bus:

Figura 1. 42 Conector db9 hembra para bus

1.15 INTRODUCCIÓN AL BASCOM AVR

El BASCOM AVR es un compilador de BASIC para la familia AVR de ATMEL, desarrollado por la empresa Holandesa MCS Electronic. Ha sido desarrollado sobre W95/98/NT y dispone de todas las características de la familia BASCOM

1.15.1 INSTRUCCIONES BÁSICAS

1.15.1.1 $regfile

Esta instrucción va al inicio de cada proyecto que realicemos, ya que esta se encarga de direccionar el respectivo microprocesador que vamos a utilizar.

Por ejemplo:

ATMEGA 164P **$regfile**=”m164P.dat”

1.15.1.2 $crystal

Esta instrucción va a especificar la frecuencia de oscilación con la que va a trabajar nuestro microcontrolador.

Por ejemplo:

$crystal=8000000 para 8Mhz

1.15.1.3 Config

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

Por ejemplo:

Config portb=output	Puerto B como salida
Config pina.0=input	Pin a.0 como entrada
Config LCD=20x4	LCD de 20 caracteres y 4 líneas

1.15.1.4 Wait, Waitms, Waitus

Esta instrucción sirve para crear un retardo, ya sea en segundos, milisegundos y microsegundos respectivamente.

Por ejemplo:

Wait 3	Espera 3 segundos
Waitms 4	Espera 4 milisegundos
Waitus 5	Espera 5 microsegundos

1.15.1.5 Do – Loop

Esta instrucción es un lazo cerrado, en el cual se ejecuta un conjunto de instrucciones de forma indeterminada

1.15.1.6 Do – Until

Es un lazo definido por la condición de una variable que está dentro del lazo, la cual define cuando termina de ejecutarse el conjunto de instrucciones.

Por ejemplo:

Do	
A=a+1	
Loop Until a=10	Termina el lazo cuando a es igual a 10

1.15.1.7 Toogle

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

Por ejemplo:

Toogle Portb.0 Complementa el puerto B.0

1.15.1.8 Dim

Sirve para dimensionar el tipo de variable que se va a utilizar, entre los tipos están los siguientes:

TIPO	DIMENSION
Bit	0 - 1
Byte	0 a 255
Word	0 a 65535
Long	-2147483648 a 2147483647
Integer	-32768 a 32767
Single	1.5×10^{-45} x 3.4×10^{38}
String	Cadena de caracteres máximo 254
Array	Matriz 65535
Double	5.0×10^{324} a 1.7×10^{308}

Figura 1. 43 Dimensiones de Variables de BASCOM AVR

1.15.1.9 Alias

Sirve para dar un nombre general dentro de un proyecto, ya sea a un puerto o a un pin de un puerto.

Por ejemplo:

Foco Alias Portb.0 El puerto b.0 ahora se llamará foco.

1.15.1.10 LCD (Display de Cristal Líquido)

En el caso del BASCOM AVR podemos controlar el LCD de dos maneras: por comandos o por configuración en cuadro de diálogo.

1.15.1.10.1 Mediante Comandos

1.15.1.10.1.1 Config Lcd

Sirve para configurar la clase de LCD que vamos a utilizar, puede ser de 16 caracteres por 2 líneas (16x2), de 20 caracteres por 4 líneas (20x4), etc.

Por ejemplo:

Config Lcd 20x4 (Display de 20x4)

1.15.1.10.1.2 Config Lcdpin

Sirve para configurar pines por los cuales se va a manejar la información la clase de LCD que vamos a utilizar, puede ser de 16 caracteres por 2 líneas (16x2), de 20 caracteres por 4 líneas (20x4), etc.

Por ejemplo:

Config Lcdpin= Pin, Db4=Porta.4, Db5=Porta.5, Db6=Porta.6, Db7=Porta.7, E=Porc.7, Rs=Portc.6

1.15.1.10.1.3 Config Lcdbus

Esta instrucción sirve para configurar cual será el modo de envío de datos, ya que puede ser hecho por 4 pines u 8 pines.

Por ejemplo:

Config Lcdbus=4 (4 pines de datos)

1.15.1.10.1.4 Lcd “”

Sirve para escribir cualquier frase en el LCD, sin importar la localización del cursor.

Por ejemplo:

Lcd=”HOLA”

1.15.1.10.1.5 Locate x,y

Sirve para localizar el cursor en la línea y columna adecuada, para poder empezar a escribir en el LCD.

Por ejemplo:

Locate=1,1 (Localización del cursor en la fila 1, columna1)

1.15.1.10.1.6 Shiftlcd

Sirve para mover todo el texto del LCD, ya sea para izquierda o derecha, con las instrucciones:

Shiftlcd left

Shiftlcd right

1.15.1.10.2 Mediante cuadro de dialogo tenemos:

BASCOM AVR, nos permite interactuar con el hardware, mediante cuadros de diálogo, a los cuales podemos ingresar, mediante el menú de opciones, el mismo que nos aprueba la configuración de los pines que ocuparemos para realizar la comunicación de los distintos dispositivos o periféricos de un microcontrolador AVR.

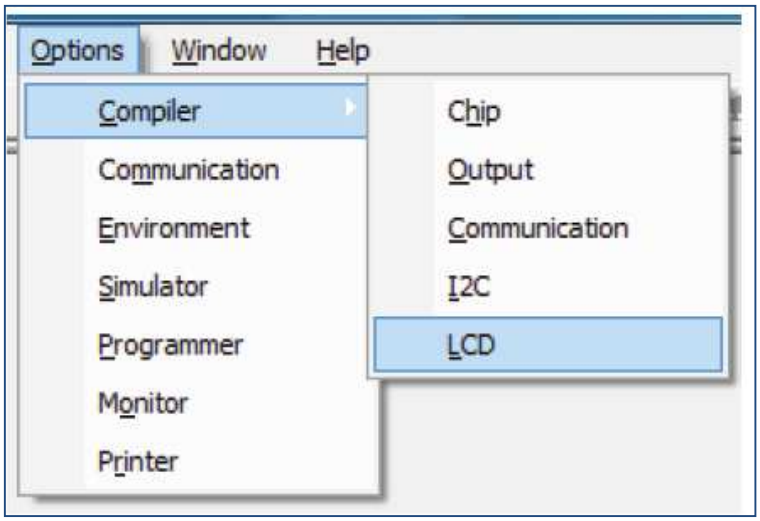


Figura 1. 44 Cuadro de diálogo del BASCOM AVR para configurar LCD

La siguiente pantalla nos muestra cómo podemos configurar los pines y el tipo de LCD que utilizaremos.

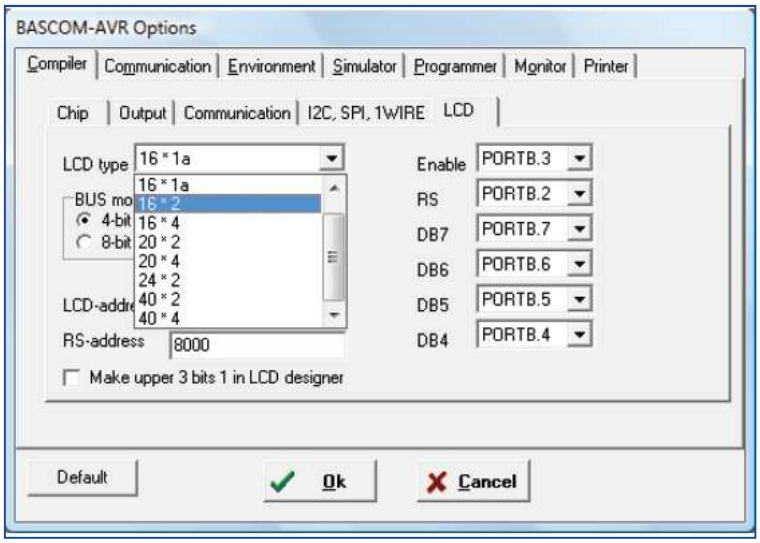


Figura 1. 45 Cuadro de dialogo del BASCOM AVR para escoger el funcionamiento de un LCD

Es recomendable realizar todas las configuraciones de dispositivos mediante código y mediante cuadros de diálogo, de esta manera nos aseguramos que la información de las interfaces de salida esté doblemente escrita y no se pierda en ningún momento.

1.15.1.11 DBRx, PORTx, PINx

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

DBR 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

1.15.1.12 If - Then – Else

Son sentencias condicionales las cuales responden a un estado de voltaje (0L, 1L), de contenido (caracteres), etc.

1.15.1.13 For – Next

Son instrucciones de repetición, ya que se ejecutan un conjunto de instrucciones, dependiendo de una variable incremental que se encuentre dentro del lazo.

1.15.1.14 Select – Case

Son sentencias que se pueden ejecutar, dependiendo del estado de una variable de selección.

1.15.1.15 Símbolos Operadores

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

Además se debe tomar en cuenta que BASCOM nos permite realizar operaciones únicamente con dos variables a la vez.

A continuación podemos observar los más comunes:

1.15.1.15.1 Operadores matemáticos

Suma a = b + c

Resta a = b - c

Multiplicación a = b * c

División a MOD b

1.15.1.15.2 Operadores de Relación

= Igual X=Y

<> No es igual X<>Y

= Igual X=Y

< Menor que X<Y

> Mayor que X>Y

< =Menor o igual que X<=Y

> =Mayor o igual que X>=Y

1.15.1.15.3 Operadores Lógicos

NOT Complemento (Negación)

AND Conjunción (Y)

OR Disyunción (O)

XOR OR Exclusiva

1.15.1.15.4 Representación lógica digital

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

Por ejemplo:

Porta=&HF9 Número Hexadecimal

CAPÍTULO 2 ESTRUCTURA, DISEÑO, FUNCIONAMIENTO Y DIAGRAMAS DEL SISTEMA DE ACCESO

2.1 ESTRUCTURA DEL PROYECTO

Este proyecto está estructurado básicamente de 3 partes; programación del AVR, el circuito electrónico y la comunicación entre el AVR y PC.

Bascom AVR, es el lenguaje de programación de alto nivel seleccionado para la realización de este proyecto, La carga de programas se puede realizar desde un PC a través del bus SPI con el AVR-ISP in system programing de Atmel por el puerto serie, para verificar la correcta elaboración del programa se emulo en el programa Proteus 7.6. Para comenzar tenemos que mencionar que para programar en alto nivel como es el caso este circuito, al realizar la programación del AVR en Bascom, que estructuramos el programa en 4 partes.

En primer lugar se realiza las configuraciones y dimensionamiento de variables y subrutinas, esto quiere decir que tendremos que especificar varios aspectos tales como por ejemplo: microcontrolador a utilizarse, frecuencia del cristal, asignar configuraciones como entrada o salida de información a todo un puerto o a un pin, declaración de tiempos de espera a subrutinas, etc.

En segundo lugar elaboramos el programa principal, esto quiere decir que tendremos que definir cuáles son las instrucciones principales y cuál será la frecuencia con la que éstas se repiten.

Un tercer lugar ocupa las subrutinas, estas son pequeños programas, a los cuales se les llama cuando el programa principal así lo requiera.

El cuarto lugar en la estructura de nuestro programa va a ser la elaboración de tablas de datos, estas van a ser llamadas por el programa principal de ser requeridas.

El circuito electrónico tendrá como unidad de control al AVR, el mismo será el encargado de permitir o no el acceso, también controlará un circuito de alarma el

mismo que consiste en sensores magnéticos normalmente cerrados y una sirena. Tiene como fuente de alimentación un adaptador de 12 voltios y dado el caso una batería de reserva de 12 voltios para ello cuenta con pequeño circuito de fuente de emergencia. También cuenta con un circuito para el reloj en tiempo real, para aquello utiliza el Dallas DS1307.

Este circuito, permite interactuar con el LCD, teclado hexadecimal, para ello se pone resistencias como protección, también nos asegura que el voltaje que llega al AVR y al LCD sea de 5 voltios para ello se hace el uso del regulador 7805, nos permite también la grabación directa del programa en el AVR ya que incluye un circuito de ICSP.

El circuito posee un circuito de reset, en caso de que se presenten fallas, un interruptor para el encendido, con unas luces indicadoras, para el encendido, activación de alarma y acceso.

Para la comunicación entre el AVR y el PC vamos a realizar un programa en el Visual Basic este se crea a partir de unidades de creación estándar. El programa del proyecto contiene varios formularios y un módulo para su ensamblado, todos estos contienen todo el código para que el programa final pueda ser ejecutado.

El funcionamiento del proyecto va a ser independiente del programa en Visual Basic, la comunicación para la obtención de información que se realice será solo cuando el administrador del proyecto lo requiera.

Este programa va a ser el encargado de enviar usuarios y recibir la información de la hora en que éstos accedan con el ingreso de su respectiva clave.

2.2 DISEÑO DEL PROYECTO

Este sistema figura 2. 1 está diseñado para controlar el acceso a los usuarios de una oficina, u organización y puedan interactuar con este, mediante un teclado matricial pueden ingresar la contraseña de los usuarios registrados en el sistema, para que puedan visualizar si el ingreso de ellas es correcto.

Este sistema también permite al administrador recibir información para aquello, se incorpora un conector DB9 para que pueda conectarse con este a un computador a través de un cable.

Se construyo un prototipo en madera emulando una entrada, la cual consta de una puerta y dos ventanas en las cuales se les incorpora sensores magnéticos normalmente cerrados, a este prototipo de una entrada se le instaló la cerradura eléctrica y el dispositivo sonoro.

2.1 Dispositivo sonoro

Los sensores magnéticos controlan el acceso de personas no autorizadas por intrusión. El teclado es llamado unidad de operación y sirve para comunicarse con el sistema. Los dispositivos de salida son la sirena que es un dispositivo sonoro para la notificación local de alarma y el computador para adquisición de información.

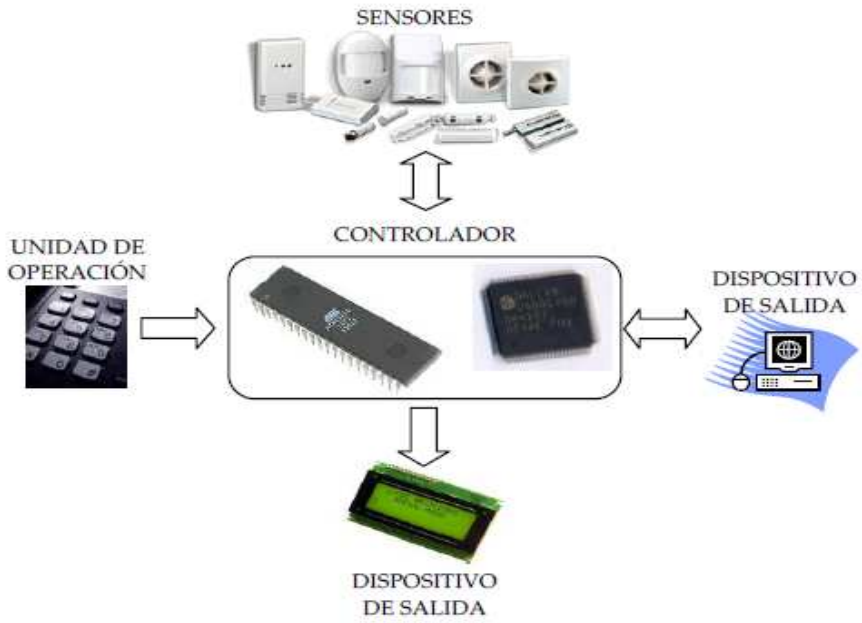


Figura 2. 2 Organización del sistema

El proyecto está diseñado para el control en el ingreso de personal a una oficina u organización, para lo cual cada una de las personas va a tener una clave de cuatro dígitos para el ingreso, también está diseñado para presentar la hora a la cual los usuarios acceden al lugar.

2.3 FUNCIONAMIENTO DEL SISTEMA

Este sistema está diseñado para controlar el acceso del personal a una empresa u organización de 20 personas, cada usuario tiene una clave o contraseña y nombre del usuario, que estarán registrados en una base de datos que se alojara en el programa, los nombres de usuarios como las claves podrán ser modificadas desde el programa creado en Visual Basic.

En la pantalla LCD se muestra “INGRESE CLAVE”, se ingresa clave mediante el teclado, la misma que tendrá “4” dígitos, si se ingresa clave correcta se puede acceder al lugar, ya que este desactivara el dispositivo sonoro y activara la cerradura electrónica la puerta se abrirá.

Si al contrario se ingresa tres veces claves erróneas, actuará un dispositivo sonoro acoplado al circuito.

Si tratan de ingresar personas no autorizadas al lugar por intrusión, protegidas con este dispositivo de igual manera este activará un dispositivo sonoro.

El ingreso de claves se registra en las memorias internas del aparato electrónico, horas de entradas de todo el personal que puede ser hasta 20 personas en este caso.

Solo el administrador puede tener acceso al sistema, y será capaz de manipular los datos tales como usuarios y claves, revisar la información, para ello se elabora el programa en Basic que solo el usuario administrador puede manejarlo.

A continuación funcionamiento de los programas a implementar y las herramientas necesarias para esto.

2.3.1 PROTEUS VERSION 7.6

El proteus es una herramienta importante porque podemos simular el funcionamiento del circuito.

La instalación de este software es práctica y sencilla solo se selecciona la carpeta que contiene el software y damos clic en setup y luego activamos el programa.

2.3.2 ARMADO DEL CIRCUITO EN EL PROTEUS VERSION 7.6

Elaboración del circuito de control de acceso con todos los elementos antes mencionados en el Proteus.

Es necesario mencionar que la sirena y la cerradura eléctrica serán reemplazadas en la simulación por leds que se encenderá si esta se activa al igual que la cerradura eléctrica que se activa con un pulso.

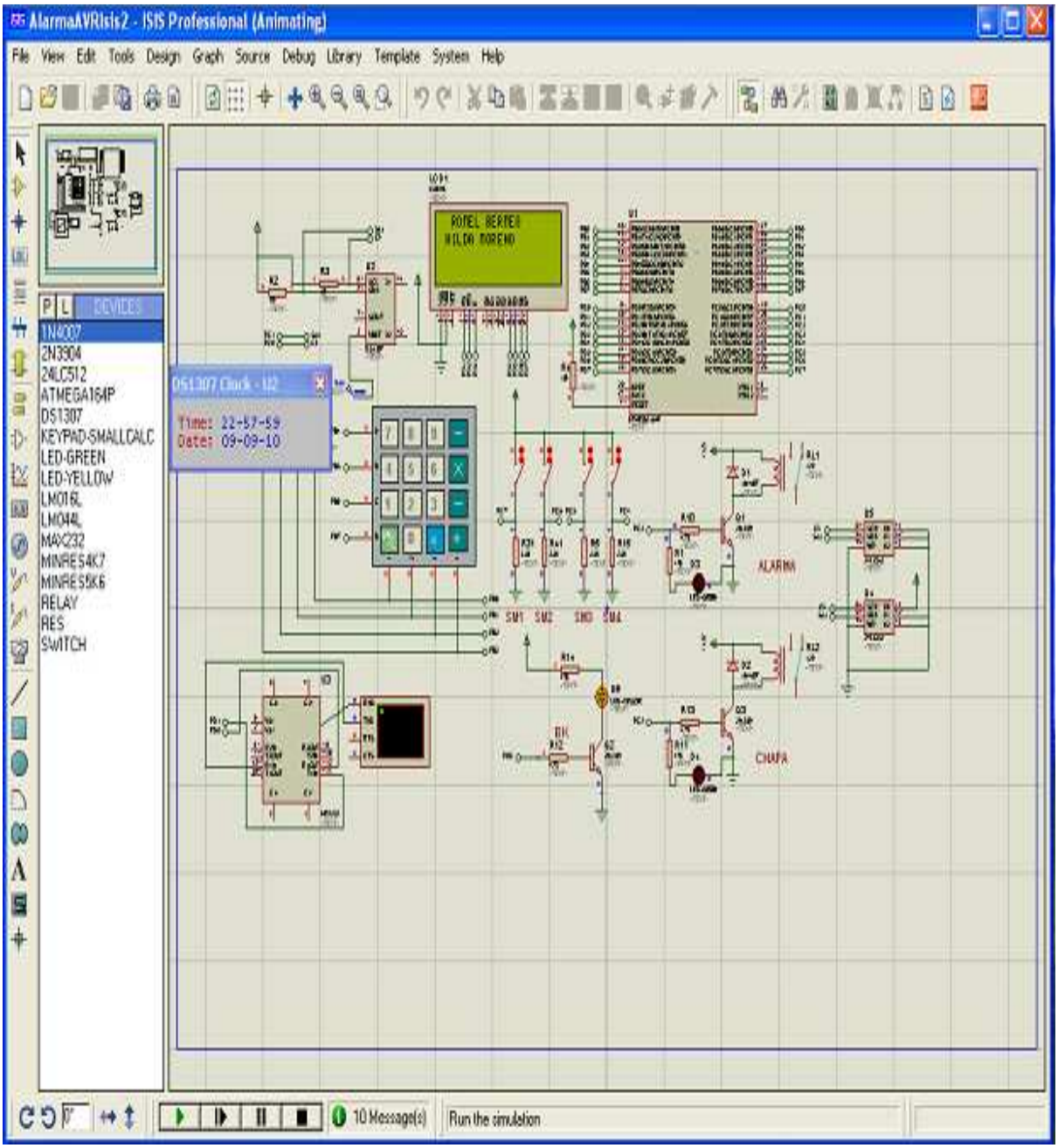


Figura 2. 3 Circuito armado en el Proteus versión 7.6

Para la simulación se necesita incluir el programa en el AVR Atmega 164p, para ello fue necesario la instalación del Bascom AVR y la programación del AVR.

2.3.3 BASCOM AVR

El Bascom AVR es un programa de fácil instalación, solo es necesario hacer clic en el archivo de instalación, este programa no necesita licencias adicionales para su funcionamiento, su utilización se la menciono en la información teórica.

2.3.4 ELABORACIÓN DEL PROGRAMA PARA EL CONTROL DE ACCESO EN BASCOM AVR

Primero abrimos el Bascom AVR, nos indica la siguiente ventana figura 2.4, guardamos el proyecto, procedemos a realizar la programación.

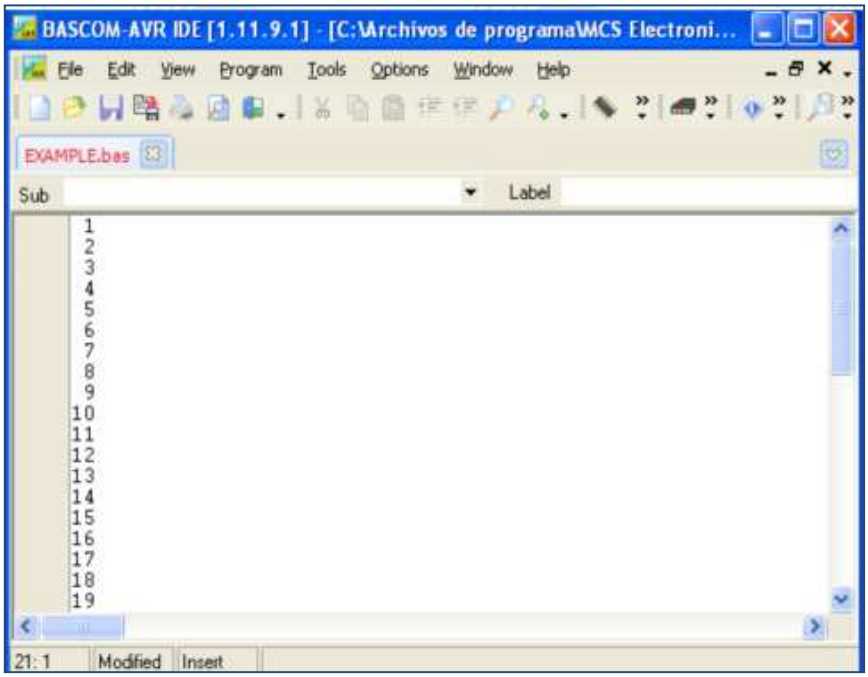


Figura 2. 4 Pantalla para la Programación en Bascom AVR

PROGRAMA:

```
'******************CONFIGURACIONES Y DIMENSIONAMIENTOS******************
$regfile = "m164Pdef.dat "                    'ATMEGA 164P
$crystal = 8000000                            'frecuencia del cristal 8 Mhz
$initmicro
```

```
'$lib "mcsbyte.lbx"

Ddra 0.7 = 1                              'Portico A.7 como salida

Porta 0.7 = 1                             'BK=1

Config Debounce = 150

'**************DECLARACIÓN DE FUNCIONES Y SUBRUTINAS ***************

Declare Sub Writebyte 512a(address As Word , Value As Byte)

Declare Function Readbyte 512a(address As Word) As Byte

Declare Sub Writebyte 512b(address As Word , Value As Byte)

Declare Function Readbyte 512b(address As Word) As Byte

Dim B As Byte, L As Byte, H As Byte

'****************** CONFIGURACIÓN LCD ALFANUMÉRICO ********************

Config Lcdpin = Pin, Rs = Porta.0 , E = Porta.2 , Db4 = Porta.3 , Db5 = Porta.4 ,
Db6 = Porta.5 , Db7 = Porta.6

Config Lcd = 20 * 4

'****************************** INTERRUPCIÓN  DE SERIAL *************************

On Urxc Rec_isr                           'Define subrutina de Interrupción
Serial

Enable Urxc

Enable Interrupts

'********************************** DS1307 RTC ***************************************

Config Sda = Portd.4                      'Configura I2C pin Datos PORTC.5

Config Scl = Portd.3                      'Configura I2C pin Reloj PORTC.4
```

```
'Dirección del Ds1307

Const Ds1307w = &HD0                              'Constante escritura DS1307

Const Ds1307r = &HD1                              'Constante lectura DS1307

'*****************************DIRECCIÓN DE MEMORIAS*********************************

'address of 24LC512(A0=0 A1=0 A2=0)

Const M24lc512aw = &HA0                           'Dirección de Memoria1

Const M24lc512ar = &HA1

'address of 24LC512(A0=1 A1=0 A2=0)

Const M24lc512bw = &HA2                           'Dirección de Memoria2

Const M24lc512br = &HA3

'*********DEFINICIONES INICIALES PARA CONFIGURAR EL TECLADO **********

Ddrb 0# = 0

Portb 0# = 1

Ddrb 0.1 = 0

Portb 0.1 = 1

Ddrb 0.2 = 0

Portb 0.2 = 1

Ddrb 0.3 = 0

Portb 0.3 = 1

Ddrb 0.4 = 1

Portb 0.4 = 1

Ddrb 0.5 = 1
```

```
Portb 0.5 = 1

Ddrb 0.6 = 1

Portb 0.6 = 1

Ddrb 0.7 = 1

Portb 0.7 = 1

F1 Alias Portb.4                    'nombres para los pines de las filas

F2 Alias Portb.5

F3 Alias Portb.6

F4 Alias Portb.7

C1 Alias Pinb.0                     'nombres para los pines de las columnas

C2 Alias Pinb.1

C3 Alias Pinb.2

C4 Alias Pinb.3

'********************************** ALARMA (BUZZER) **********************************

Ddrc 0.6 = 1

Portc 0.6 = 0

Alarma Alias Portc.6

'**********************************CERRADURA**************************************

Ddrc 0.7 = 1

Portc 0.7 = 0

Chapa Alias Portc.7
```

```
'************************************SENSOR1*****************************************

Ddrc 0.5 = 0

Portc 0.5 = 1

Sensor1 Alias Pinc.5

'************************************SENSOR2*****************************************

Ddrc 0.4 = 0

Portc 0.4 = 1

Sensor2 Alias Pinc.4

'************************************SENSOR3*****************************************

Ddrc 0.3 = 0

Portc 0.3 = 1

Sensor3 Alias Pinc.3

'************************************SENSOR4 *****************************************

Ddrc 0.2 = 0

Portc 0.2 = 1

Sensor4 Alias Pinc.2

'**********DECLARA VARIABLES NECESARIAS PARA DS1307**********

Dim Dias As Byte

Dim Segu As Byte

Dim Minu As Byte

Dim Hora As Byte

Dim Diam As Byte
```

```
Dim Mes As Byte
Dim Anio As Byte
Dim Dias1 As String * 3
Dim Texto1 As String * 3, Texto2 As String * 2, Texto3 As String * 2
Dim Texto4 As String * 30, Texto5 As String * 30
Dim Serial1 As String * 10
Dim Flag1 As Bit                    'variable tipo bit
Dim Flag2 As Bit                    'variable tipo bit
Dim Flag3 As Bit                    'variable tipo bit
Dim Flag4 As Bit                    'variable tipo bit
Dim Flag10 As Bit                   'variable tipo bit

Dim Pos As Byte                     'variable para Posición del cursor
Dim Tempod As Byte                  'variable para Posición del cursor
Dim Tempou As Byte                  'variable para Posición del cursor
Dim Tempo As Byte                   'variable para Posición del cursor
Dim Tecla As Byte                   'variable para Posición del cursor

Dim Cuenta As Byte                  'variable para Posición del cursor
Dim Cuenta1 As Byte                 'variable para Posición del cursor
Dim Clave1(5) As Byte               'variable para Posición del cursor
Dim X As Word
```

```
Dim Y As Word

Dim Z As Word

Dim Registro As Word

Dim Registro1 As Word

Dim Flag11 As Bit

Dim Usuarios As Word

Dim Usuariosb As Word

Dim Dir1 As Word

Dim Dir2 As Word

Dim Z1 As Byte

Dim Temporal As String * 1

Dim Contador1 As Word

Dim Contador2 As Word

Dim Clave As String * 6

Dim Nombreusuario As String * 25

Dim Nombreusuario1 As String * 25

Dim Numeroc As Word

Dim Claveu(100) As Word

Dim Clavenum(6) As Word

Dim Bandera(100) As Byte


Dim Senin1 As Bit
```

```
Dim Senin2 As Bit
Dim Senin3 As Bit
Dim Senin4 As Bit
Const Delay1 = 80
Const Retardo = 3
'Dias = 5
'dia de la semana (D=1,L=2,M=3,M=4,J=5,V=6,S=7)
'Diam = 02
'Mes = 9
'Anio = 10
'Gosub Setdate
'Segu = 0
'Minu = 01
'Hora = 19
'Gosub Settime
Deflcdchar 1, 2, 4, 14, 1, 15, 17, 15, 32          ' replace ? with number (0-7)
Deflcdchar 2, 2, 4, 14, 17, 31, 16, 14, 32         ' replace ? with number (0-7)
Deflcdchar 3, 2, 4, 12, 4, 4, 4, 14, 32            ' replace ? with number (0-7)
Deflcdchar 4, 2, 4, 14, 17, 17, 17, 14, 32         ' replace ? with number (0-7)
Deflcdchar 5, 2, 4, 17, 17, 17, 19, 13, 32         ' replace ? with number (0-7)
Deflcdchar 6, 14, 32, 22, 25, 17, 17, 17, 32       ' replace ? with number (0-7)
Deflcdchar 7, 14, 32, 17, 25, 21, 19, 17, 32       ' replace ? with number (0-7)
```

```
Cursor Off

Cls

Flag2 = 0

Dir2 = 0

Readeeprom X, 2

Usuariosb = X

Usuarios = X - 1

Readeeprom X, 6

Registro = X

For Y = 1 To Usuariosb

Bandera(Y) = 0

Next Y

Cls

GoSub Presentacion

For Y = 0 To Usuarios

Numeroc = Y

GoSub Traerclaves

Claveu(Numeroc + 1) = Val(Clave)

'   Dato = Str(claveu(numeroc + 1))

'   Dato = Format(dato , "0000")

'   Lcd Dato

Next Y
```

```
'Wait 5
Locate 1, 1
Lcd "Para comenzar elija "
 Locate 2, 1
Lcd "---> Tecla: " ; Chr(34) ; "A" ; Chr(34) ; " <---"
Do
 Alarma = 0
 Tecla = 255
GoSub Barrido
GoSub ReboteSelect Case Tecla
Case 10
GoSub Clavemaster
End Select
GoSub Getdatetime
Locate 3, 7
Lcd Texto4
Locate 4, 5
Lcd Texto5
'******************************** INGRESO DE USUARIOS ****************************
If Flag1 = 1 Then
Flag1 = 0
GoSub Grabarmem
```

```
 Enable Interrupts
Enable Urxc
Readeeprom X, 2
Usuariosb = X
Usuarios = X - 1
For Y = 0 To Usuarios
Numeroc = Y
GoSub Traerclaves
Claveu(Numeroc + 1) = Val(Clave)
 Next Y
End If
'****************************** DESCARGAR DATOS *********************************
If Flag2 = 1 Then
Flag2 = 0
Cls
Home Upper                              'Limpia la pantalla
Lcd "   Descargando  "
 Lowerline
Lcd "   Datos en PC "
If Registro = 0 Then
Cls
Home Upper                              'Limpia la pantalla
```

```
Lcd "NO EXISTE  DATOS"
Lowerline
Lcd " PARA DESCARGAR”
Serial1 = Str(Registro)
Serial1 = Format(Serial1, "00000")
Print Serial1
Wait Retardo
Locate 1, 1
Lcd "Para comenzar elija "
Locate 2, 1
Lcd "---> Tecla: " ; Chr(34) ; "A" ; Chr(34) ; " <---"
Else
Serial1 = Str(Registro)
Serial1 = Format(Serial1, "00000")
 Print Serial1
Waitms 10
Dir2 = 0
Registro1 = Registro – 1
For X = 0 To Registro1
Input Serial1 Noecho
Nombreusuario1 = ""
Dir2 = X * 40
```

```
For Y = 0 To 39
B = Readbyte 512b(Dir2)
Temporal = Chr(B)
Nombreusuario1 = Nombreusuario1 + Temporal
Dir2 = Dir2 + 1
Next
Print Nombreusuario1;
Waitms 10
Input Serial1 Noecho
Print "FF";
Waitms 10
Next X
Cls
Home Upper                                          'Limpia la pantalla
Lcd "    Descarga    "
Lowerline
Lcd "  Completa  OK  "
Wait Retardo
Locate 1, 1
Lcd "Para comenzar elija "
Locate 2, 1
Lcd "---> Tecla: " ; Chr(34) ; "A" ; Chr(34) ; " <---"
```

```
For Y = 0 To Usuarios
Numeroc = Y
GoSub Traerclaves
Claveu(Numeroc + 1) = Val(Clave)
Next Y
End If
Enable Interrupts
Enable Urxc
End If
*****************************************************************************************
If Flag4 = 1 Then                                   'Resetear variable CONTAR
Flag4 = 0
Cls                                                  'Limpia la pantalla
Lcd "   Reset Contador   "
Registro = 0
Writeeeprom Registro, 6
Readeeprom Registro, 6
Enable Interrupts
Enable Urxc
Wait Retardo
Locate 1, 1
Lcd "Para comenzar elija "
```

```
Locate 2, 1
Lcd "---> Tecla: " ; Chr(34) ; "A" ; Chr(34) ; " <---"
End If
'*************************************************************************************************
Loop
'********************SUBRUTINA PARA LEER LOS SENSORES **********************
Sensores:
Senin1 = Sensor1
 If Senin1 = 1 Then                          'Revisar Sensor Magnético
 Alarma = 1
Else
'      Alarma = 0
End If
Senin2 = Sensor2
If Senin2 = 1 Then                          'Revisar Sensor Magnético
Alarma = 1
Else
'      Alarma = 0
End If
Senin3 = Sensor3
If Senin3 = 1 Then                          'Revisar Sensor Magnético
Alarma = 1
```

```
Else
'      Alarma = 0
End If
Senin4 = Sensor4
If Senin4 = 1 Then                              'Revisar Sensor Magnético
Alarma = 1
Else
'      Alarma = 0
End If
If Sensor1 = 0 And Sensor2 = 0 And Sensor3 = 0 And Sensor4 = 0 Then
Alarma = 0
End If
Return
'*****************************SUBRUTINA LEER TECLAS ***************************
Clavemaster:
Flag10 = 1
Cuenta = 1
Cuenta1 = 0
While Flag10 = 1
GoSub Sensores
GoSub Getdatetime
Locate 3, 7
```

```
Lcd Texto4
Locate 4, 5
Lcd Texto5
Tecla = 255
If Cuenta = 1 Then
Home Upper
Lcd " INGRESE CLAVE      "
Lowerline
Lcd "USUARIO :          "
Locate 2, 10
End If
GoSub Barrido
GoSub Rebote
Select Case Tecla
If Cuenta < 5 And Tecla < 10 Then
Clave1(Cuenta) = Tecla
Select Case Cuenta
Case 1
Locate 2, 10
Case 2
Locate 2, 11
Case 3
```

```
Locate 2, 12

Case 4

Locate 2, 13

End Select

Lcd "*"

Cuenta = Cuenta + 1

End If

Case 10

Locate 1, 1

Lcd "Para comenzar elija "

Locate 2, 1

Lcd "---> Tecla: " ; Chr(34) ; "A" ; Chr(34) ; " <---"

Flag10 = 0

Case 11

GoSub Cambiarreloj

Case 15

If Cuenta = 5 Then

Clavenum(1) = Clave1(1) * 1000

Clavenum(2) = Clave1(2) * 100

Clavenum(3) = Clave1(3) * 10

Clavenum(4) = Clavenum(1) + Clavenum(2)

Clavenum(5) = Clavenum(3) + Clave1(4)
```

```
Clavenum(6) = Clavenum(4) + Clavenum(5)

Flag11 = 0

For X = 1 To Usuariosb

If Claveu(X) = Clavenum(6) Then

Flag11 = 1

Exit For

End If

Next X

If Flag11 = 1 Then

Numeroc = X – 1

GoSub Traerusuarios

Home Upper

Lcd "   BIENVENIDO (A)   "       'Ir a la Primera línea

Lowerline

Lcd Nombreusuario                  'Ir a la Segunda línea

Alarma = 0

Chapa = 1

GoSub Grabarmem1

Registro = Registro + 1

Writeeeprom Registro, 6

Waitms 500

Chapa = 0
```

```
Cuenta1 = 0
Wait 2
Else
Cls
Lcd "CLAVE INCORRECTA”
Wait Retardo
End If
Else
Cls
Lcd “CLAVE INCORRECTA"
Wait Retardo
End If
Cuenta = 1
Cuenta1 = Cuenta1 + 1
If Cuenta1 > 3 Then
Cls
Lcd " Alarma  Activada ”
Cuenta = 1
Cuenta1 = 0
Alarma = 1
'Flag10 =
Wait Retardo
```

```
End If
Case 14
Cuenta = 1                                  'Borra dato mal ingresado
End Select
Wend
Waitms 1
Return
'*****************************************************************************************
Getdatetime:
I2cstart                        'Genera inicio de I2C
I2cwbyte Ds1307w                'Envia Constante escritura del DS1307
I2cwbyte 0                      'Direccion Inicial de memoria del DS1307
I2cstart                        'Genera inicio de I2C
I2cwbyte Ds1307r                'Envia Constante lectura del DS1307
I2crbyte Segu, Ack              'Lee los segundos
I2crbyte Minu, Ack              'Lee los minutos
I2crbyte Hora, Ack              'Lee la hora
I2crbyte Dias, Ack      'Lee el día de la semana (D=1,L=2,M=3,M=4,J=5,V=6,S=7)
I2crbyte Diam, Ack              'Lee día del mes
I2crbyte Mes, Ack               'Lee el mes del año
I2crbyte Anio, Nack             'Lee el año
I2cstop
```

```
'  Segu = Makedec(segu)
'  Minu = Makedec(minu)
'  Hora = Makedec(hora)
'  Dias = Makedec(dias)
'  Diam = Makedec(diam)
'  Mes = Makedec(mes)
'  Anio = Makedec(anio)
Select Case Dias
Case 1
Dias1 = "Dom"                    'Mostrar Día De La Semana
Case 2
Dias1 = "Lun"                    'Mostrar Día De La Semana
Case 3
Dias1 = "Mar"                    'Mostrar Día De La Semana
Case 4
Dias1 = "Mie"                    'Mostrar Día De La Semana
Case 5
Dias1 = "Jue"                    'Mostrar Día De La Semana
Case 6
Dias1 = "Vie"                    'Mostrar Día De La Semana
Case 7
Dias1 = "Sab"                    'Mostrar Día De La Semana
```

```
End Select

Texto1 = Hex(Hora)

Texto2 = Hex(Minu)

Texto3 = Hex(Segu)

Texto4 = Format(Texto1, "00") + ":" + Format(Texto2, "00") + ":" + Format(Texto3, "00")

Texto1 = Hex(Anio)

Texto2 = Hex(Diam)

Texto3 = Hex(Mes)

Texto5 = Dias1 + "-" + Format(Texto2, "00") + "/" + Format(Texto3, "00") + "/" + Format(Texto1, "00")          'Dias1 + "-" +

Return

'*****************************************************************************************

Setdate:

'  Dias = Makebcd(dias)

'  Diam = Makebcd(diam)

'  Mes = Makebcd(mes)

'  Anio = Makebcd(anio)

I2cstart                          'Genera inicio de I2C

I2cwbyte Ds1307w                  'Envía Constante escritura del DS1307

I2cwbyte 3                        'Dirección del día del mes del DS1307

I2cwbyte Dias                     'Nuevo día de la semana

I2cwbyte Diam                     'Nuevo día del mes
```

```
I2cwbyte Mes                    'Nuevo mes del año
I2cwbyte Anio                   'Nuevo Año
I2cstop
Return
'*****************************************************************************************
Settime:
'  Segu = Makebcd(segu)
'  Minu = Makebcd(minu)
'  Hora = Makebcd(hora)
I2cstart                        'Genera inicio de I2C
I2cwbyte Ds1307w                'Envía Constante escritura del DS1307
I2cwbyte 0                      'Dirección de segundos del DS1307
I2cwbyte Segu                   'Nuevo Segundo
I2cwbyte Minu                   'Nuevo Minuto
I2cwbyte Hora                   'Nueva Hora
I2cstop
Return

'*****************************************************************************************

'*******************SUBRUTINA DE BARRIDO DE TECLAS ************************
Barrido:
```

```
F1 = 0
If C1 = 0 Then: Tecla = 1: Return: End If
If C2 = 0 Then: Tecla = 2: Return: End If
If C3 = 0 Then: Tecla = 3: Return: End If
If C4 = 0 Then: Tecla = 10: Return: End If
F1 = 1: F2 = 0
If C1 = 0 Then: Tecla = 4: Return: End If
If C2 = 0 Then: Tecla = 5: Return: End If
If C3 = 0 Then: Tecla = 6: Return: End If
If C4 = 0 Then: Tecla = 11: Return: End If
F2 = 1: F3 = 0
If C1 = 0 Then: Tecla = 7: Return: End If
If C2 = 0 Then: Tecla = 8: Return: End If
If C3 = 0 Then: Tecla = 9: Return: End If
If C4 = 0 Then: Tecla = 12: Return: End If
F3 = 1: F4 = 0
If C1 = 0 Then: Tecla = 14: Return: End If
If C2 = 0 Then: Tecla = 0: Return: End If
If C3 = 0 Then: Tecla = 15: Return: End If
If C4 = 0 Then: Tecla = 13: Return: End If
F4 = 1
Waitms 1
```

```
Return

'**********************SUBRUTINA DE REBOTE DE TECLAS ************************

Rebote:

While C1 = 0: Wend

While C2 = 0: Wend

While C3 = 0: Wend

While C4 = 0: Wend

Waitms 1

Return

'************************************PRESENTACIÓN*************************************

Presentacion:

Cls                                 'Limpia la pantalla

Lcd "ESC. POLITECNICA"

Lowerline

 Lcd "     NACIONAL   "

Wait Retardo                        'Esperar 3 seg.

Cls                                 'Limpia la pantalla

Lcd "   E S F O T "

Lowerline                           'Ir a la Segunda línea

Lcd "

Wait Retardo                        'Esperar 3 seg.

Cls                                 'Limpia la pantalla
```

```
Lcd " ELECTRONICA  Y "
Lowerline                          'Ir a la Segunda línea
Lcd "TELECOMUNICACIONES"
Wait Retardo                       'Esperar 3 seg.
Cls                                'Limpia la pantalla
Lcd " ACCESO DE PERSONAL"
Lowerline                          'Ir a la Segunda línea
Lcd "REALIZADO   POR:"
Wait Retardo                       'Esperar 3 seg.
Cls                                'Limpia la pantalla
Lcd " ROMMEL BERMEO  "
Lowerline                          'Ir a la Segunda línea
Lcd " HILDA MORENO   "
Wait Retardo                       'Esperar 3 seg.
Cls                                'Limpia la pantalla
Return
'*************************GRABACIÓN EN MEMORIA*********************************
Grabarmem:
Dir1 = 0
Readeeprom X, 2
For Y = 1 To X
For Z = 1 To 24
```

```
Input Serial1 Noecho

Z1 = Asc(Serial1)

Call Writebyte512a(Dir1, Z1)

Dir1 = Dir1 + 1

Next Z

Next Y

Return

'*************************GRABACIÓN EN MEMORIA*********************************

Grabarmem1:

Dir2 = Registro * 40

For Z = 1 To 20

Temporal = Mid(Nombreusuario1, Z, 1)

Z1 = Asc(Temporal)

Call Writebyte512b(Dir2, Z1)

Dir2 = Dir2 + 1

Next Z

For Z = 1 To 8

Temporal = Mid(Texto4, Z, 1)

Z1 = Asc(Temporal)

Call Writebyte512b(Dir2, Z1)

Dir2 = Dir2 + 1

Next Z
```

```
For Z = 1 To 12

Temporal = Mid(Texto5, Z, 1)

Z1 = Asc(Temporal)

Call Writebyte512b(Dir2, Z1)

Dir2 = Dir2 + 1

Next Z

Return

'*****************************************************************************************

'write a byte to the 24LC512 memory

Sub Writebyte512a(address As Word, Value As Byte)

H = High(address)

L = Low(address)

I2cstart

I2cwbyte M24lc512aw

I2cwbyte H                    'MSB

I2cwbyte L                    'LSB

I2cwbyte Value                'value

I2cstop

Waitms 10

End Sub


'read a byte from the 24LC512 memory
```

```
Function Readbyte512a(address As Word) As Byte
H = High(address)
L = Low(address)
I2cstart
I2cwbyte M24lc512aw
I2cwbyte H
I2cwbyte L
I2cstart                                'repeated start
I2cwbyte M24lc512ar
I2crbyte Readbyte512a, Nack         'read byte
I2cstop
End Function
'*****************************************************************************************
'write a byte to the 24LC512 memory
Sub Writebyte512b(address As Word, Value As Byte)
H = High(address)
L = Low(address)
I2cstart
I2cwbyte M24lc512bw
I2cwbyte H                              'MSB
I2cwbyte L                             'LSB
I2cwbyte Value                          'value
```

```
I2cstop
Waitms 10
End Sub
'read a byte from the 24LC512 memory
Function Readbyte512b(address As Word) As Byte
H = High(address)
L = Low(address)
I2cstart
I2cwbyte M24lc512bw
I2cwbyte H
I2cwbyte L
I2cstart                                'inicio repetido
 '   Hdr.0 = 1                          'ahora se leerá
I2cwbyte M24lc512br
I2crbyte Readbyte512b, Nack        'leer byte
I2cstop
End Function
'*********************************LEER CLAVES*****************************************
Traerclaves:
Clave = ""
Temporal = ""
Contador1 = Numeroc * 24
```

```
Contador2 = Contador1 + 4

For X = Contador1 To Contador2

B = Readbyte512a(X)

Select Case B

Case 225

B = 1

Case 233

B = 2

Case 237

B = 3

Case 243

B = 4

Case 250

B = 5

Case 241

B = 6

Case 209

B = 7

End Select

Temporal = Chr(B)

Clave = Clave + Temporal

Next X
```

```
Return

'***********************************LEER USUARIOS************************************

Traerusuarios:

Nombreusuario = ""

Nombreusuario1 = ""

Temporal = ""

Contador1 = Numeroc * 24

Contador1 = Contador1 + 4

Contador2 = Contador1 + 19

For X = Contador1 To Contador2

B = Readbyte512a(X)

Temporal = Chr(B)

Nombreusuario1 = Nombreusuario1 + Temporal

Select Case B

Case 225

B = 1

Case 233

B = 2

Case 237

B = 3

Case 243

B = 4
```

```
Case 250

B = 5

Case 241

B = 6

Case 209

B = 7

End Select

Temporal = Chr(B)

Nombreusuario = Nombreusuario + Temporal

Next X

Return


'************************ SUBRUTINA CAMBIAR EL TIEMPO ************************

Cambiarreloj:

Flag3 = 1

Pos = 0

Locate 3, 8

Cursor On

While Flag3 = 1

Tecla = 255

GoSub Barrido

GoSub Rebote
```

```
Select Case Tecla
Case 1
Select Case Pos
Case 0
Tempo = Hora
GoSub Bcdabin
Tempo = Tempo + 1
If Tempo = 24 Then
Tempo = 0
End If
GoSub Binabcd
Hora = Tempo
Locate 3, 7
Lcd Hex(Hora)
Locate 3, 8
Case 1
Tempo = Minu
GoSub Bcdabin
Tempo = Tempo + 1
If Tempo = 60 Then
Tempo = 0
End If
```

```
GoSub Binabcd
Minu = Tempo
Locate 3, 10
Lcd Hex(Minu)
Locate 3, 11
Case 2
Tempo = Segu
GoSub Bcdabin
Tempo = Tempo + 1
If Tempo = 60 Then
Tempo = 0
End If
GoSub Binabcd
Segu = Tempo
Locate 3, 13
Lcd Hex(Segu)
Locate 3, 14
Case 3
Dias = Dias + 1
If Dias = 8 Then
Dias = 1
End If
```

```
Select Case Dias

Case 1

Locate 4, 5

Lcd "Dom"

Locate 4, 7

Case 2

Locate 4, 5

Lcd "Lun"

Locate 4, 7

Case 3

Locate 4, 5

Lcd "Mar"

Locate 4, 7

Case 4

Locate 4, 5

Lcd "Mie"

Locate 4, 7

Case 5

Locate 4, 5

Lcd "Jue"

Locate 4, 7

Case 6
```

```
Locate 4, 5
Lcd "Vie"
Locate 4, 7
Case 7
Locate 4, 5
Lcd "Sab"
Locate 4, 7
End Select
Case 4
Tempo = Diam
GoSub Bcdabin
Tempo = Tempo + 1
If Tempo = 32 Then
Tempo = 1
End If
GoSub Binabcd
Diam = Tempo
Locate 4, 9
Lcd Hex(Diam)
Locate 4, 10
Case 5
Tempo = Mes
```

```
GoSub Bcdabin
Tempo = Tempo + 1
If Tempo = 13 Then
Tempo = 1
End If
GoSub Binabcd
Mes = Tempo
Locate 4, 12
Lcd Hex(Mes)
Case 6
Tempo = Anio
GoSub Bcdabin
Tempo = Tempo + 1
If Tempo = 100 Then
Tempo = 0
End If
GoSub Binabcd
Anio = Tempo
Locate 4, 15
Lcd Hex (Anio)
Locate 4, 16
End Select
```

```
Case 2
Select Case Pos
Case 0
Tempo = Hora
GoSub Bcdabin
Tempo = Tempo - 1
If Tempo = 255 Then
Tempo = 23
End If
GoSub Binabcd
Hora = Tempo
Locate 3, 7
Lcd Hex(Hora)
Locate 3, 8
Case 1
Tempo = Minu
GoSub Bcdabin
Tempo = Tempo - 1
If Tempo = 255 Then
Tempo = 59
End If
GoSub Binabcd
```

```
Minu = Tempo
Locate 3, 10
Lcd Hex(Minu)
Locate 3, 11
Case 2
Tempo = Segu
GoSub Bcdabin
Tempo = Tempo - 1
If Tempo = 255 Then
Tempo = 59
End If
GoSub Binabcd
Segu = Tempo
Locate 3, 13
Lcd Hex (Segu)
Locate 3, 14
Case 3
Dias = Dias - 1
If Dias = 0 Then
Dias = 7
End If
Select Case Dias
```

```
Case 1
Locate 4, 5
Lcd "Dom"
Locate 4, 7
Case 2
Locate 4, 5
Lcd "Lun"
Locate 4, 7
Case 3
Locate 4, 5
Lcd "Mar"
Locate 4, 7
Case 4
Locate 4, 5
Lcd "Mie"
Locate 4, 7
Case 5
Locate 4, 5
Lcd "Jue"
Locate 4, 7
Case 6
Locate 4, 5
```

```
Lcd "Vie"
Locate 4, 7
Case 7
Locate 4, 5
Lcd "Sab"
Locate 4, 7
End Select
Case 4
Tempo = Diam
GoSub Bcdabin
Tempo = Tempo - 1
If Tempo = 0 Then
Tempo = 31
End If
GoSub Binabcd
Diam = Tempo
Locate 4, 9
Lcd Hex(Diam)
Locate 4, 10
Case 5
Tempo = Mes
GoSub Bcdabin
```

```
Tempo = Tempo - 1
If Tempo = 0 Then
Tempo = 12
End If
GoSub Binabcd
Mes = Tempo
Locate 4, 12
Lcd Hex(Mes)
Locate 4, 13
Case 6
Tempo = Anio
GoSub Bcdabin
Tempo = Tempo - 1
If Tempo = 255 Then
Tempo = 99
End If
GoSub Binabcd
Anio = Tempo
Locate 4, 15
Lcd Hex(Anio)
Locate 4, 16
End Select
```

```
Case 3
Pos = Pos + 1
If Pos = 7 Then
Pos = 0
End If
Select Case Pos
Case 0
Locate 3, 8
Case 1
Locate 3, 11
Case 2
Locate 3, 14
Case 3
Locate 4, 7
Case 4
Locate 4, 10
Case 5
Locate 4, 13
Case 6
Locate 4, 16
End Select
Case 11
```

```
GoSub Setdate

GoSub Settime

Cursor Off

Flag3 = 0

End Select

Wend

Return

'************************** SUBRUTINA BCD A BINARIO ****************************

Bcdabin:

   Tempou = Tempo And $0f

   Shift Tempo, Right, 4

   Tempod = Tempo * 10

   Tempo = Tempod + Tempou

Return

'****************************** SUBRUTINA BINARIO A BCD ************************

Binabcd:

   Tempod = Tempo \ 10

   Shift Tempod, Left, 4

   Tempou = Tempo Mod 10

   Tempo = Tempod + Tempou

Return

'*******************************************************************************
```

```
Rec_isr:
Disable Interrupts
Disable Urxc
Input Serial1 Noecho
Select Case Serial1
Case "A"
Input Serial1 Noecho
X = Val(Serial1)
Writeeeprom X, 2
Readeeprom X, 2
Usuarios = X - 1
Flag1 = 1
Case "1"
Flag2 = 1
Case "2"
Flag4 = 1
End Select
Return
'*****************************************************************************************
_init_micro:
Ddra 0.1 = 1
Porta 0.1 = 0
```

Return

2.3.5 PROGRAMA VISUAL BASIC

Visual Basic es un lenguaje de programación, simplifica la programación utilizando un ambiente de desarrollo completamente gráfico que facilitara la creación de interfaces gráficas y, en cierta medida, también la programación misma, posee un editor de código (programa donde se escribe el código fuente), un depurador que es un programa que corrige errores en el código fuente para que pueda ser bien compilado, un compilador, programa que traduce el código fuente a lenguaje de máquina, y un constructor de interfaz gráfica lo que es una forma de programar en la que no es necesario escribir el código para la parte gráfica del programa, sino que se puede hacer de forma visual.

2.3.6 ELABORACIÓN DEL PROGRAMA EN VISUAL BASIC:

Es de vital importancia mencionar que la comunicación entre el AVR y PC se realiza mediante la interfaz RS 232, debido a que este tendrá que interactuar directamente con el PC, el MAX232 soluciona las conexiones necesarias para poder comunicarse con el PC, para registrar los datos que se están enviando desde el circuito al PC, realizamos un programa en Visual Basic este programa nos hace un registro de todos los acontecimientos que se puedan suscitar en el sistema de control de acceso, también se puede visualizar las horas en que cada usuario entra al establecimiento. Y además modificar las claves y los nombres de los usuarios de requerir algún cambio.

Para la elaboración de este programa se utilizo el Programa Visual Basic 6.0, para lo cual tendremos que familiarizarnos con las barras y ventanas de este programa, las cuales son las mostradas en la figura 2.5.

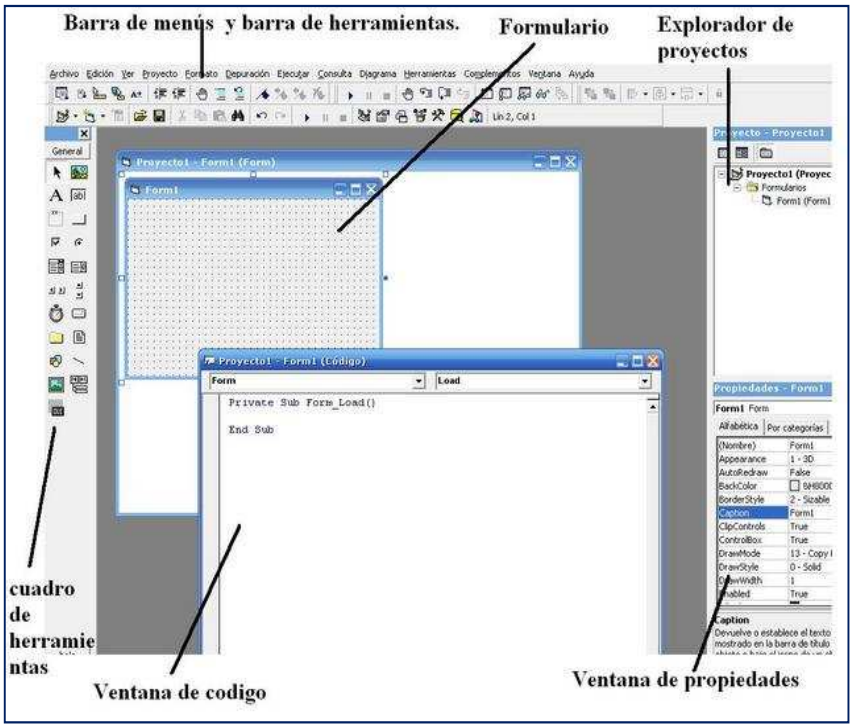


Figura 2. 5 Barras y ventanas de Visual Basic

Una vez familiarizados con este programa primeramente creamos formularios, para ello abrimos el programa y nos aparece la ventana de la figura 2.5 y damos clic en exe standard, luego en abrir.

Figura 2. 6 Ventana de inicio en Visual Basic

Nos aparece la siguiente ventana

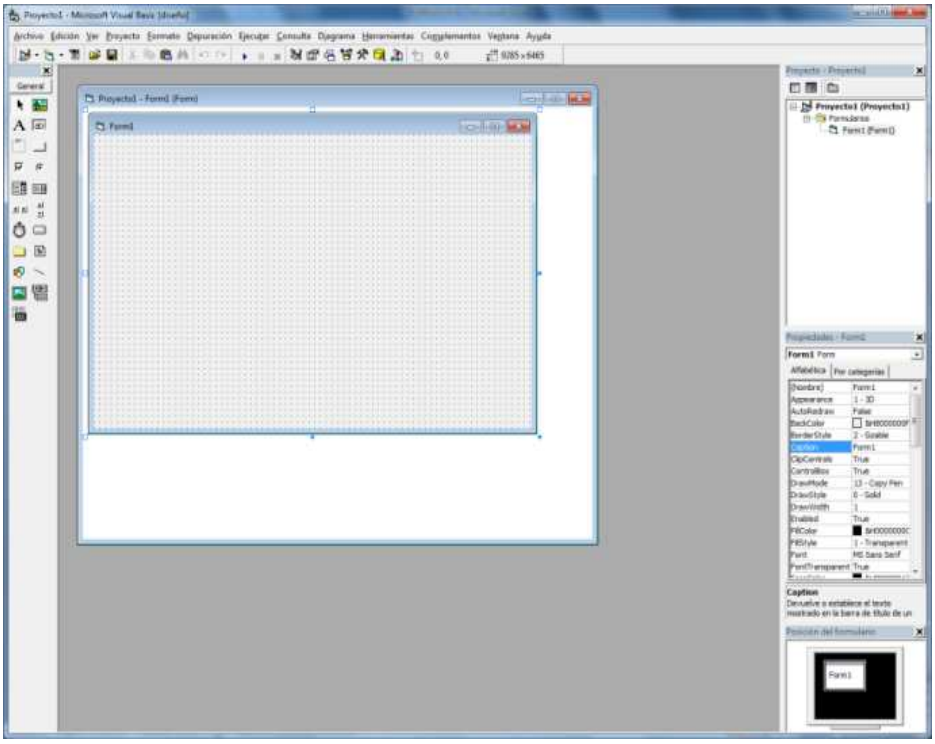


Figura 2. 7 Pantalla al crear un nuevo formulario

Es en esta ventana donde vamos a empezar a crear el nuevo formulario donde vamos a dibujar botones como se muestra en la figura 2.7, que van a estar activos en nuestras ventanas, de otros programas ya realizados tomamos como modelo para la realización de nuestros formularios, en estos formularios nos valemos del cuadro de herramientas y vamos cogiendo botones y dibujando en el formulario. Nos vamos a la ventana de propiedades y la utilizamos para la edición de los tipos de letra, nombres, colores, etc. de nuestros botones.

El primer formulario después de dibujarle es el siguiente, figura 2.8:

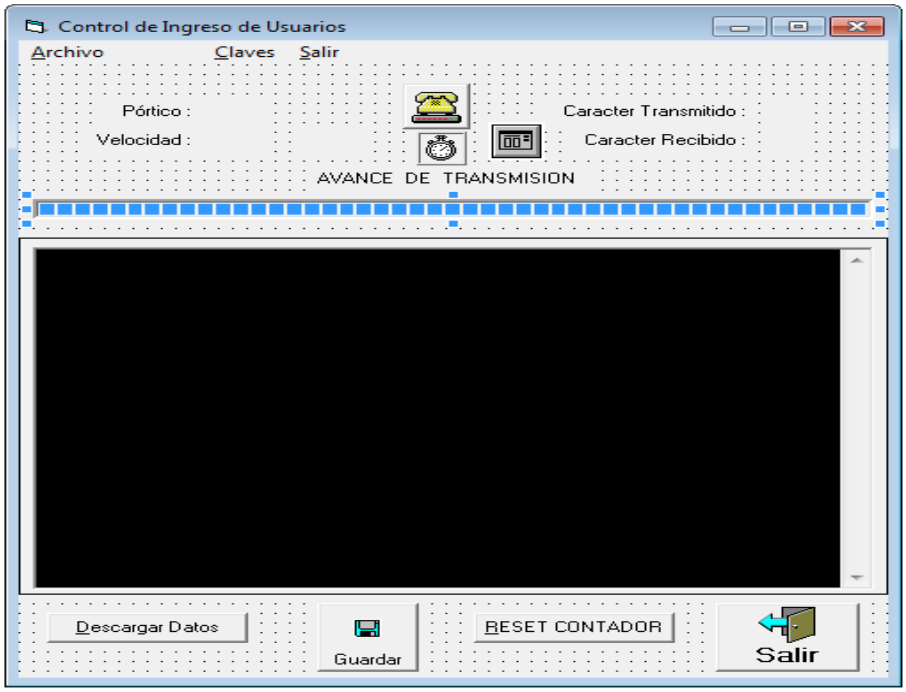


Figura 2. 8 Formulario 1

Una vez dibujado el formulario, damos doble clic en cualquiera de nuestros botones creados y se nos abre una ventana de código. La misma que por default nos presenta:

Private Sub Command1_Click ()

End Sub: como se muestra en la figura 2.9.

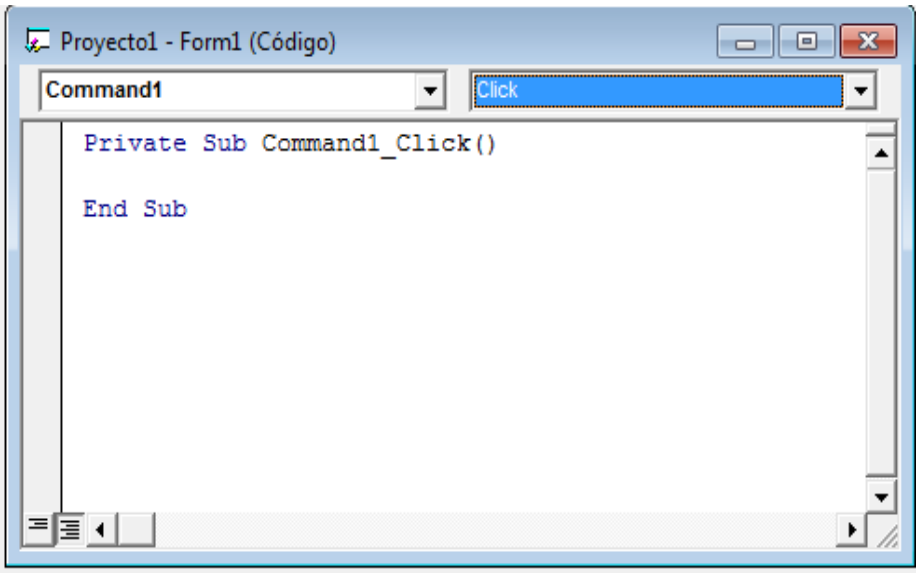


Figura 2. 9 Ventana de Código por defecto

Aquí empezamos a programar, teniendo el siguiente programa:

```
Dim dato1 As String
Dim CI As String
Dim Dato As Integer
Dim X As Integer
Sub Pausams()
e = Timer
Do
f = Timer - e
Loop While f < 0.02
End Sub
Sub recibir()
form1.Timer1.Enabled = True
 Do
  Ret = DoEvents()
 Loop While form1.MSComm1.InBufferCount = 0
form1.Timer1.Enabled = False
dato1 = form1.MSComm1.Input
form1.Label10.Caption = dato1
 '       If Form1.Label4.Caption <> Form1.Label10.Caption Then
 '         Respuesta = MsgBox("error")
 '       End If
Call Pausams
```

```
End Sub

Private Sub Command1_Click()

Dim OpenFlag

Dim X1 As String

Dim y As Integer

Dim z As Integer

   X = 0

   y = 1

   z = 0

Label8.Caption = Format(y, "00000")

OpenFlag = form1.MSComm1.PortOpen

If OpenFlag Then

Else

form1.MSComm1.PortOpen = Not form1.MSComm1.PortOpen

End If

Command1.Enabled = False

Command2.Enabled = False

Command5.Enabled = False

CmdSalir.Enabled = False

form1.Label4.Caption = "1"

form1.MSComm1.Output = "1" + Chr(13)

Do
```

```
Ret = DoEvents()
Loop Until form1.MSComm1.OutBufferCount = 0
Call Pausams
Call recibir
Dato = Val(dato1)
If Dato = 0 Then
Mensaje = " No existen DATOS "    ' Define el mensaje.
Estilo = vbOKOnly + vbExclamation   ' Define los botones.
Título = "Control de Tiempo"   ' Define el título.
Respuesta = MsgBox(Mensaje, Estilo, Título, Ayuda, Ctxt)
form1.Label4.Caption = ""
form1.Label10.Caption = ""
Command1.Enabled = True
Command2.Enabled = True
Command5.Enabled = True
CmdSalir.Enabled = True
ProgressBar1.Value = 0
form1.MSComm1.InBufferCount = 0
form1.MSComm1.OutBufferCount = 0
form1.MSComm1.PortOpen = False
Exit Sub
Else
```

```
Text1.Text = "Número de Datos a Descargar = " + dato1 + vbCrLf + vbCrLf

Text1.Text = Text1.Text + "Items " + "USUARIO " + "HORA " + "FECHA" + vbCrLf + vbCrLf

Text1.Text = Text1.Text + Label8.Caption + "  "

ProgressBar1.Max = Dato * 2

End If

Do

form1.MSComm1.Output = "w" + Chr(13)

Do

Ret = DoEvents()

Loop Until form1.MSComm1.OutBufferCount = 0

Call Pausams

Call Pausams

Call Pausams

Call recibir

If dato1 = "FF" Then

y = y + 1

If y <= Dato Then

Label8.Caption = Format(y, "00000")

Text1.Text = Text1.Text + vbCrLf + Label8.Caption + "  "

End If

Else

dato2 = Mid$(dato1, 1, 20) + "  "
```

```
dato3 = Mid$(dato1, 21, 8) + "  "

dato4 = Mid$(dato1, 29, 12)

dato5 = dato2 + dato3 + dato4

Text1.Text = Text1.Text + dato5

End If

X = X + 1

ProgressBar1.Value = X

Loop While (Dato * 2 > X)

Command1.Enabled = True

Mensaje = " DESCARGA OK "    ' Define el mensaje.

Estilo = vbOKOnly + vbExclamation    ' Define los botones.

Título = "Control de Tiempo"    ' Define el título.

Respuesta = MsgBox(Mensaje, Estilo, Título, Ayuda, Ctxt)

form1.Label4.Caption = ""

form1.Label10.Caption = ""

Command1.Enabled = True

Command2.Enabled = True

Command5.Enabled = True

CmdSalir.Enabled = True

ProgressBar1.Value = 0

form1.MSComm1.InBufferCount = 0

form1.MSComm1.OutBufferCount = 0
```

```
form1.MSComm1.PortOpen = False
End Sub
Private Sub Command2_Click()
Dim OpenFlag
Label8.Caption = Format(y, "00000")
OpenFlag = form1.MSComm1.PortOpen
If OpenFlag Then
Else
form1.MSComm1.PortOpen = Not form1.MSComm1.PortOpen
End If
Command1.Enabled = False
Command5.Enabled = False
CmdSalir.Enabled = False
form1.Label4.Caption = "2"
form1.MSComm1.Output = "2" + Chr(13)
Do
Ret = DoEvents()
Loop Until form1.MSComm1.OutBufferCount = 0
Call Pausams
Mensaje = " CONTADOR RESET "    ' Define el mensaje.
Estilo = vbOKOnly + vbExclamation   ' Define los botones.
Título = "Control de Tiempo"   ' Define el título.
```

```
Respuesta = MsgBox(Mensaje, Estilo, Título, Ayuda, Ctxt)

form1.Label4.Caption = ""

form1.Label10.Caption = ""

Command1.Enabled = True

Command5.Enabled = True

CmdSalir.Enabled = True

ProgressBar1.Value = 0

form1.MSComm1.InBufferCount = 0

form1.MSComm1.OutBufferCount = 0

form1.MSComm1.PortOpen = False

End Sub

Private Sub Command5_Click()

If Text1.Text <> "" Then

Dim cFile, cCadena As String

Command1.Enabled = False

Command5.Enabled = False

Dialogo.Filter = "Archivos de Texto|*.Txt|"

Dialogo.DialogTitle = "Guardar Como Archivo de Texto"

Dialogo.ShowSave

cFile = Dialogo.FileName

Open cFile For Output As #1

cCadena = Text1.Text
```

```
Print #1, cCadena
Close #1
End If
Command1.Enabled = True
Command5.Enabled = True
End Sub
Private Sub CmdSalir_Click()
If form1.MSComm1.PortOpen Then form1.MSComm1.PortOpen = False
End
End Sub
Private Sub Form_Load()
Dim c As Integer
form1.lblportico.Caption = "Com" & form1.MSComm1.CommPort
form1.lblvelocidad.Caption = Left$(form1.MSComm1.Settings,
Len(form1.MSComm1.Settings) - 6)
Command5.Enabled = False
End Sub
Private Sub MnuAcerca_Click()
frmAbout.Show vbModal
End Sub
Private Sub MnuClaves_Click()
Form3.Show vbModal
End Sub
```

```
Private Sub MnuPropiedades_Click()

Form2.Show vbModal

End Sub

Private Sub MnuSalir_Click()

CmdSalir_Click

End Sub

Private Sub Text1_Change()

End Sub

Private Sub Timer1_Timer()

form1.Timer1.Enabled = False

form1.MSComm1.InBufferCount = 0

form1.MSComm1.OutBufferCount = 0

form1.MSComm1.PortOpen = False

Mensaje = " Error en Comunicación "    ' Define el mensaje.

Estilo = vbOKOnly + vbExclamation   ' Define los botones.

Título = " ERROR EN COMUNICACION "    ' Define el título.

Respuesta = MsgBox(Mensaje, Estilo, Título)

'        CmdTrans.Enabled = True

        CmdSalir.Enabled = True

End Sub.
```

Una vez realizado el primer formulario el procedimiento realizado es el mismo para los otros.

Procedemos a realizar otro formulario, cuya ventana se presentará así, figura 2.10.

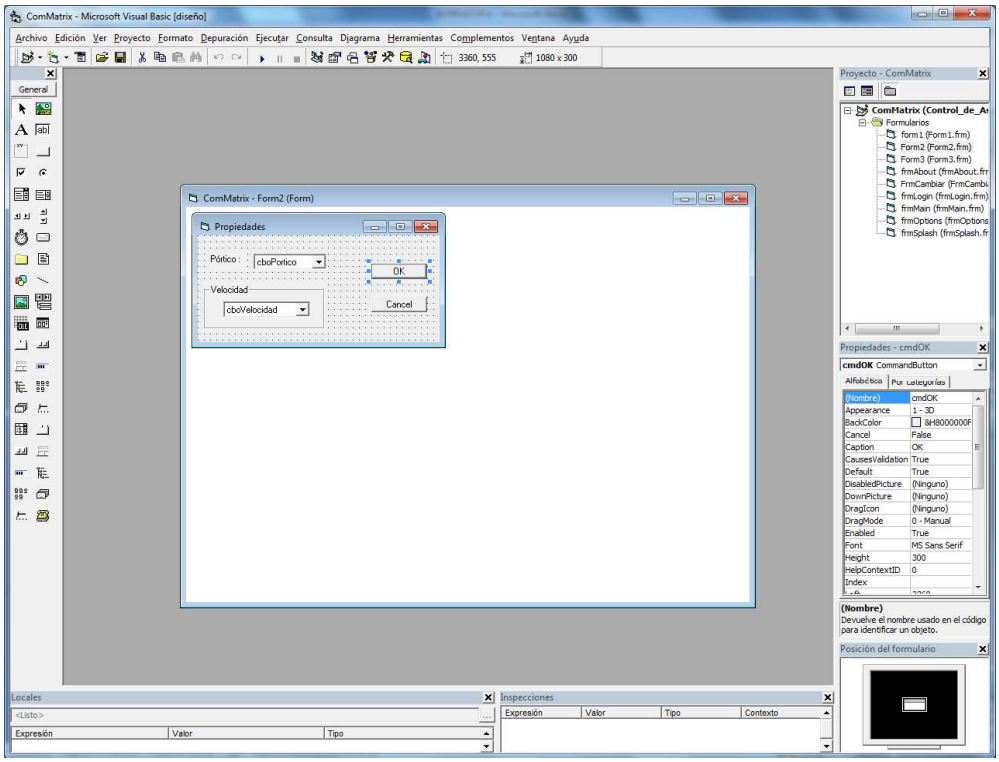


Figura 2. 10 Formulario 2

El programa implementado es el siguiente, el cual nos va servir en la comunicación para la selección del pórtico a utilizar:

```
Private Sub cmdCancel_Click()

Unload Me

End Sub

Private Sub cmdOK_Click()

form1.MSComm1.CommPort = cboPortico.ListIndex + 1

form1.MSComm1.Settings = Trim$(cboVelocidad.Text) & "," & "n" & "," & "8" & "," & "1"

form1.lblPortico.Caption = "Com" & form1.MSComm1.CommPort
```

```
form1.lblvelocidad.Caption = Left$(form1.MSComm1.Settings,
Len(form1.MSComm1.Settings) - 6)

Unload Me

End Sub

Private Sub Form_Load()

For i = 1 To 15

cboPortico.AddItem "Com" & Trim$(Str$(i))

Next i

cboPortico.ListIndex = form1.MSComm1.CommPort - 1

cboVelocidad.AddItem "110"

cboVelocidad.AddItem "300"

cboVelocidad.AddItem "600"

cboVelocidad.AddItem "1200"

cboVelocidad.AddItem "2400"

cboVelocidad.AddItem "4800"

cboVelocidad.AddItem "9600"

cboVelocidad.AddItem "14400"

cboVelocidad.AddItem "19200"

cboVelocidad.AddItem "28800"

cboVelocidad.AddItem "38400"

cboVelocidad.Text = Left$(form1.MSComm1.Settings,
Len(form1.MSComm1.Settings) - 6)

End Sub
```

El tercer formulario es el siguiente:

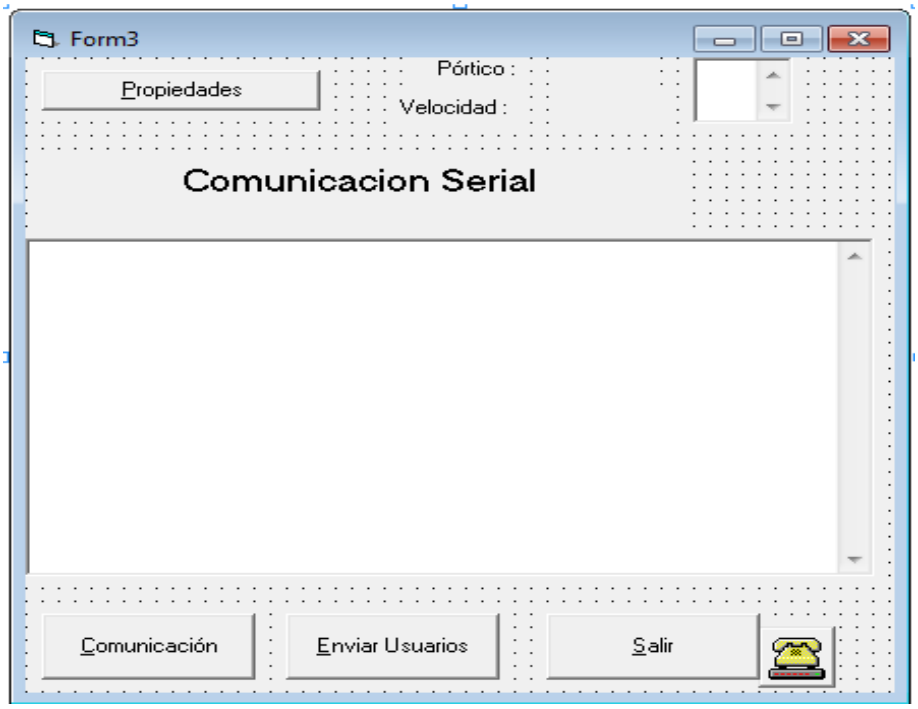


Figura 2. 11 Formulario 3

Este formulario estamos programándole para el envío de claves de los usuarios y para la comunicación figura 2.11.

```
Dim y As Byte

Sub Pausams()

e = Timer

Do

f = Timer - e

Loop While f < 0.02

End Sub

Private Sub CmdPropiedades_Click()

Form2.Show vbModal
```

```
End Sub

Private Sub CmdSalir_Click()

If form1.MSComm1.PortOpen Then form1.MSComm1.PortOpen = False

Unload Me

End Sub

Private Sub CmdTrans_Click()

Dim OpenFlag

OpenFlag = form1.MSComm1.PortOpen

If OpenFlag Then

Else

form1.MSComm1.PortOpen = Not form1.MSComm1.PortOpen

End If

CmdPropiedades.Enabled = False

CmdTrans.Enabled = False

End Sub

Private Sub Command2_Click()

Command2.Enabled = False

Text2.Text = ""

form1.MSComm1.Output = "A" + Chr(13)

Do

Ret = DoEvents()

Loop Until form1.MSComm1.OutBufferCount = 0
```

```
Open "Usuarios1.txt" For Input As #1    ' Abre el archivo.

flag1 = 1

Do While Not EOF(1)    ' Repite el bucle hasta el final del archivo.Line Input #1, LíneaTexto    ' Lee el carácter en la variable.

If flag1 = 1 Then

flag1 = 0

Text1.Text = Mid(LíneaTexto, 1, 4)

form1.MSComm1.Output = LíneaTexto + Chr(13)

Do

Ret = DoEvents()

Loop Until form1.MSComm1.OutBufferCount = 0

For X = 1 To 10

Call Pausams

Next X

Else

'            Form1.MSComm1.Output = LíneaTexto + Chr(13)

           Text2.Text = Text2.Text + LíneaTexto + vbCrLf

For c = 1 To 24

d = Mid(LíneaTexto, c, 1)

form1.MSComm1.Output = d + Chr(13)

Do

Ret = DoEvents()

Loop Until form1.MSComm1.OutBufferCount = 0
```

```
Call Pausams

Next c

End If

Call Pausams

Loop

Close #1   ' Cierra el archivo.

Command2.Enabled = True

End Sub

Private Sub Form_Load()

form1.lblportico.Caption = "Com" & form1.MSComm1.CommPort

form1.lblvelocidad.Caption = Left$(form1.MSComm1.Settings, Len
(form1.MSComm1.Settings) - 6)

y = 0

End Sub
```

El siguiente formulario nos muestra la información del programa figura 2.12.

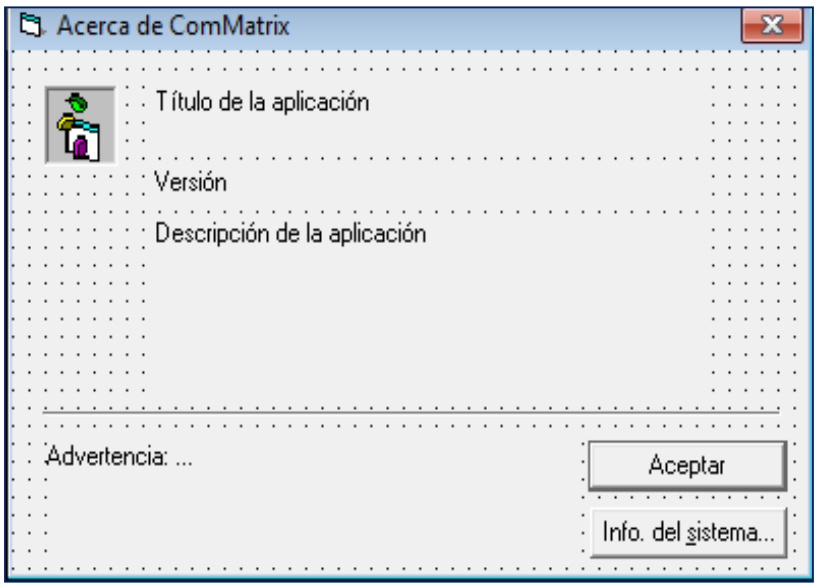


Figura 2. 12 Formulario 4

La programación para este es la siguiente:

```
' Opciones de seguridad de clave del Registro...

Const KEY_ALL_ACCESS = &H2003F

 ' Tipos ROOT de claves del Registro...

Const HKEY_LOCAL_MACHINE = &H80000002

Const ERROR_SUCCESS = 0

Const REG_SZ = 1                    ' cadena terminada en valor nulo Unicode

Const REG_DWORD = 4                 ' número de 32 bits

Const gREGKEYSYSINFOLOC = "SOFTWARE\Microsoft\Shared Tools Location"

Const gREGVALSYSINFOLOC = "MSINFO"

Const gREGKEYSYSINFO = "SOFTWARE\Microsoft\Shared Tools\MSINFO"

Const gREGVALSYSINFO = "PATH"

Private Declare Function RegOpenKeyEx Lib "advapi32" Alias "RegOpenKeyExA" (ByVal hKey As Long, ByVal lpSubKey As String, ByVal ulOptions As Long, ByVal samDesired As Long, ByRef phkResult As Long) As Long

Private Declare Function RegQueryValueEx Lib "advapi32" Alias "RegQueryValueExA" (ByVal hKey As Long, ByVal lpValueName As String, ByVal lpReserved As Long, ByRef lpType As Long, ByVal lpData As String, ByRef lpcbData As Long) As Long

Private Declare Function RegCloseKey Lib "advapi32" (ByVal hKey As Long) As Long

Private Sub Form_Load()

lblVersion.Caption = "Versión " & App.Major & "." & App.Minor & "." & App.Revision

lblTitle.Caption = App.Title
```

```
End Sub

Private Sub cmdSysInfo_Click()

Call StartSysInfo

End Sub

Private Sub cmdOK_Click()

Unload Me

End Sub

Public Sub StartSysInfo()

On Error GoTo SysInfoErr

Dim rc As Long

Dim SysInfoPath As String

' Intentar obtener el nombre y la ruta del programa en el Registro...

If GetKeyValue(HKEY_LOCAL_MACHINE, gREGKEYSYSINFO, gREGVALSYSINFO, SysInfoPath) Then

' Intentar obtener sólo la ruta del programa en el Registro...

ElseIf GetKeyValue(HKEY_LOCAL_MACHINE, gREGKEYSYSINFOLOC, gREGVALSYSINFOLOC, SysInfoPath) Then

' Validar la existencia de versión conocida de 32 bits de archivo

If (Dir(SysInfoPath & "\MSINFO32.EXE") <> "") Then

SysInfoPath = SysInfoPath & "\MSINFO32.EXE"

'Error: no se encuentra el archivo...

Else

GoTo SysInfoErr
```

```
End If

'Error: no se encuentra la entrada del Registro...

Else

GoTo SysInfoErr

End If

Call Shell(SysInfoPath, vbNormalFocus)

Exit Sub

SysInfoErr:

MsgBox "La información del sistema no está disponible en este momento", vbOKOnly

End Sub

Public Function GetKeyValue(KeyRoot As Long, KeyName As String, SubKeyRef As String, ByRef KeyVal As String) As Boolean

Dim i As Long                           ' Contador de bucle

Dim rc As Long                          ' Código de retorno

Dim hKey As Long                        ' Controlador a una clave de Registro abierta

Dim hDepth As Long                      '

Dim KeyValType As Long                  ' Tipo de datos de una clave del Registro

Dim tmpVal As String                    ' Almacenamiento temporal para un valor de clave del Registro

Dim KeyValSize As Long                  ' Tamaño de variable de clave del Registro

'------------------------------------------------------------
```

```
' Abrir RegKey bajo KeyRoot {HKEY_LOCAL_MACHINE...}

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

rc = RegOpenKeyEx(KeyRoot, KeyName, 0, KEY_ALL_ACCESS, hKey) ' Abrir la clave del Registro

If (rc <> ERROR_SUCCESS) Then GoTo GetKeyError          ' Controlar error...

tmpVal = String$(1024, 0)                               ' Asignar espacio de variable

KeyValSize = 1024                                       ' Marcar tamaño de variable

'-----------------Obtener valor de clave del Registro...

rc = RegQueryValueEx(hKey, SubKeyRef, 0, KeyValType, tmpVal, KeyValSize)    ' Get/Create Key Value

If (rc <> ERROR_SUCCESS) Then GoTo GetKeyError          ' Controlar errores

tmpVal = VBA.Left(tmpVal, InStr(tmpVal, VBA.Chr(0)) - 1)

  '-------------Determinar el tipo de valor de clave para conversión...

Select Case KeyValType                                  ' Buscar tipos de datos...

Case REG_SZ                                             ' Tipo de datos String de clave del Registro

KeyVal = tmpVal                                         ' Copiar valor String

Case REG_DWORD                                          ' Tipo de datos Double Word de clave del Registro

For i = Len(tmpVal) To 1 Step -1                        ' Convertir cada bit

KeyVal = KeyVal + Hex(Asc(Mid(tmpVal, i, 1)))     ' Generar valor carácter a carácter

Next

KeyVal = Format$("&h" + KeyVal)                         ' Convertir Double Word a String
```

```
End Select
GetKeyValue = True                          ' Operación realizada correctamente
rc = RegCloseKey(hKey)                      ' Cerrar clave del Registro
Exit Function                               ' Salir
GetKeyError:    ' Limpiar después de que se produzca un error...
KeyVal = ""                                 ' Establecer el valor de retonor a la cadena vacía
GetKeyValue = False          ' La operación no se ha realizado correctamente
rc = RegCloseKey(hKey)                      ' Cerrar clave del Registro
End Function
```

El siguiente formulario nos permitirá el ingreso de una clave al inicio del programa, figura 2.13.

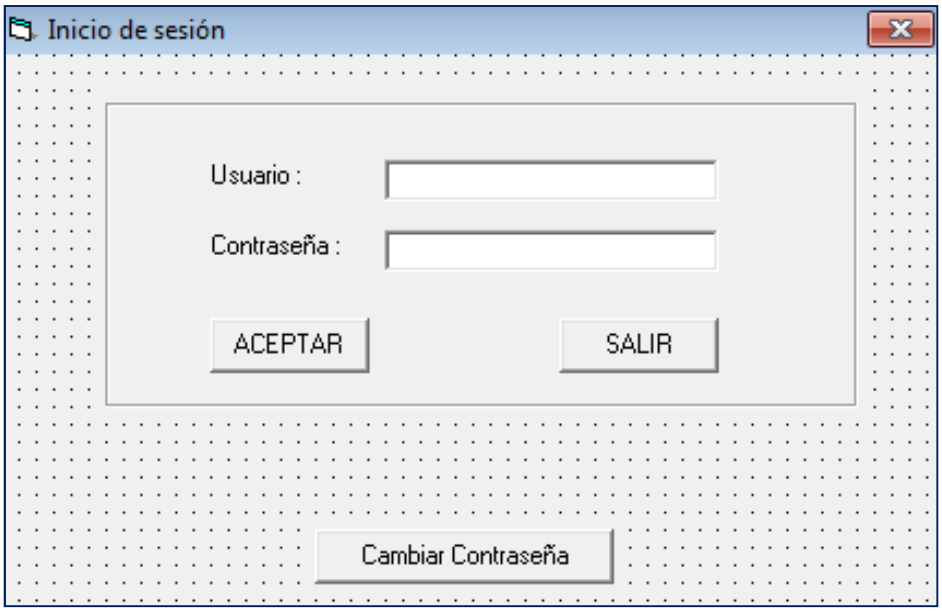


Figura 2. 13 Formulario 5

Para lo cual se realiza el siguiente programa:

```
Private Sub CmdCambiarp_Click()
```

```
FrmCambiar.Show vbModal

End Sub

Private Sub Form_Load()

Dim h As String

On Error GoTo a

Dim sBuffer As String

Dim lSize As Long

sBuffer = Space$(255)

lSize = Len(sBuffer)

Call GetUserName(sBuffer, lSize)

If lSize > 0 Then

txtUserName.Text = Left$(sBuffer, lSize)

Else

txtUserName.Text = vbNullString

End If

Open App.Path & "\data.dat" For Input As #1

Line Input #1, h

txtdpassw.Text = h

Close

Dim Textoriginal As String

Textoriginal = EncryptString("1", txtdpassw, DECRYPT)

txtdpassw.Text = Textoriginal
```

```
a:
If Err.Number = 76 Then
Mensaje = " No existe el archivo DATA.DAT  "    ' Define el mensaje.
Estilo = vbOKOnly + vbExclamation    ' Define los botones.
Título = " Inicio de sesión "    ' Define el título.
Respuesta = MsgBox(Mensaje, Estilo, Título)
End
End If
End Sub
Private Sub CmdSalir_Click()
OK = False
Me.Hide
End Sub
Private Sub cmdOK_Click()
'Pendiente: crear comprobación de contraseña
'comprobar si la contraseña es correcta
Static numintentos As Byte
If TxtPassword.Text = frmLogin.txtdpassw.Text Then
OK = True
Me.Hide
Else
MsgBox "La contraseña no es válida; vuelva a intentarlo", vbCritical, "Inicio de sesión"
```

```
TxtPassword.SetFocus

TxtPassword.SelStart = 0

TxtPassword.SelLength = Len(TxtPassword.Text)

numintentos = numintentos + 1

End If

If numintentos > 2 Then

OK = False

Me.Hide

End If

End Sub

Private Sub TxtPassword_KeyPress(KeyAscii As Integer)

If KeyAscii = 13 Then

Call cmdOK_Click

End If

End Sub
```

El siguiente formulario nos va a permitir un cambio de contraseña del programa.

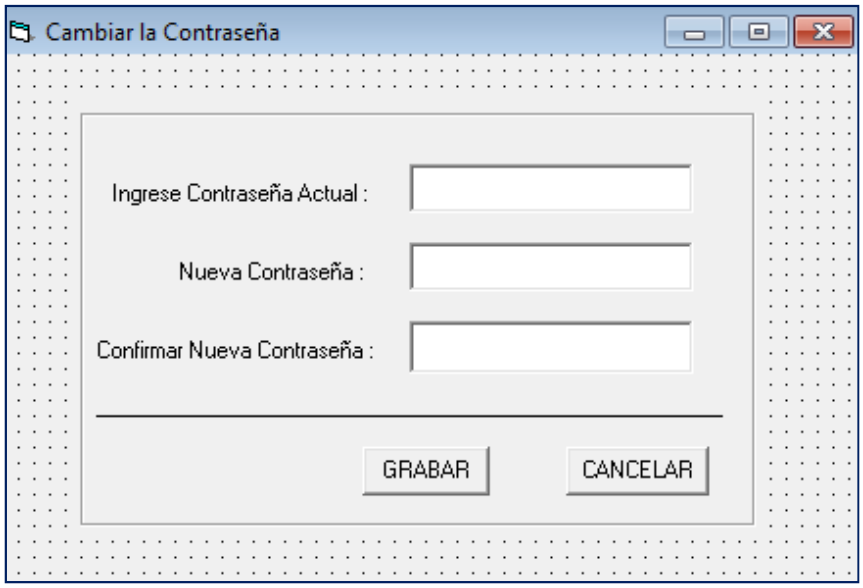


Figura 2. 14 Formulario 4

La programación es la siguiente:

```
Function Comprobar () As Boolean

If TxtActual.Text = "" Then

NCA = MsgBox ("Debe ingresar la contraseña actual.", vbExclamation, "Cambiar Contraseña de Usuario.")

TxtActual.SetFocus

Comprobar = False

Exit Function

End If

If TxTNueva.Text = "" Then

NCN = MsgBox("Debe ingresar la nueva contraseña.", vbExclamation, "Cambiar Contraseña de Usuario.")

TxTNueva.SetFocus

Comprobar = False

Exit Function
```

```
End If

If TxtConfirmar.Text = "" Then

NCon = MsgBox("Debe confirmar la contraseña.", vbExclamation, "Cambiar Contraseña de Usuario.")

TxtConfirmar.SetFocus

Comprobar = False

Exit Function

End If

If TxtActual.Text <> frmLogin.txtdpassw.Text Then

NCA = MsgBox("Es incorrecta la contraseña actual.", vbExclamation, "Cambiar Contraseña de Usuario.")

TxtActual.SelStart = 0

TxtActual.SelLength = Len (TxtContraseñaA)

TxtActual.SetFocus

Comprobar = False

Exit Function

End If

Comprobar = True

End Function

Private Sub CmdCancelar_Click()

FrmCambiar.Hide

Unload Me

End Sub
```

```
Private Sub CmdGrabar_Click()

If Comprobar = True Then

Preguntar = MsgBox("Confirma cambiar contraseña de usuario?", vbQuestion + vbYesNo, "Cambiar Contraseña de Usuario.")

If Preguntar = 6 Then

If TxtConfirmar.Text <> TxTNueva.Text Then

OK = MsgBox("La contraseña Nueva y Confirmar son diferentes.", vbInformation, "Cambiar Contraseña de Usuario.")

Else

If Len (TxTNueva.Text) < 6 Or Len(TxTNueva.Text) > 15 Then

MsgBox "La contraseña debe tener un minimo de 6 caracteres y un máximo de 15", vbExclamation, "Atencion!"

TxTNueva.SetFocus

SendKeys "{home}+{end}"

Else

Dim textocodificado As String

frmLogin.txtdpassw.Text = TxTNueva.Text

textocodificado = EncryptString("1", TxTNueva, ENCRYPT)

Open App.Path & "\data.dat" For Output As #1

Close

OK = MsgBox("La contraseña se ha cambiado satisfactoriamente.", vbInformation, "Cambiar Contraseña de Usuario.")

FrmCambiar.Hide
```

```
Unload Me

End If

End If

End If

End If

End Sub
```

Después se realiza el formulario, como una carátula, que se mostrará cuando se abra el programa figura 2.15.

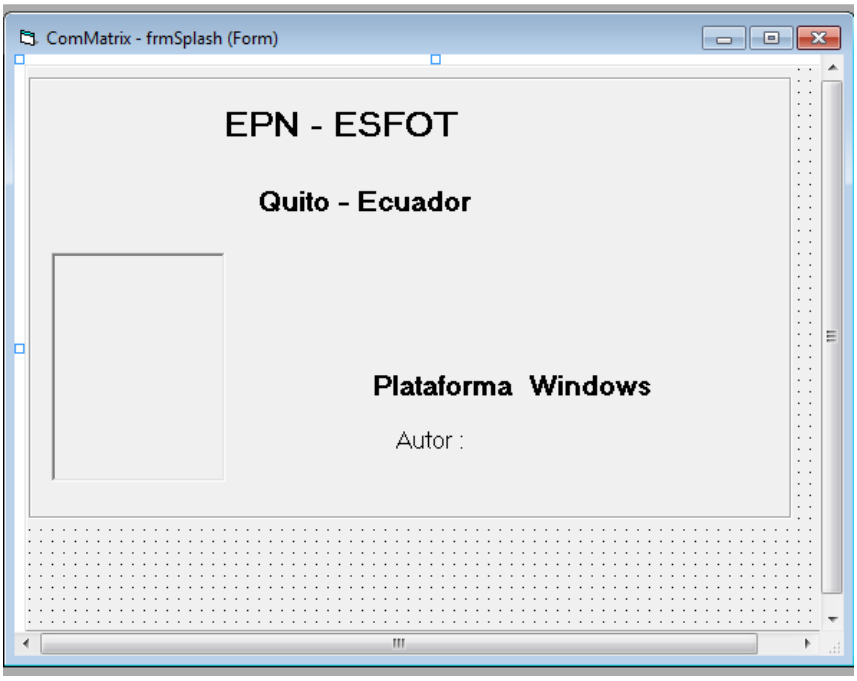


Figura 2. 15 formulario 7

La programación es la siguiente:

```
Private Sub Form_Load()

'       lblVersion.Caption = "Version" & App.Major & "." & App.Minor & "." & App.Revision

    lblProductName.Caption = App.Title

End Sub
```

```
Private Sub fraMainFrame_DblClick()

End Sub

Private Sub fraMainFrame_DragDrop(Source As Control, X As Single, Y As Single)

End Sub

Private Sub fraMainFrame_OLECompleteDrag(Effect As Long)

End Sub

Private Sub fraMainFrame_OLEStartDrag(Data As DataObject, AllowedEffects As Long)

End Sub

Private Sub lblCompanyProduct_Click()

End Sub

Private Sub lblPlatform_Click()

End Sub
```

Después de haber realizado los formularios tenemos que unirlos, para ello en la barra de menús escogemos proyecto, agregar nuevo modulo y creamos un módulo, este nos va a servir para unir todos los formularios y crear nuestro programa, como se muestra en la figura 2.16.

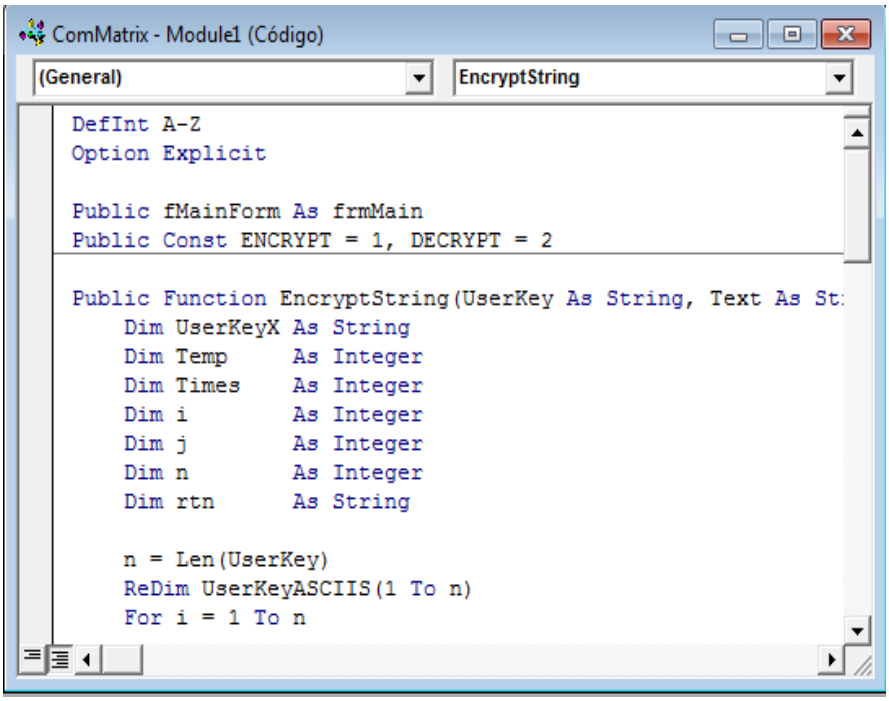


Figura 2. 16 Modulo del Programa

La programación es la siguiente:

DefInt A-Z

Option Explicit

Public fMainForm As frmMain

Public Const ENCRYPT = 1, DECRYPT = 2

Public Function EncryptString(UserKey As String, Text As String, Action As Single) As String

Dim UserKeyX As String

Dim Temp As Integer

Dim Times As Integer

Dim i As Integer

Dim j As Integer

Dim n As Integer

```
Dim rtn      As String

n = Len(UserKey)

ReDim UserKeyASCIIS(1 To n)

For i = 1 To n

UserKeyASCIIS(i) = Asc(Mid$(UserKey, i, 1))

Next

ReDim TextASCIIS(Len(Text)) As Integer

For i = 1 To Len(Text)

TextASCIIS(i) = Asc(Mid$(Text, i, 1))

Next

If Action = ENCRYPT Then

For i = 1 To Len(Text)

j = IIf(j + 1 >= n, 1, j + 1)

Temp = TextASCIIS(i) + UserKeyASCIIS(j)

If Temp > 255 Then

Temp = Temp - 255

End If

rtn = rtn + Chr$(Temp)

Next

ElseIf Action = DECRYPT Then

For i = 1 To Len(Text)

j = IIf(j + 1 >= n, 1, j + 1)
```

```
Temp = TextASCIIS(i) - UserKeyASCIIS(j)
If Temp < 0 Then
Temp = Temp + 255
End If
rtn = rtn + Chr$(Temp)
Next
End If
EncryptString = rtn
End Function
Sub Main()
Dim e As Currency
Dim f As Currency
Dim fLogin As New frmLogin
fLogin.Show vbModal
If Not fLogin.OK Then
'Fallo al iniciar la sesión, se sale de la aplicación
End
End If
Unload fLogin
frmSplash.Show
frmSplash.Refresh
e = Timer
```

```
Do

f = Timer - e

Loop While f < 2

Set fMainForm = New frmMain

Load fMainForm

Unload frmSplash

form1.Show

End Sub
```

Finalmente compilamos y ejecutamos el programa.

2.4 DIAGRAMAS DEL SISTEMA

2.4.1 DIAGRAMA DE FLUJO

El Diagrama de Flujo se utiliza para conocer de una forma general el funcionamiento del circuito. Como se muestra en la figura 2.17.

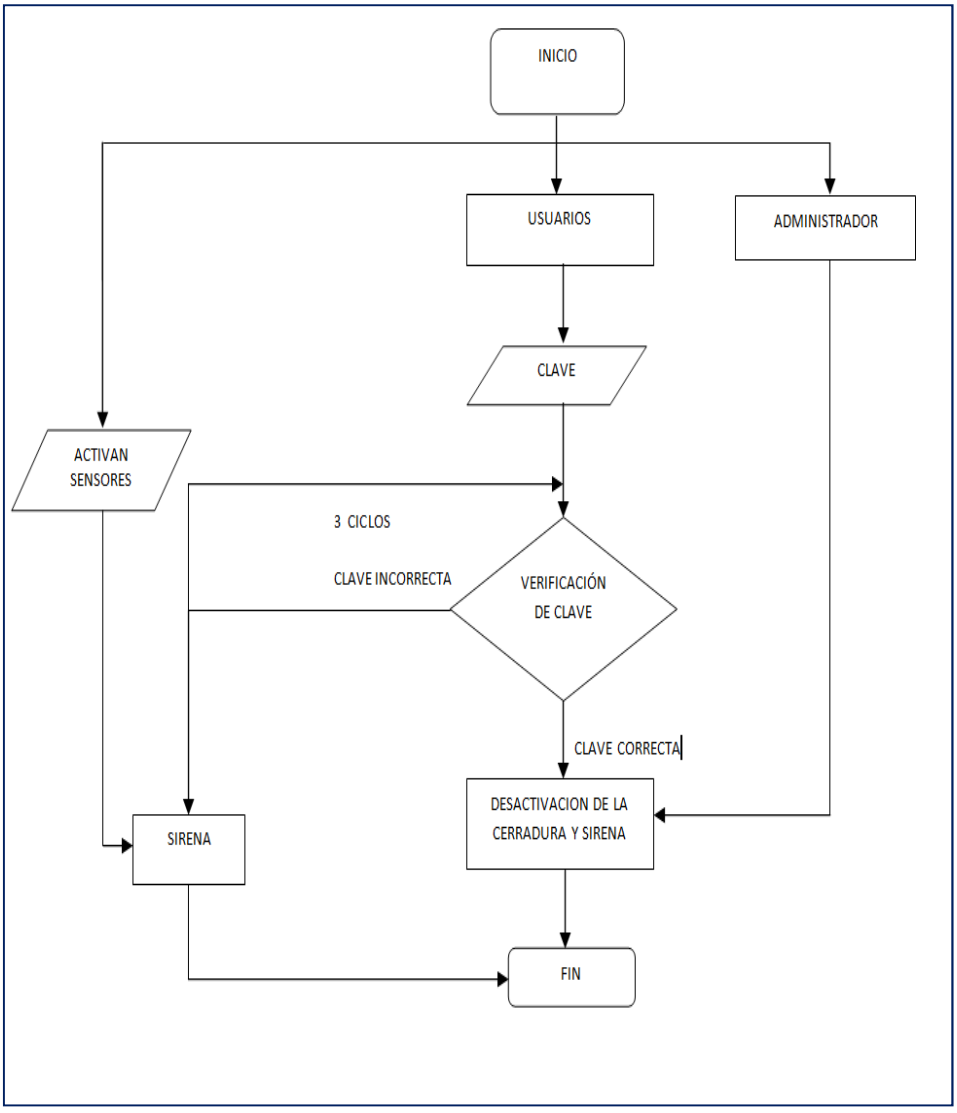


Figura 2. 17 Diagrama de flujo del programa a implementar

2.4.2 DIAGRAMA DE BLOQUES DEL SISTEMA.

Se detalla todos los dispositivos, circuitos y elementos que se incluyen en el diagrama de bloques, como se muestra en la figura 2.18.

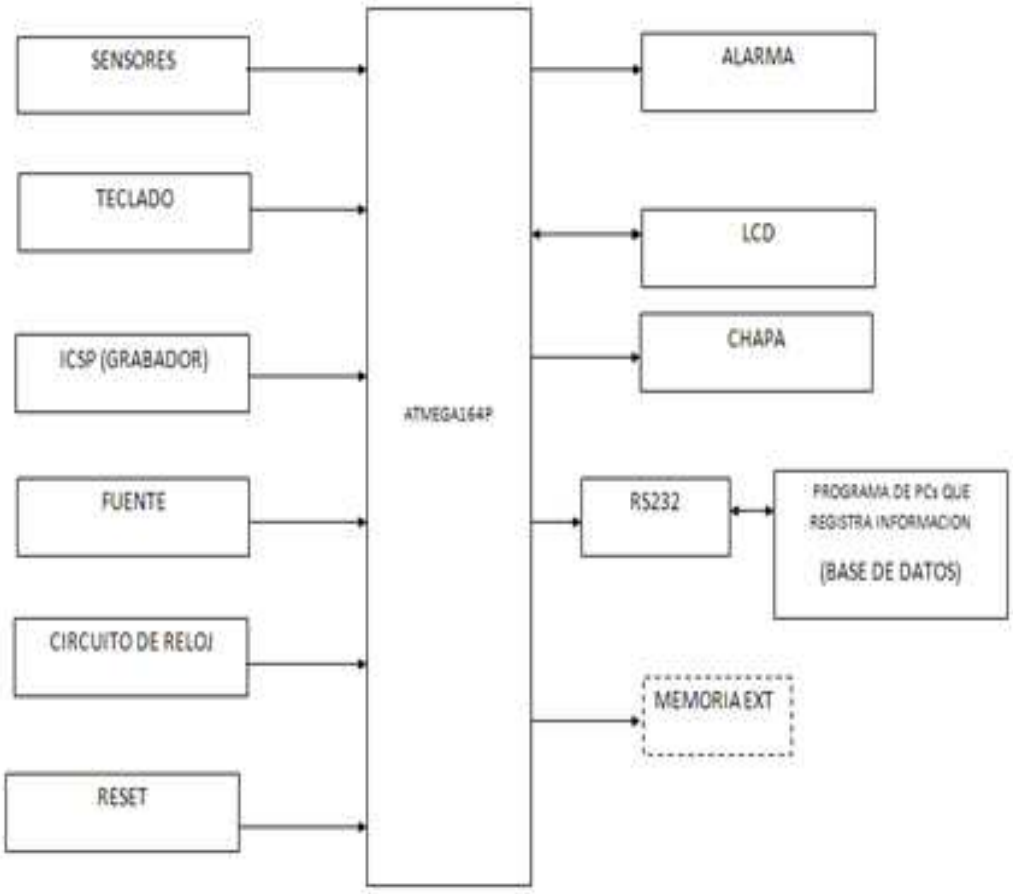


Figura 2. 18 Diagrama de bloques del sistema

Sensores, Teclado, ICSP, fuente, Circuito de reloj y reset constituyen las entradas del sistema que son elementos que controlaran el acceso, el AVR representa el núcleo o parte principal del sistema, este se encarga de recibir las señales de entrada procesa la información y ordena dependiendo del programa a los dispositivos de salida que en este caso son: sirena, LCD, cerradura eléctrica, RS 232, memorias externas.

A continuación se describirá todos los elementos y circuitos.

2.4.3 DIAGRAMAS ESQUEMÁTICOS DE LOS CIRCUITOS QUE CONFORMAN SISTEMA DE ACCESO

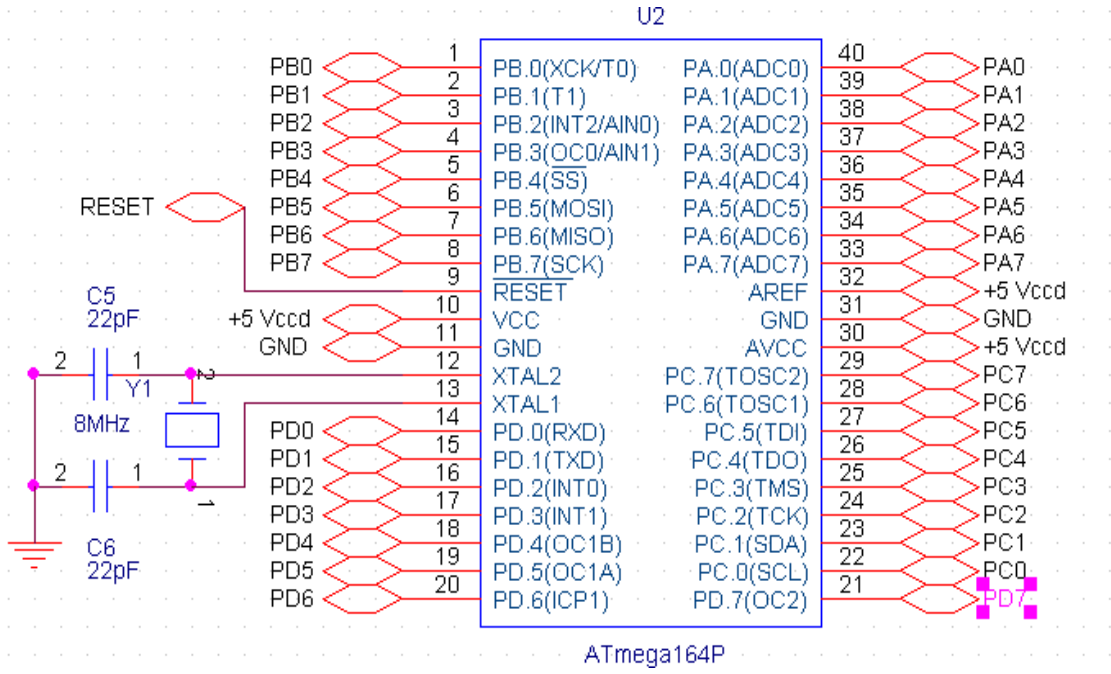


Figura 2. 19 Conexión del AVR

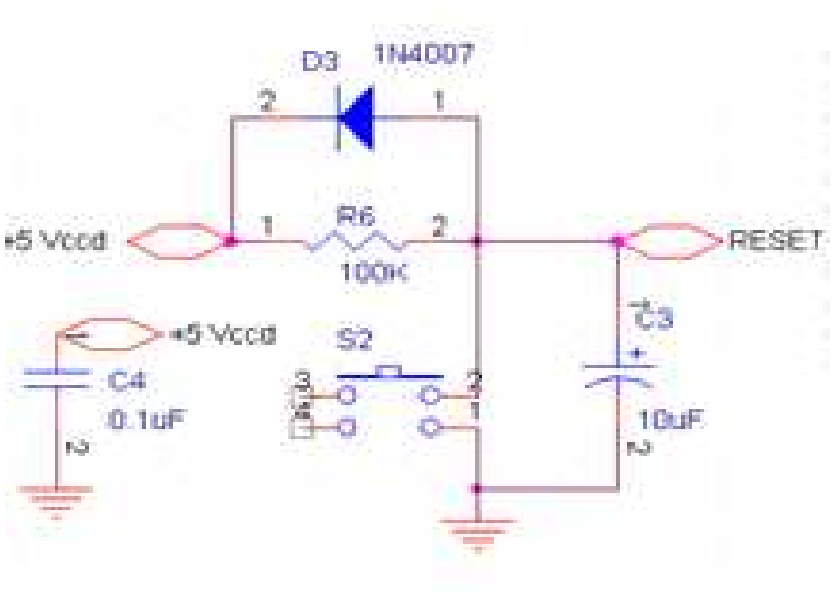


Figura 2. 20 Circuito del Reset

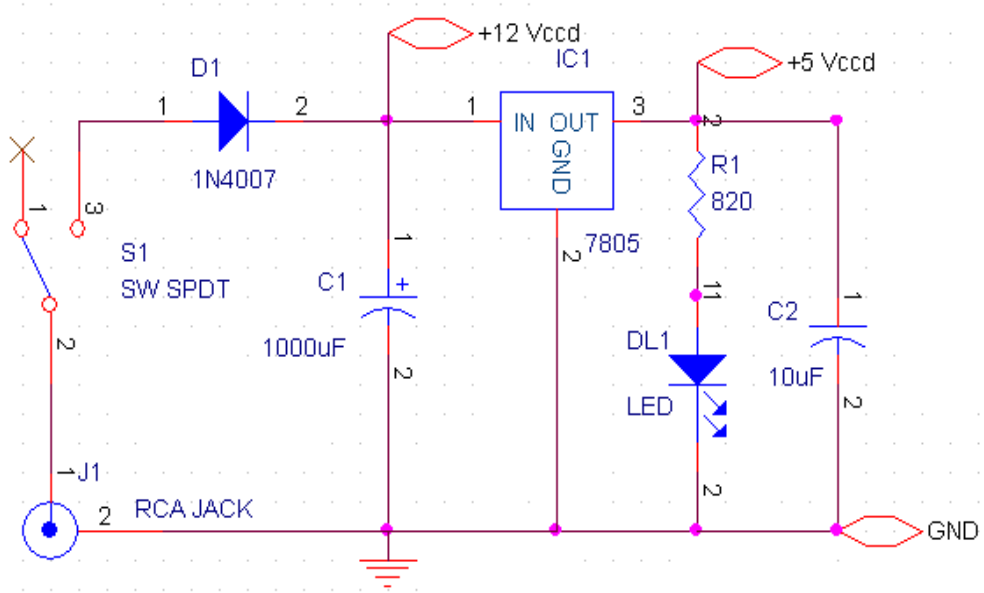


Figura 2. 21 Circuito esquemático de la Fuente de energía.

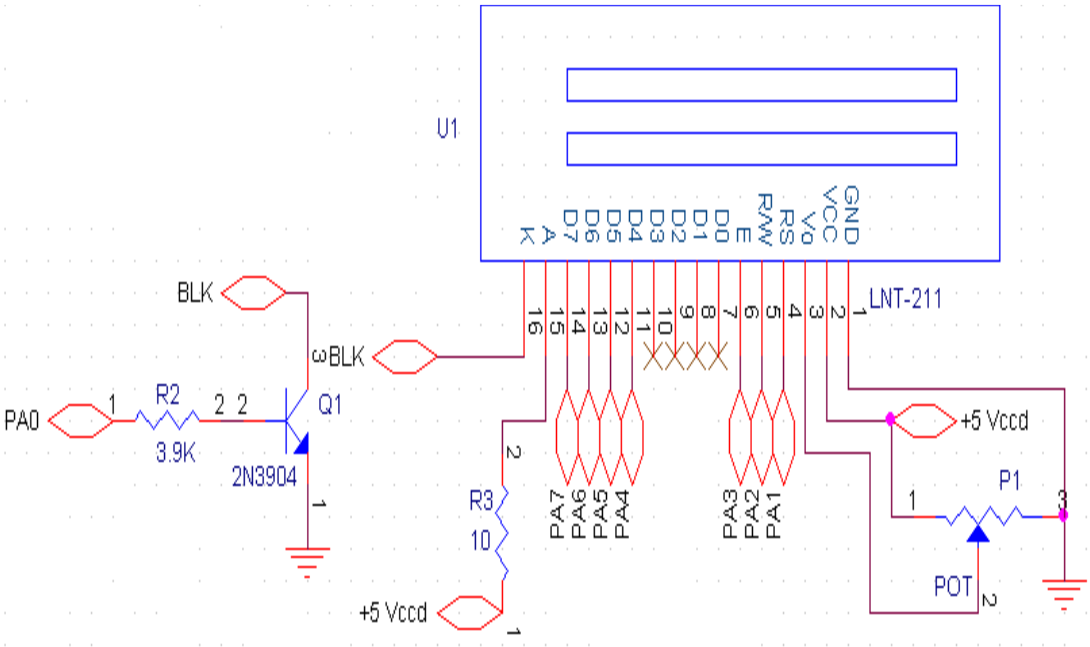


Figura 2. 22 Conexiones LCD 20x4.

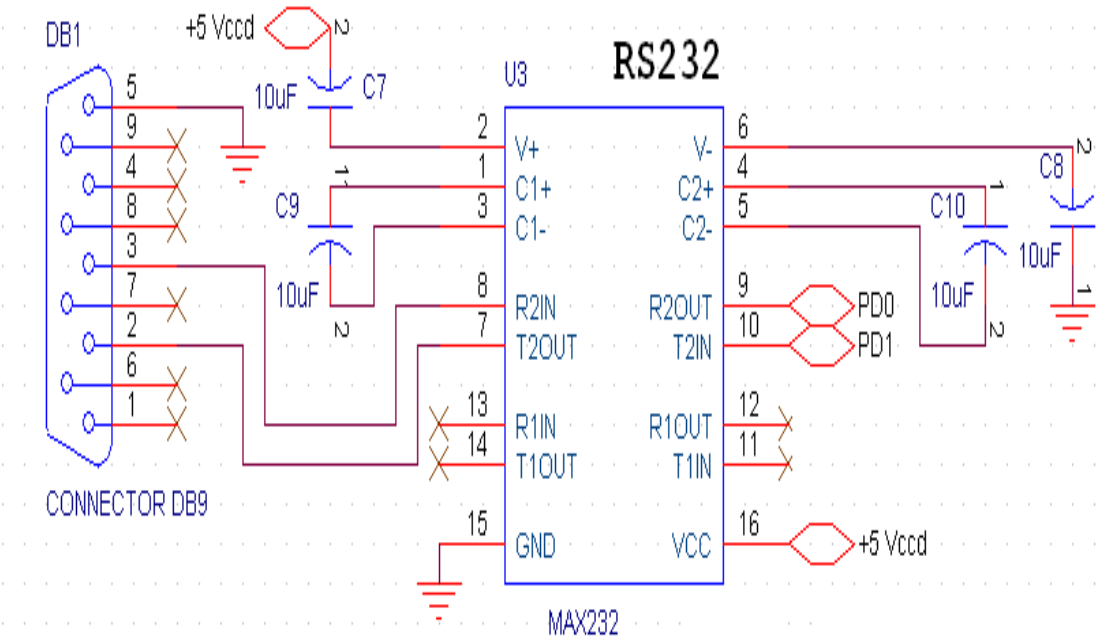


Figura 2. 23 Conexiones del puerto serial RS232.

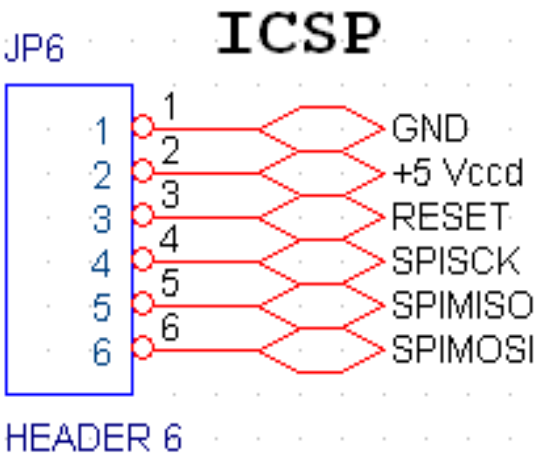


Figura 2. 24 Conexión del grabador ICSP

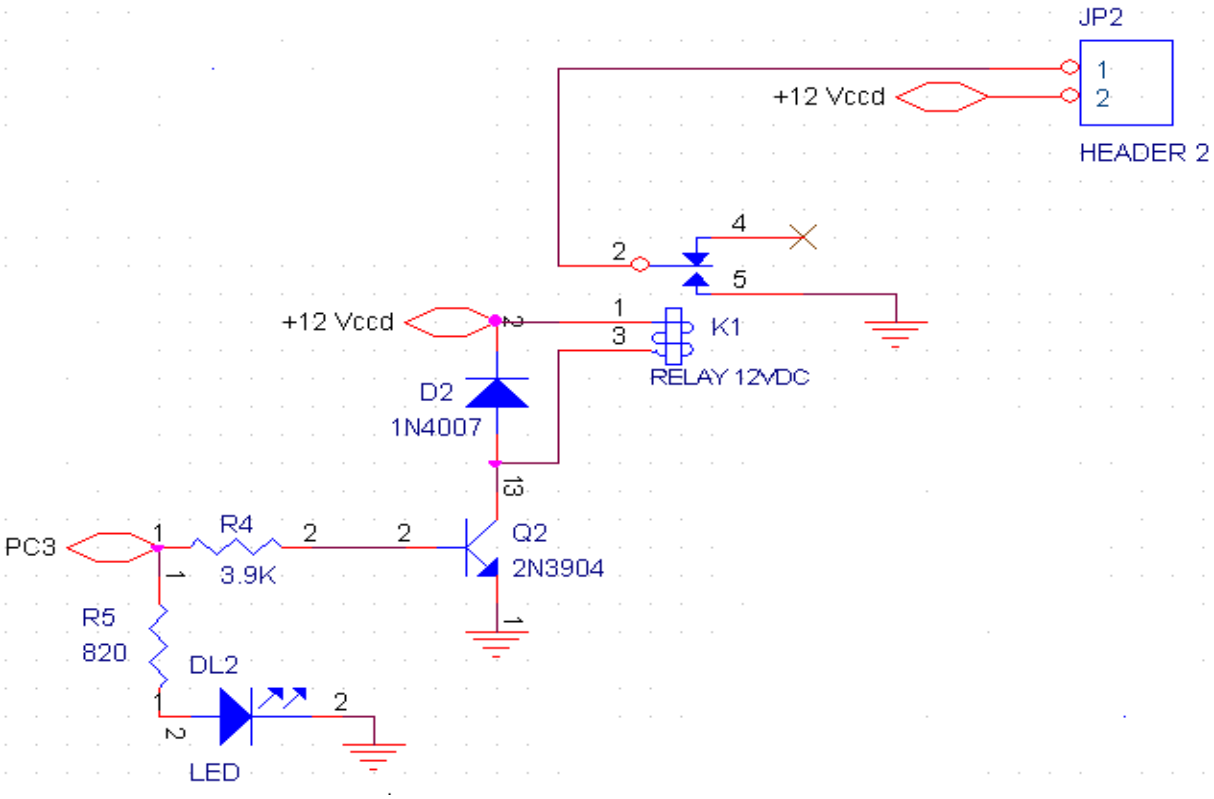


Figura 2. 25 Circuito de la Alarma

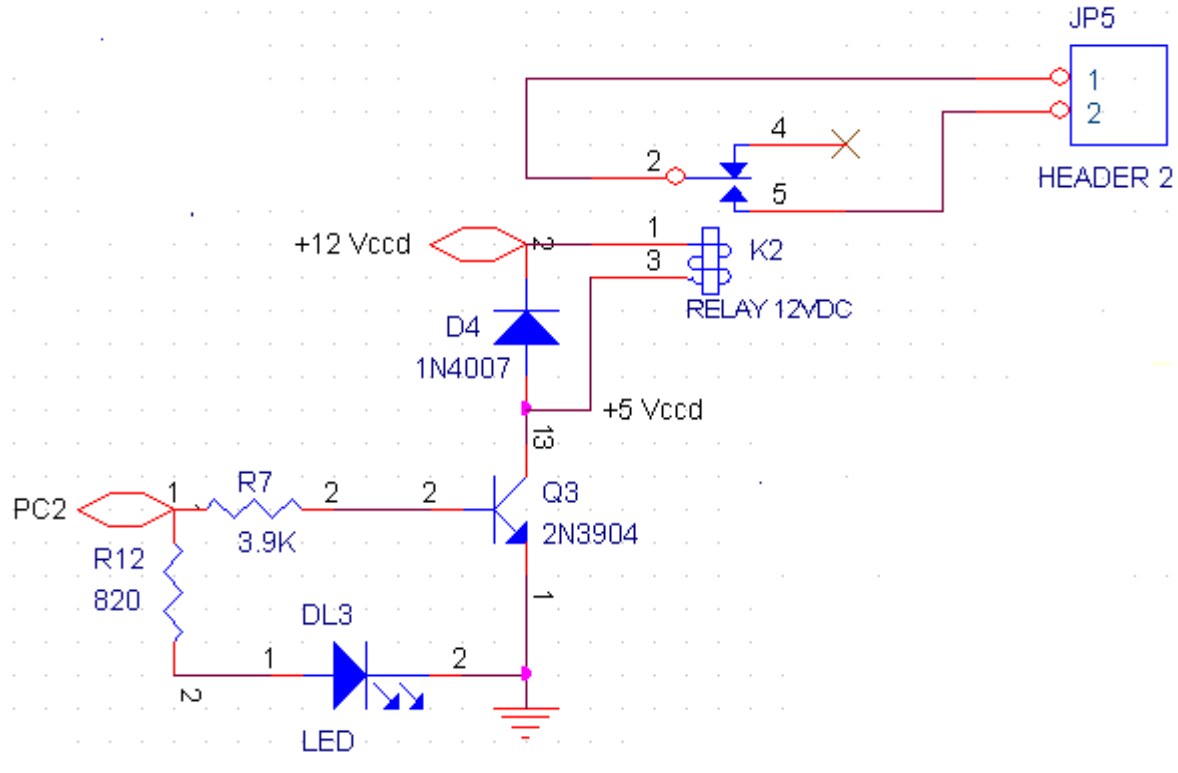


Figura 2. 26 Conexión de la cerradura

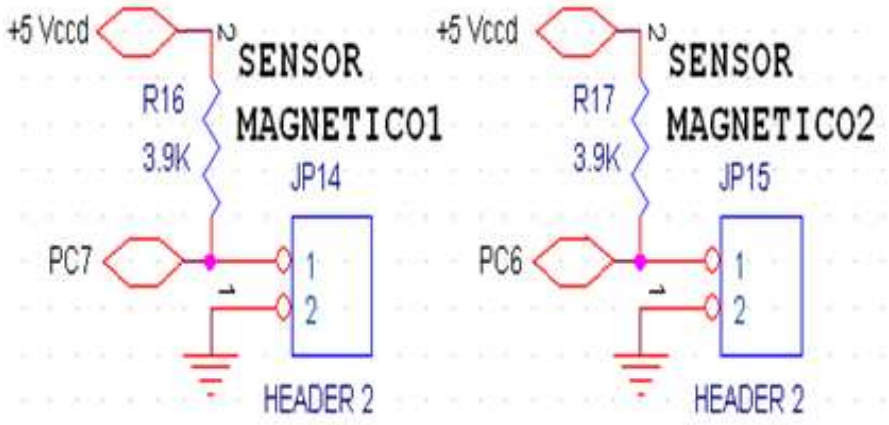


Figura 2. 27 Conexiones de los sensores.

CAPÍTULO 3: IMPLEMENTACIÓN DEL SISTEMA Y PRUEBAS DE FUNCIONAMIENTO.

3.1 IMPLEMENTACIÓN

En la implementación de control de acceso:

Se empezó por la obtención de los todos los elementos necesarios previamente utilizados en el Programa Proteus, es importante mencionar que todos estos elementos los obtuvimos en el mercado local.

En la figura 3.1 se puede observar la conexión del circuito, se realizaron las pruebas de funcionamiento en el protoboard, simulamos los sensores, la cerradura y la sirena con leds, y la comunicación serial se la realizó directamente.

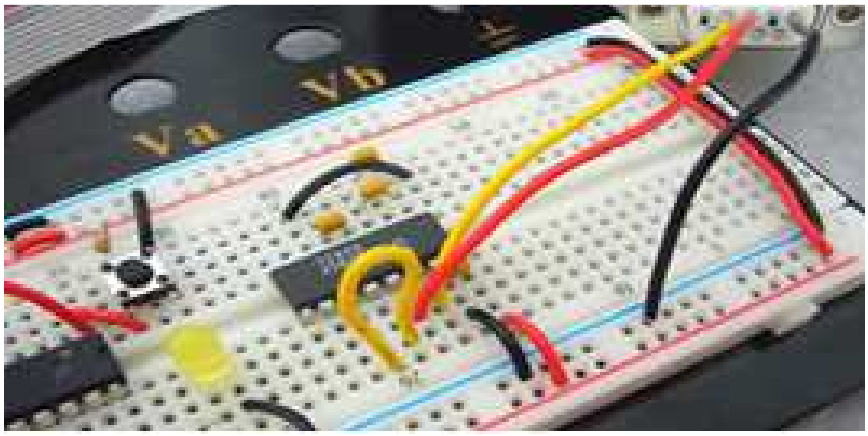


Figura 3. 1 Comunicación Serial realizada desde Protoboard

Una vez realizada las pruebas de funcionamiento en el protoboard y verificado el funcionamiento de circuito, se realizo el formato de las pistas en el programa Eagle, se imprimió las pistas en acetato y se las adhirió a la placa, como se muestra en la figura 3.2

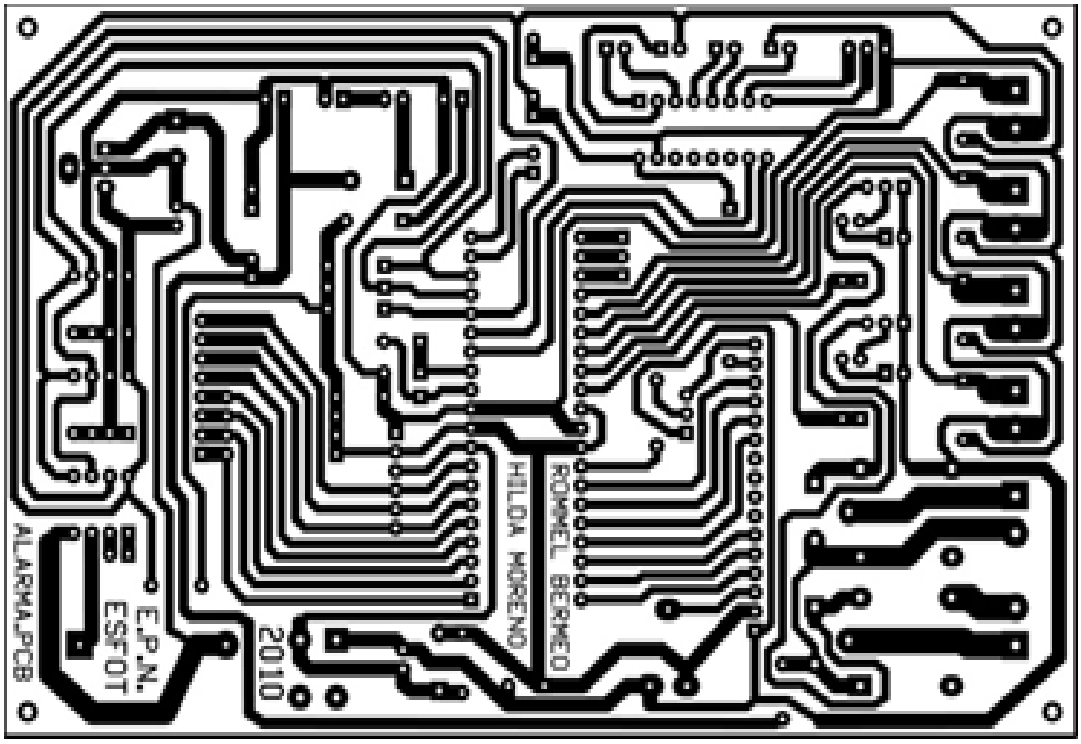


Figura 3. 2 Acetato de las pistas

Se quemó la baquelita siguiendo los pasos para realizar este proceso, siempre preocupándose de todos los detalles, tips para la obtención de buenos resultados.

Finalizado este proceso de la baquelita, se procedió a soldar todos los dispositivos que conforman el circuito del control de acceso, el AVR y otros elementos no se los soldó directamente se utilizo sócalos para protegerlos de posibles daños al momento de soldar, verificamos continuidad en las pistas y que no existan sueldas frías. Para soldar se utilizo un cautín adecuado y de bajo vatiaje para no dañar las pistas.

Una vez verificado todos estos parámetros verificamos el funcionamiento del la placa comprobando voltajes y corrientes.

Figura 3. 3 Elementos Soldados en la placa

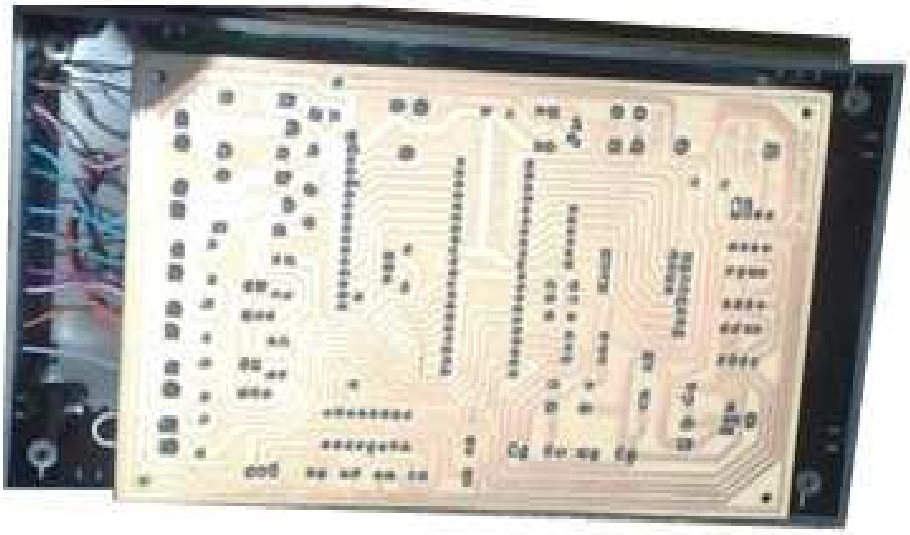


Figura 3. 4 Parte posterior de la placa.

Una vez verificado todo el funcionamiento del circuito, se la ensambló en una caja y se la empotro en la parte frontal de la entrada para mejorar la presentación del proyecto, como se muestra en la figura 3.4

En la implementación de la construcción del prototipo se realizo un diseño básico de la entrada, en la cual se puede observar una puerta y dos ventanas construidas de madera, en la puerta realizamos la instalación de la cerradura eléctrica siguiendo las instrucciones del manual, se instaló a un costado la sirena y en la parte posterior de la puerta se coloco un pulsador que tiene por función abrir la puerta desde el interior.

Figura 3. 5 Instalación de la Caja en la entrada.

En las dos ventanas y en la puerta se instaló los sensores magnéticos, como se muestra en la figura 3.6 y que tienen por función detectar el ingreso y enviar señales eléctricas al AVR, el estado de estos sensores es normalmente cerrado.

Figura 3.6 Instalación de los sensores magnéticos

Una vez finalizado la construcción del circuito, se verificó el funcionamiento realizando las pruebas correspondientes.

3.2 PRUEBAS DE FUNCIONAMIENTO

3.2.1 SIMULACIÓN DEL CIRCUITO

Se utilizó el Proteus versión 7.6, este es un software que nos permite emular el circuito, previamente el programa ya debe estar incluido en la memoria del AVR Atmega164p para realizar las pruebas y verificar el funcionamiento, no se puede emular todo el funcionamiento del circuito como mencionamos anteriormente.

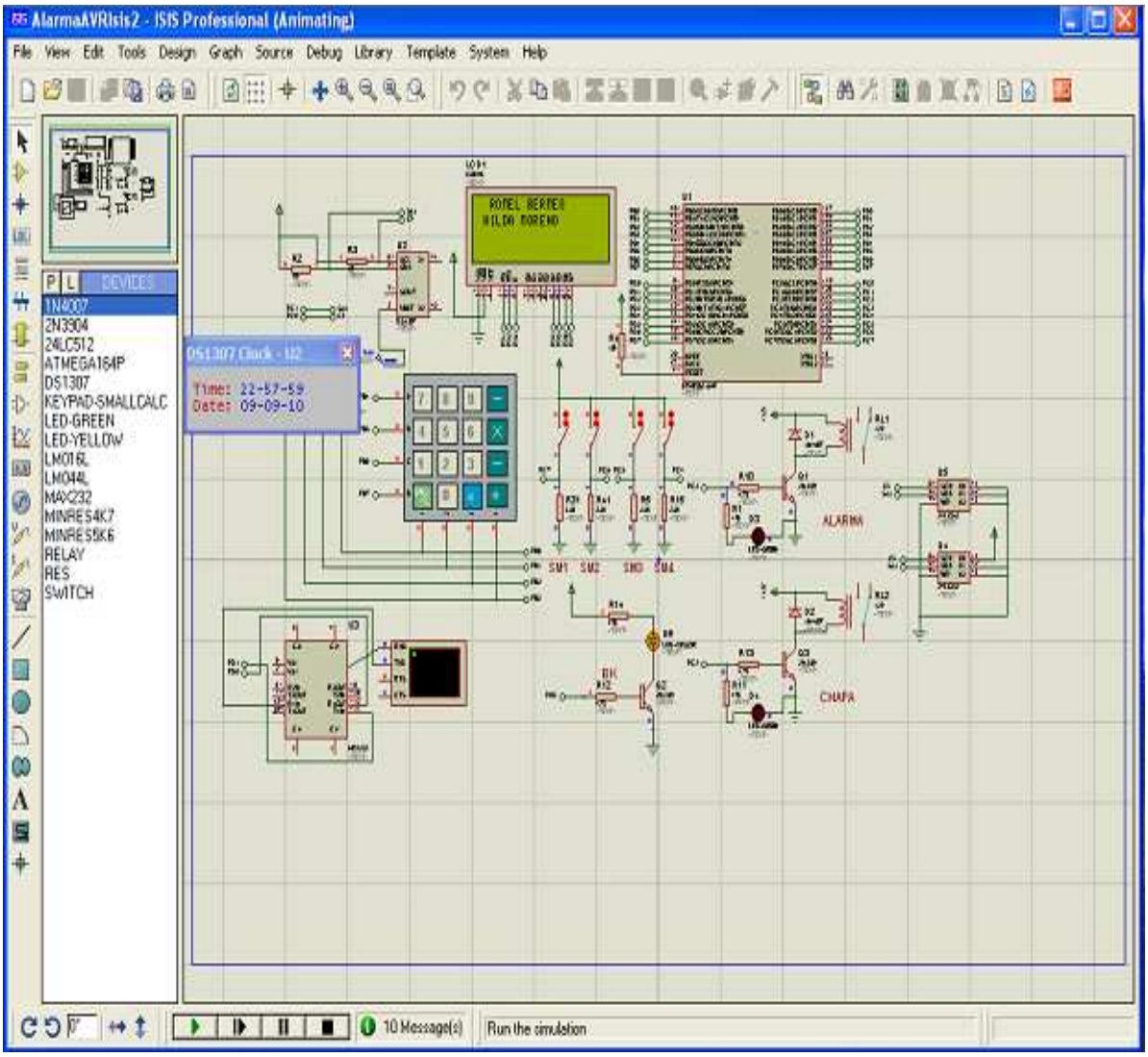


Figura 3.7 Simulación en Proteus del Circuito

Las pruebas finales del circuito ya implementado las realizaremos en vivo, con el funcionamiento del circuito.

3.2.2 PRUEBAS DE CIRCUITO EN VIVO

Conectamos la fuente de alimentación del circuito, activamos el interruptor e inmediatamente se visualiza en la pantalla LDC.

ESC. POLITECNICA NACIONAL, ESFOT, ELECTRÓNICA Y TELECOMUNICACIONES, ACCESO DE PERSONAL, REALIZADO POR: ROMMEL BERMEO e HILDA MORENO, inicializamos el circuito presionando la tecla A. Especificado en un mensaje que se muestra en la Pantalla LCD “Para comenzar elija la tecla A” como nos muestra la figura 3.8

Figura 3.8 Pantalla de Inicio del Control

Luego se visualiza un mensaje “INGRESE LA CLAVE” figura 3.9, esta clave está programada para que presente características como: sea de cuatro dígitos, solo se puedes ingresar números decimales, no permite ingresar más de cuatro dígitos, y al ingresar la clave se visualiza el símbolo (*), se podrá borrar el o los dígitos erróneos con la tecla “*”, para aceptar seleccionamos la tecla “#”.

Figura 3. 9 Solicitud de Ingreso de clave de usuario

Si no se ha ingresado correctamente la clave en el display se mostrará “CLAVE INCORRECTA”.

Caso contrario si hemos introducido correctamente los caracteres de la clave registrados en la memoria, la pantalla nos presenta la bienvenida al usuario mostrando en la pantalla el nombre del usurario, y este podrá ingresar sin ningún inconveniente, inmediatamente se abre la cerradura eléctrica.

Realizamos otra prueba introduciendo intencionalmente una clave incorrecta tres veces y se muestra en el display “CLAVE INCORRECTA”, después de un instante nos aparece “ALARMA ACTIVADA” y el dispositivo sonoro se acciona de inmediato.

El dispositivo sonoro se podrá desactivar presionando la tecla D, esto solo deben conocer la o las personas autorizadas para controlar el sistema de control de acceso, en este caso el administrador, para reactivar la alarma presionamos la tecla C.

Realizamos una prueba por intrusión, en este caso se abrió una a una las ventanas, y también la puerta utilizando la llave, si se intenta ingresar de esta

forma, se acciona la sirena inmediatamente, la misma dejará de sonar si se desactiva el sistema, el administrador será la persona que conozca como desactivar el dispositivo sonoro.

Probamos la comunicación serial con el computador para ello abrimos el archivo ejecutable creado en Visual Basic, y nos muestra la siguiente ventana:

Figura 3. 10 Ventana de inicio de sesión del Programa

Para ejecutar el programa en el PC se introduce la clave en este caso la clave es “asistencia”, teniendo la opción de cambio de clave en la pestaña que se encuentra en la parte inferior de la ventana “Inicio de sesión” figura 3.10, en la cual podemos modificar siguiendo los pasos como se muestra en la figura 3.11.

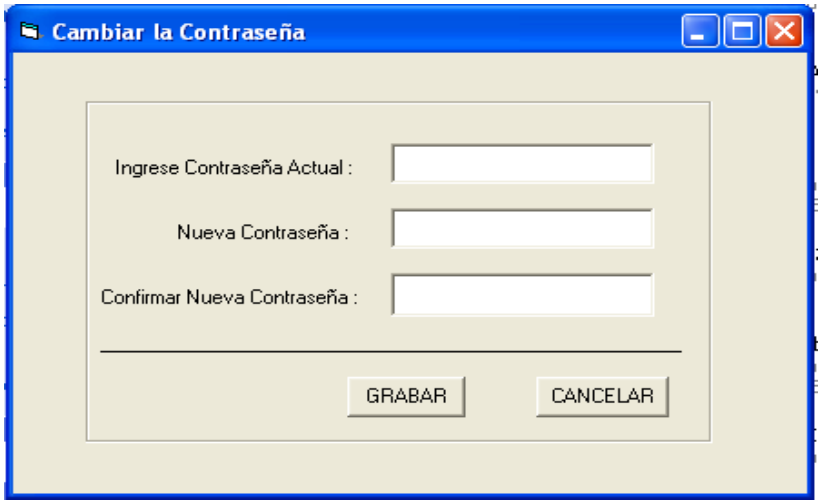


Figura 3. 11 Pantalla de cambio de clave

Una vez digitada la clave se muestra una ventana de bienvenida, e inmediatamente se despliega una ventana figura 3.12 ya con el desarrollo de las aplicaciones del control de acceso.

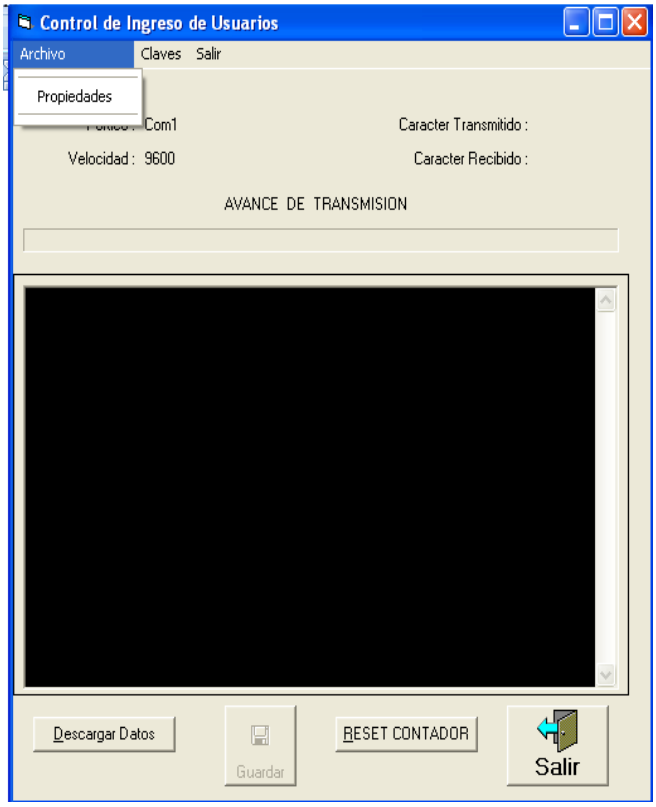


Figura 3. 12 Ventana de trabajo

En la parte superior tenemos tres pestañas: archivo, claves y salir, en la parte inferior, tenemos cuatro botones: Descargar datos, Guardar, reset contador y salir.

En el menú archivo, se configurara el puerto de comunicación, en nuestro caso en nuestra máquina contábamos con un solo puerto de comunicación, por lo que se eligió el Com 1 que vemos en la figura 3.13, puerto al cual conectaremos los conectores RS232.

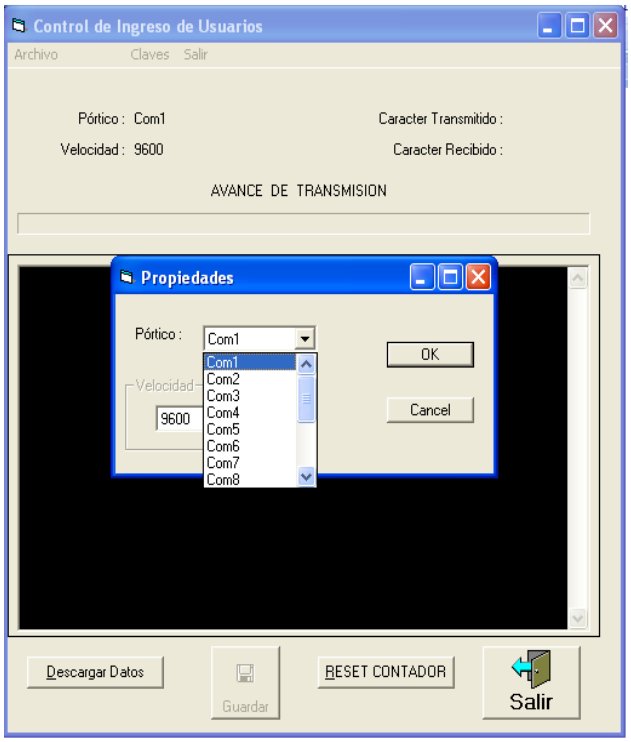


Figura 3. 13 Configuración para el puerto de comunicación

Desde el menú "Claves" desde un archivo creado por el programa Visual Basic podemos modificar los nombres de usuarios, enrolar a nuevos e incluso eliminarlos dependiendo de la necesidad del administrador, la información de los nombres con sus respectivas claves ya modificado desde esta ventana figura 3.14 haciendo clic en "Enviar Usuarios" son enviadas a las memorias externas, que tienen la funcionalidad de alojar estos datos. Para el envío de esta información el circuito debe estar fuera del menú ingrese clave, para ello debemos ayudarnos de la tecla A del teclado.

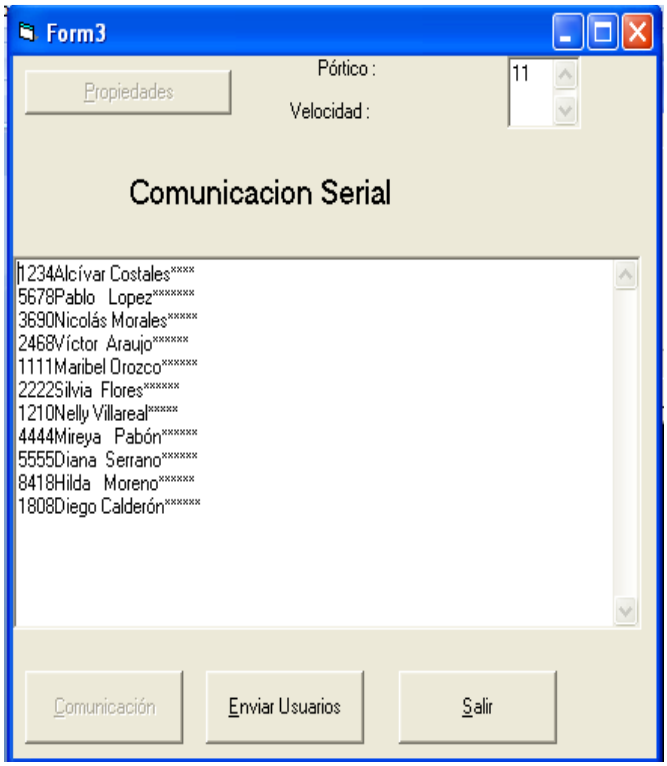


Figura 3.14 Introducción de claves

La pestaña “Salir” permite salir de la ventana, cerrando por completo el programa.

Para descargar el registro de ingreso de claves, se ubica nuevamente en la ventana principal del programa y se da clic en la pestaña “Descargar Datos”, una vez descargados se muestra una pequeña ventana DESCARGA OK seleccionamos “Aceptar” figura 3.15

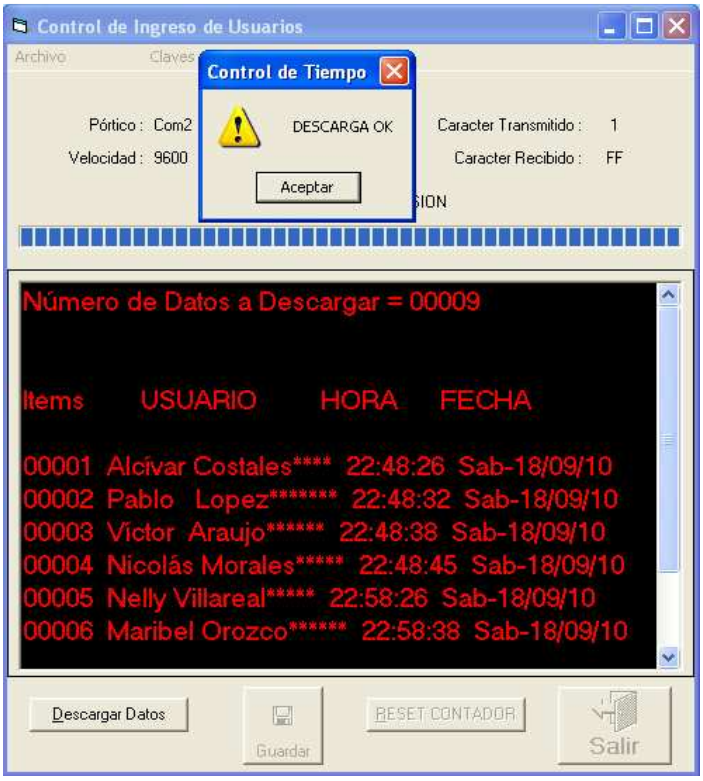


Figura 3. 15 Descarga de datos

Una vez descargados los datos del registro de entradas en la ventana, creada por el programa visual Basic se puede guardar normalmente como cualquier archivo de texto.

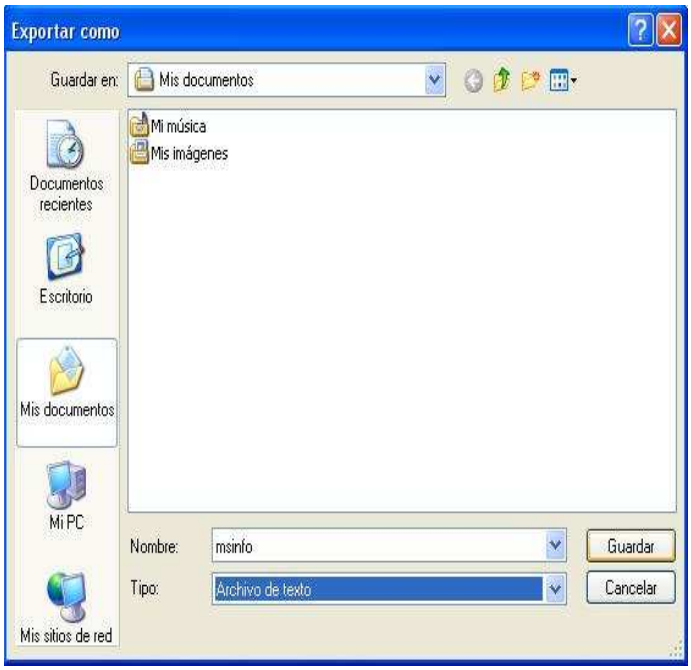


Figura 3. 16 Archivo del texto

Con este archivo de texto el administrador puede hacer un control de los usuarios que han accedido al sistema, pues nos indica el usuario, la hora y la fecha que accedió al sistema.

3.3 ANALISIS TÉCNICO Y ECONÓMICO

3.3.1 ANÁLISIS TÉCNICO

Este prototipo consta de una caja que contiene el circuito electrónico y una cerradura eléctrica marca Viro, el circuito electrónico es alimentado por una fuente de alimentación continua de 12 V, 3A, la misma que es conectada directo a la alimentación de la empresa eléctrica (105 V a 125 V) de corriente alterna, la cerradura eléctrica tiene su propio trasformador de corriente alterna de 120 a 12V y también es alimentada por la empresa eléctrica.

La caja que contiene el circuito electrónico es plástica y resistente, lo que permite que pueda ser empotrada en cualquier puerta o pared, siendo la pantalla LCD y el teclado los únicos elementos que quedan al alcance del usuario.

Los elementos sensibles a sobre voltajes son el AVR y las memorias, son elementos que pueden ser remplazados fácilmente ya que están conectados a zócalos y además se implementó un circuito programador para evitar en lo posible la manipulación directamente de estos.

La cerradura eléctrica es de fácil instalación, esta puede adaptarse a cualquier puerta metálica o de madera, la instalación eléctrica se la realizó con cable gemelo 14 AWG.

3.3.2 ANÁLISIS ECONÓMICO

3.3.2.1 PROGRAMAS UTILIZADOS EN EL PROYECTO

Se utilizó versiones gratuitas de los programas.

DESCRIPCIÓN	P. UNITARIO	P.TOTAL
Cd Proteus 7.6	3.00	3.00
Cd Visual Basic	3.00	3.00
Cd Bascom AVR	3.00	3.00
	TOTAL($)	9.00

3.3.2.2 ELEMENTOS PARA ELABORAR EL CIRCUITO ELÉCTRONICO

Se utilizó elementos que pueden adquirirse en cualquier tienda de elementos de electrónica.

CANTIDAD	DESCRIPCIÓN	P. UNITARIO	P.TOTAL
1	Circuito Integrado Atmega 164p	7.10	7.10
1	Circuito Integrado DS 1307	2.30	2.30
1	Batería para reloj de 3V	0.55	0.55
3	Sensores Magnéticos	2.50	7.50
1	Cerradura eléctrica	78.00	78.00
1	LCD 20x4	10.47	10.47
1	Teclado matricial 4x4	3.50	3.50
1	Sirena	10.00	10.00
6	Capacitores 10uf	0.05	0.30
1	Capacitores 1000uf	0.05	0.05
1	Capacitor 104 puf	0.05	0.05
11	Resistencias 3,9K	0.02	0.22
4	Resistencias 10K	0.02	0.08
3	Resistencias 820	0.02	0.06
1	Resistencias 100k	0.02	0.02
3	Zócalos	0.30	0.30
2	Espadines(24)	0.50	1.00
2	Relés	0.75	1.50
2	Memorias 24LC512	3.50	7.00
1	Cristal de cuarzo 8000Mhz	0.50	0.50
1	Regulador de voltaje 7705	0.40	0.40
1	Disipador	0.65	0.65
1	Circuito Integrado ICSP	2.80	2.80
1	Cable serial RS 232	0.45	0.45
1	Circuito Integrado MAX 232	1.30	1.30
2	Cable UTP (1m)	0.38	0.76
1	Pulsador para la cerradura	0.50	0.50
1	Pulsador (reset)	0.25	0.25
3	Leds	0.10	0.30
1	Interruptor unitario	0.50	0.50
1	Plug hembra	0.15	0.15
1	Fuente de voltaje de 12V	15.00	15.00
1	Baquelita	10.00	10.00
2	Funditas de acido	1.00	2.00
2	Brocas pequeñas	0.50	1.00
		SUB TOTAL	166.56
		IVA 12%	19.99
		TOTAL ($)	186.55

3.3.2.3 ESTRUCTURA DEL PROTOTIPO

El prototipo de la construcción del control de acceso, fue elaborado en una carpintería, se adquirió una caja plástica y se le realizaron cortes y adaptaciones para que contenga de forma adecuada el circuito electrónico.

CANTIDAD	DESCRIPCIÓN	P. UNITARIO	P.TOTAL
1	Estructura de Madera	80.00	80.00
1	Caja para el circuito	25.00	25.00
1	Broca de madera	0.80	0.80
1	Barra de Silicón	0.50	0.50
1	Pega Súper Bonder	1.60	1.60
		TOTAL ($)	107.90

3.3.2.4 GASTOS DE PAPELERIA Y OFICINA

Aquí detallamos los gastos para la elaboración del documento y otros gastos de papelería durante el proyecto.

CANTIDAD	DESCRIPCIÓN	P. UNITARIO	P.TOTAL
1	Impresiones	35.00	35.00
2	Acetatos	0.25	0.50
1	Utilización de Internet	10.00	10.00
1	Fotocopias	5.00	5.00
1	Cd	1.00	1.00
		TOTAL ($)	51.50

3.3.2.5 COSTO TOTAL

A los gastos anteriormente descritos le añadimos 25 dólares como mano de obra para poner en funcionamiento al circuito.

DESCRIPCIÓN	COSTO
Programas	9.00
Material Electrónico	186.55
Estructura del prototipo	107.90
Papelería y Oficina	51.50
Mano de obra	25.00
TOTAL ($)	379.95

El estudio del mercado se encontraron equipos para acceso de personal con precios que van desde los 300 a 549 dólares, los precios varían de acuerdo a la capacidad de los equipos, varios de los cuales encontramos publicados en paginas como mercado libre, esto nos hace pensar que este sistema podría competir con los precios, además este proyecto ofrece un sistema de seguridad el mismo que es un beneficio extra para potenciales clientes.

Los costos se pueden abaratar porque hoy en día la utilización de estos sistemas ha aumentado en empresas, edificios, organizaciones, etc.

Los elementos que utilizamos son sumamente accesibles en el mercado, variando un poco su precio de una tienda electrónica a otra, son elementos electrónicos que se encuentran sin dificultad en Ecuador.

CONCLUSIONES

- Los micros AVR tienen un menor consumo de potencia.
- La capacidad de almacenamiento es un parámetro muy importante al momento de escoger un microcontrolador, de no poder almacenar podemos escoger medios alternativos como son las memorias EEPROM.
- La implementación del ICSP dentro del circuito eléctrico nos facilitó el trabajo con el AVR sin tener que manipularle demasiado, permitiéndonos realizar cambios instantáneos.
- Bascom AVR fue una herramienta útil que nos permitió trabajar con facilidad en la programación de los AVRs.
- Con este prototipo se puede brindar seguridad y protección a cualquier organización, edificios o viviendas, dando seguridad para los mismos.
- Cada vez los microcontroladores van disminuyendo su tamaño y creciendo la capacidad de funcionamiento para distintas aplicaciones como es el caso de los AVR.
- Los precios en el mercado de un sistema de control de acceso, varía dependiendo de las características del equipo y la funcionalidad del mismo.

RECOMENDACIONES

- Se recomienda en el trasporte tener mucho cuidado con la pantalla LCD pues un elemento muy sensible.
- Se recomienda utilizar este proyecto en cualquier organización tanto para seguridad como para control de personal.
- Se recomienda tomar en cuenta la velocidad del microcontrolador en la programación puesto que la información entregada por los sensores puede tomar algo de tiempo, especialmente en lo que es las simulaciones.
- Se recomienda la implementación de la memoria externas para seguridad de los datos y evitar la pérdida de información.
- Se recomienda para la selección de memorias tomar en cuenta la capacidad de éstas, y los datos a procesar, de ser necesario en nuestro proyecto las memorias se las puede reemplazar por unas de mayor capacidad, para ello, se tendría que hacer pequeñas correcciones en la programación.
- Se recomienda para evitar demoras en la trasmisión de datos hacia el computador, darle un mantenimiento preventivo, limpieza de piezas, desinfección de virus y si el caso lo requiere formateo del mismo.
- Se recomienda el uso de los sensores de acuerdo a las necesidades de la organización, sin que esto afecte al proyecto, ya que solo necesitamos la señal emitida por éstos.
- Se recomienda que se proteja adecuadamente lo que es el circuito y sólo se deje al usuario lo que es el teclado. Y el uso del programa en Basic lo haga solo un usuario administrador capacitado.

BIBLIOGRAFÍA

- http://losmicrocontroladores.blogspot.com/
- http://www.monografias.com/trabajos12/microco/microco.shtml
- http://www.monografias.com/trabajos12/microco/microco.shtml#DIFER
- http://es.wikipedia.org/wiki/Atmel
- http://www.alegsa.com.ar/Dic/avr.php
- http://www.clubse.com.ar/DIEGO/NOTAS/2/nota18.htm
- http://www.scribd.com/doc/7828553/Capitulo1-Introduccion-del-ATmega32-espanol
- http://www.worldlingo.com/ma/enwiki/es/Atmel_AVR
- http://www.lulu.com/items/volume_38/588000/588200/1/print/SESION_1_ATMEGA8.pdf
- http://es.wikipedia.org/wiki/AVR
- http://www.webelectronica.com.ar/news32/nota06.htm
- http://www.caveo.com.ar/lcd_20x4_sinc.htm
- http://elbaroncojo.stream18.com/microcontroladores/
- http://www.clubse.com.ar/download/pdf/notasrevistas08/nota01.htm
- http://www.iearobotics.com/proyectos/cuadernos/ct4/ct4.html
- http://www.iearobotics.com/proyectos/cuadernos/ct1/ct1.html
- http://axxon.com.ar/rob/Comunicacion_max232.htm
- http://www.foxitsoftware.com
- http://es.wikipedia.org/wiki/Rel%C3%A9
- http://server-die.alc.upv.es/asignaturas/PAEEES/2004-05/A01-A04%20-%20Memoria%2024LC512%20con%20Interfaz%20I2C.pdf
- VALENCIA, RAMIRO. Aplicaciones Electrónicas con Microcontroladores Bascom. Quito-Ecuador. Rispergraf 2003. 130p.

ANEXOS

MANUAL DEL USUARIO

Este manual sirve como guía para el usuario

1. Conectamos la fuente de alimentación y la cerradura a la energía eléctrica 120V.
2. Activamos el interruptor
3. Configuración del teclado

Figura 4. 1 Teclado del circuito

3.1.1 Para inicializar presionamos la tecla A

3.1.2 Introducimos el código del usuario proporcionado por el administrador.

3.1.3 Para borrar utilizamos la tecla *

3.1.4 Para entrar utilizamos la tecla #

4. Para hacer un reset al circuito abrimos la caja y presionamos el pulsador con el indicador del reset.
5. Para cambio de uso de los usuarios y contraseñas o cualquier modificación se utilizara el programa en Basic para la computadora, el cual viene incluido para el Administrador.
6. Para igualar el reloj mantenemos presionada la tecla B

6.1 Con la tecla 3 puedes seleccionar horas minutos o segundos

6.2 Con la tecla 1 aumentar sea horas, minutos o segundos

6.3 Con la tecla 2 disminuye sea horas, minutos o segundos

6.4 Para salir digite otra vez B

7. Para la desactivación de la alarma se la realiza con la tecla D.
8. Para la reactivación de la alarma se realiza con la tecla C.
9. Para la apertura sistema desde la parte interior se debe apretar el pulsador de la cerradura eléctrica.

MANUAL DE INSTALACIÓN DE LA CERRADURA

REGULACIÓN DE LA ENTRADA Y DEL ESFUERZO DE CIERRE

La cerradura está fabricada de serie con estrada de 60mm. Si el tamaño del perfil de la puerta o del portón sobre la que se desea instalar la cerradura lo precisa es posible con una simple operación cambiar la entrada a 50-70-80mm para modificar la entrada seguiremos las siguientes instrucciones:

Destornillar los 2 dos tornillos A y quitar con un destornillador de estrella y quitar la tapa B de la cerradura.

Utilizando la llave dado en dotación desplazar el sector E hasta descubrir los tornillos 1, 2, 3, 4.

Atajar los tornillos C utilizando la llave dad en dotación 5

Levantar la pieza F y al mismo tiempo colocarla en la posición en la cual el cilindro coincida con la entrada que desea.

Instalar el agujero de la pieza F en el eje G y apretar los dos tornillos C. Para disminuir el esfuerzo del cierre (puertas ligeras) quitar el anillo elástico.

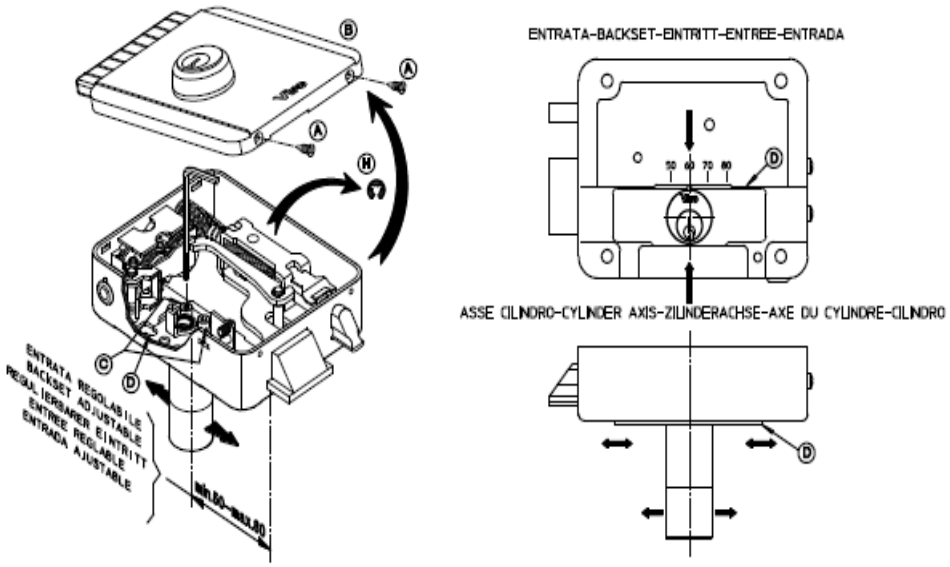


Figura 4. 2 Regulación de la entrada y del esfuerzo y del cierre

INSTALACIÓN EN LA PUERTA DE MADERA

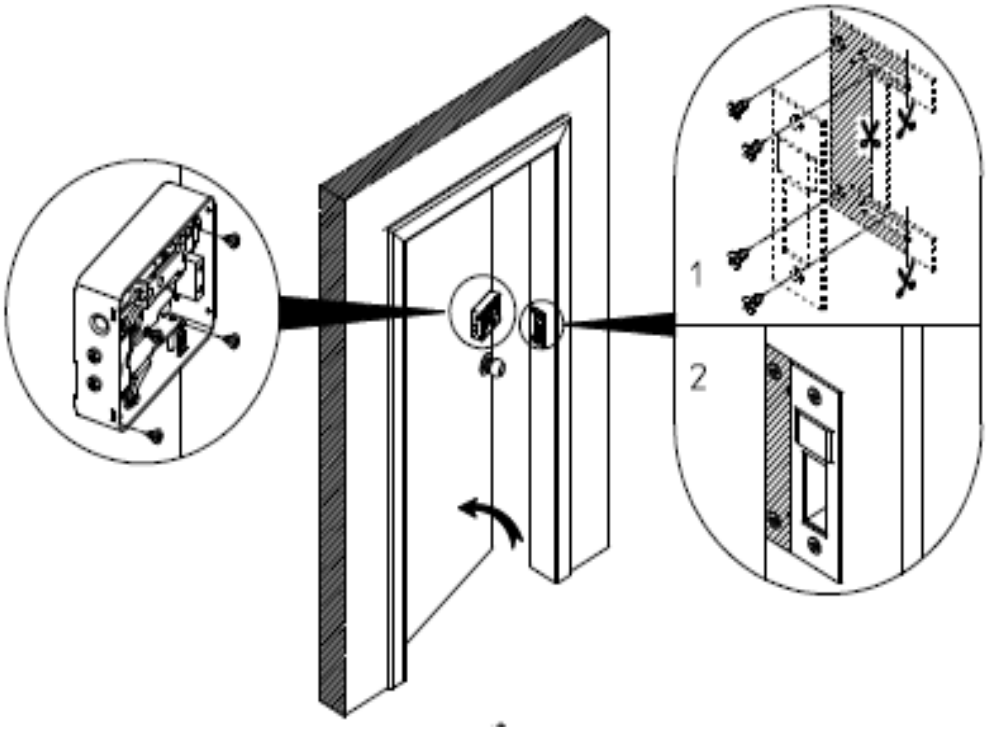


Figura 4. 3 Instalación en la puerta de madera

INSTALACIÓN ELÉCTRICA

Sujetar el cable de alimentación a la regleta de la bobina haciéndolo entrar en la zona de contacto y apretando los dos tornillos de fijación. Se recomienda realizar la instalación en conformidad con las normas nacionales en vigencia y alimentar la cerradura con un transformador conforme con las normas nacionales en vigencia de características iguales a las que figuran en la caja.

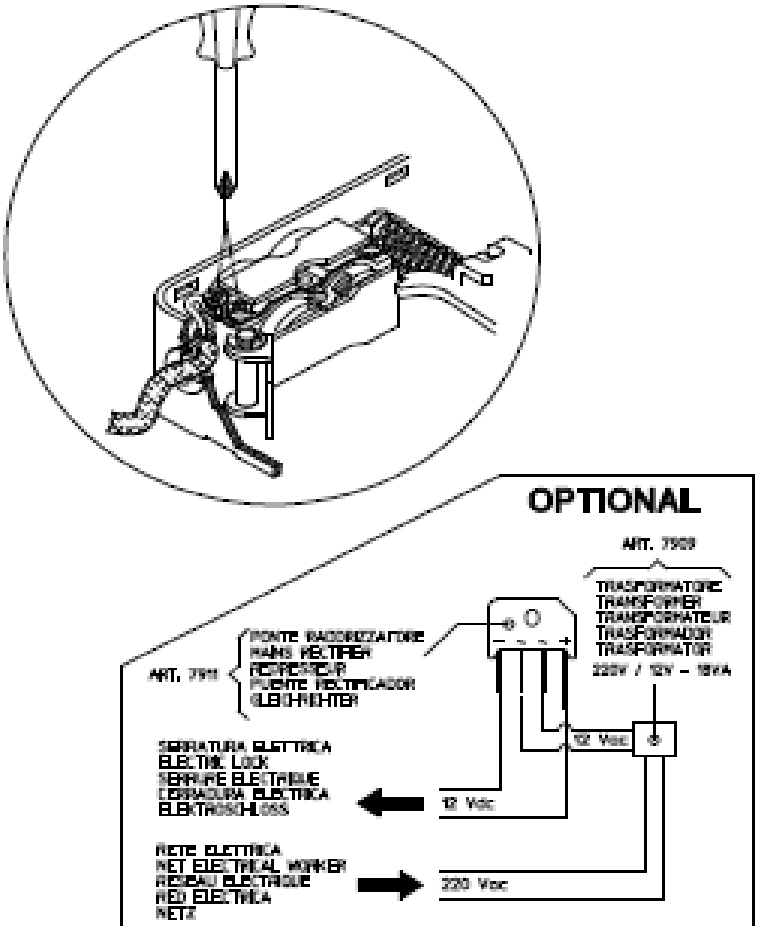
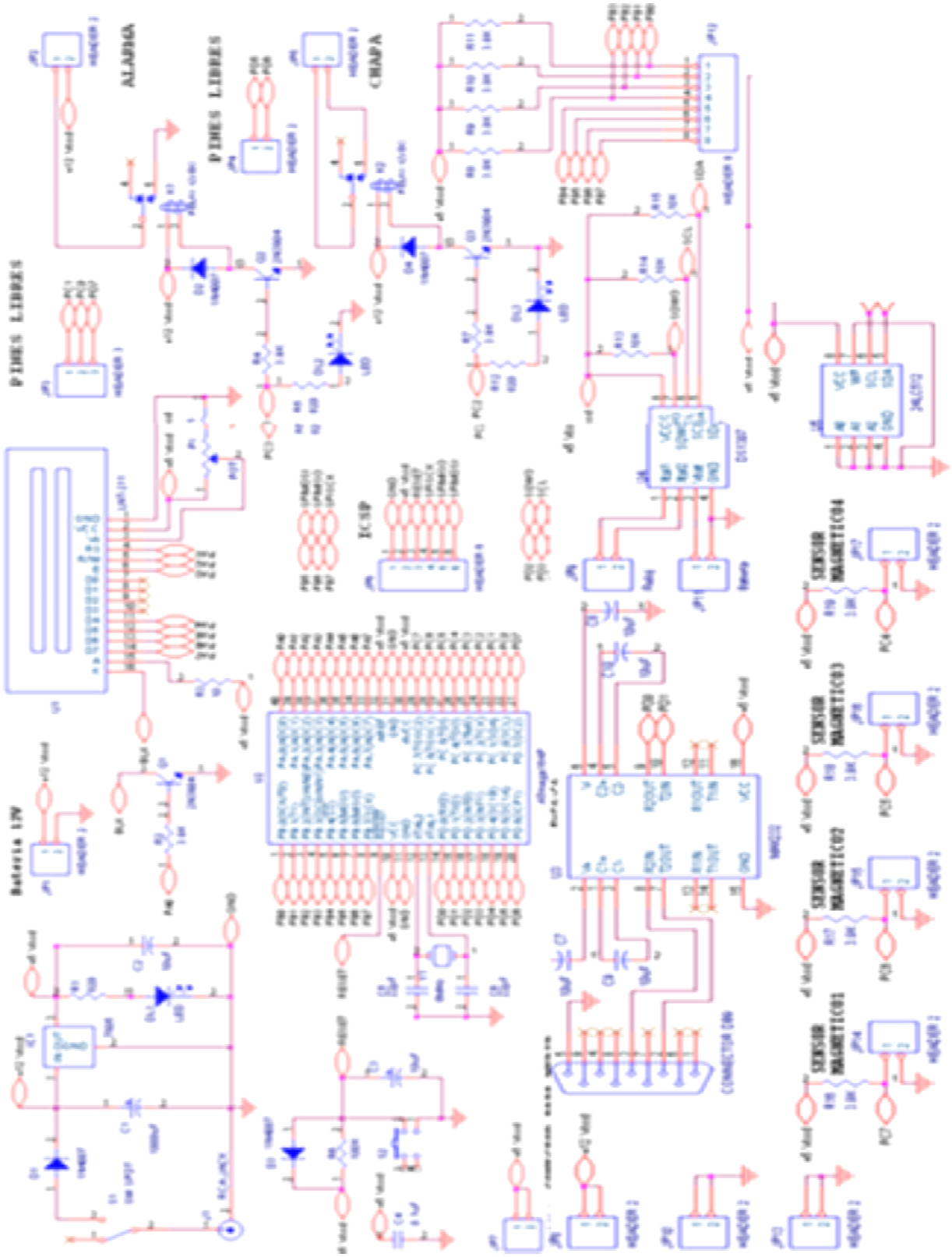
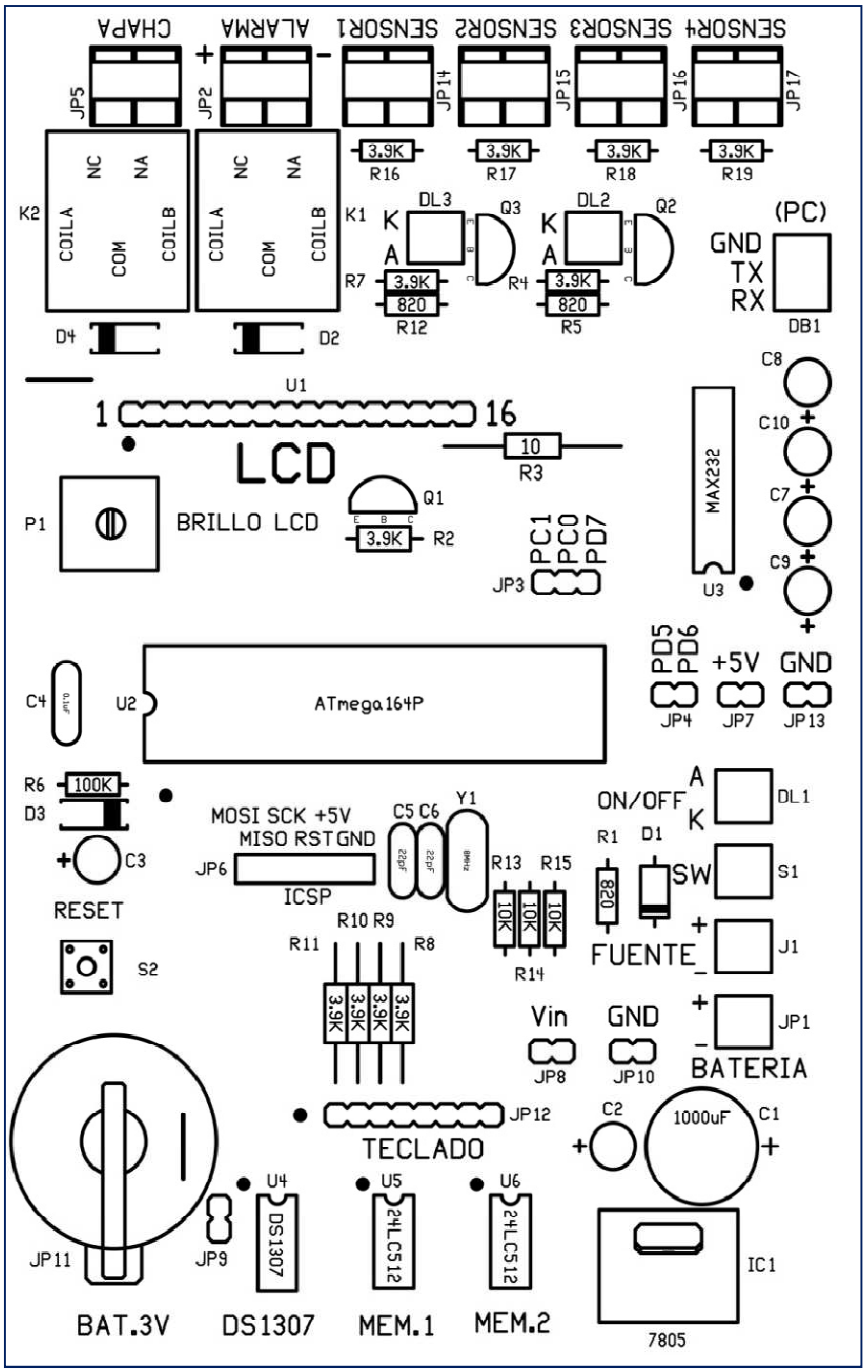


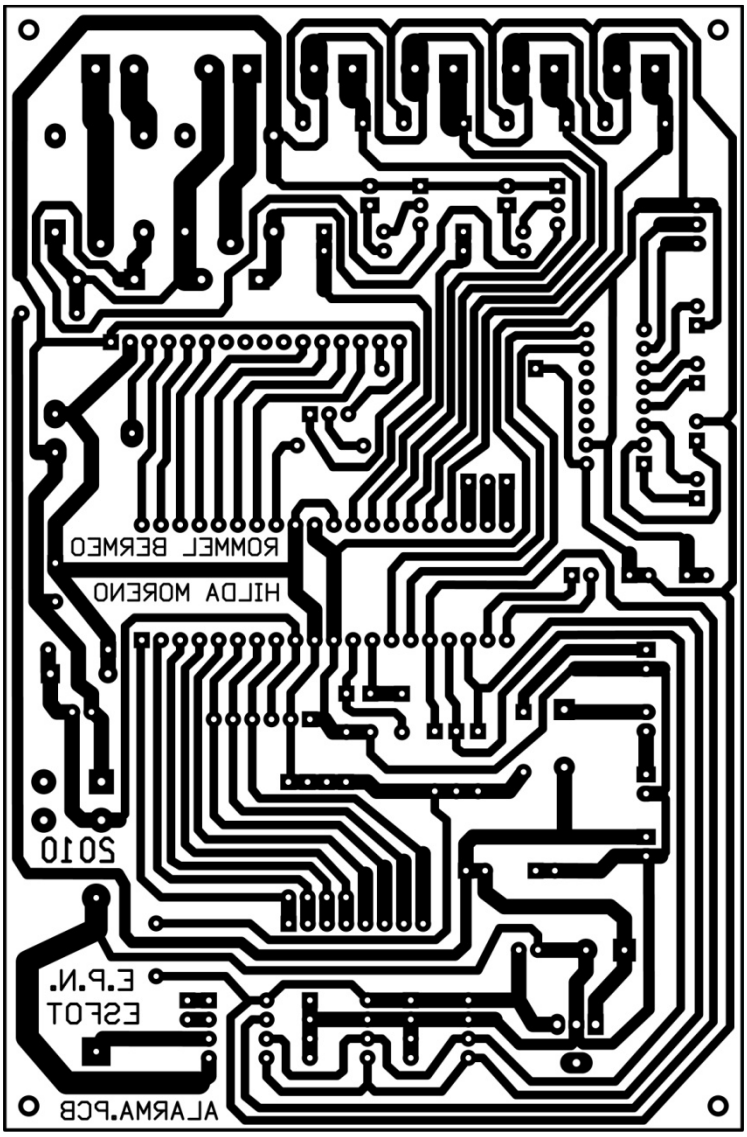
Figura 4. 4 Instalación eléctrica

ESQUEMÁTICO DEL CIRCUITO



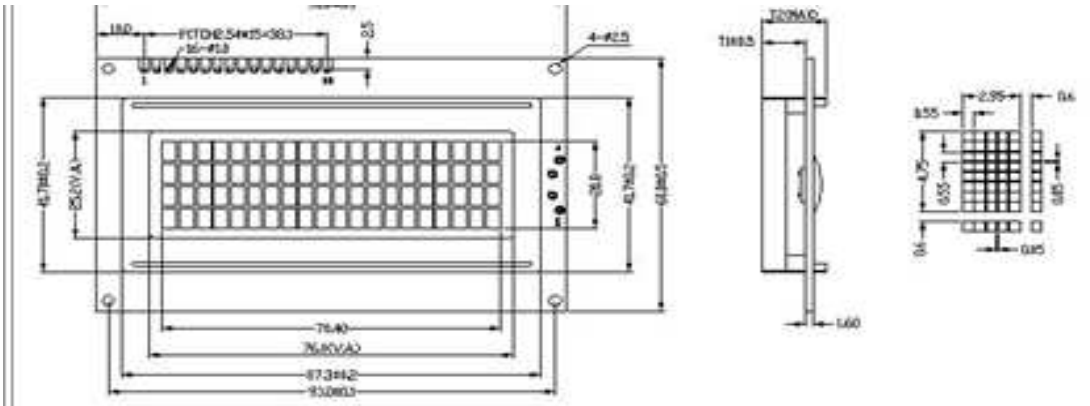
PISTAS DE CIRCUITO REALIZADAS EN EL PROGRAMA EAGLE




ROMMEL BERMEO
HILDA MORENO
2010
E.P.N.
ESFOT
ALARMA.PCB

DATASHEETS

LCD 20x4



GDM2004D

FEATURE

1. 5X8 dots with cursor
2. Built in controller(S6A0069 or equivalent)
3. +5V Power supply(Also available for+3.0V)
4. 1/16 duty cycle
5. B/L to be driven by pin1,pin2 or pin15,pin16 or A ,K
6. N.V.optional

INTERFACE PIN CONNECTIONS

PIN NO.	SYMBOL	FUNCTION
1	VSS	Power supply
2	VDD	
3	V0	Contrast adjustment
4	RS	H/L Register select signal
5	R/W	H/L Read/Write signal
6	E	H/L Enable signal
7	DB0	H/L Data bus for 4 bit or 8 bit mode
8	DB1	
9	DB2	
10	DB3	
11	DB4	
12	DB5	
13	DB6	
14	DB7	
15	LED+	+4.2V for B/L
16	LED-	Power supply for B/L (0V)

THICKNESS

VERSION	T1	T2	UNIT
EL&NO BACKLIGHT	4.6	9.5	mm
LED BACKLIGHT	9.3	14.0	

MECHANICAL DATA

ITEM	STANDARD	UNIT
Module dimension	98.0×60.0	mm
Viewing area	76.0×25.2	mm
Dot size	0.55×0.55	mm
Character size	2.95×4.75	mm

MAXIMUM RATING

ITEM	SYMBOL	STANDARD			UNIT
		MIN.	TYP.	MAX.	
Power supply	VDD-VSS	-0.3	—	7.0	V
Input voltage	VI	-0.3	—	VDD+0.3	

ELECTRICAL CHARACTERISTICS

ITEM	SYMBOL	CONDITION	STANDARD			UNIT
			MIN.	TYP.	MAX.	
Input voltage	VDD	-	4.5	5.0	5.5	V
		-	2.7	3.0	3.3	
Supply current	IDD	VDD=5.0	-	1.5	3.0	mA
Recommended LCD driving voltage for normal temp version module	VDD-V0	-20℃	-	-	-	V
		0℃	4.7	4.7	5.0	
		25℃	4.3	4.5	4.7	
		50℃	4.1	4.3	4.5	
		70℃	-	-	-	
LED B/L voltage	VF	25℃	-	4.2	4.6	
LED B/L current	IF	VF=4.2V	-	240	-	mA

DISPLAY CHARACTER ADDRESS CODE

DISPLAY POSITON	1	---	16	17	18	19	20
DDRAM address	00	—	0F	10	11	12	13
DDRAM address	40	—	4F	50	51	52	53
DDRAM address	14	—	23	24	25	26	27
DDRAM address	54	—	63	64	65	66	67

ATMEGA 164P

Features

- High-performance, Low-power AVR® 8-bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single-clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 16 MIPS Throughput at 16 MHz
  - On-chip 2-cycle Multiplier
- Nonvolatile Program and Data Memories
  - 16/32/64K Bytes of In-System Self-Programmable Flash
    Endurance: 10,000 Write/Erase Cycles
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - 512B/1K/2K Bytes EEPROM
    Endurance: 100,000 Write/Erase Cycles
  - 1/2/4K Bytes Internal SRAM
  - Programming Lock for Software Security
- JTAG (IEEE std. 1149.1 Compliant) Interface
  - Boundary-scan Capabilities According to the JTAG Standard
  - Extensive On-chip Debug Support
  - Programming of Flash, EEPROM, Fuses, and Lock Bits through the JTAG Interface
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescalers and Compare Modes
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - 8-channel, 10-bit ADC
    Differential mode with selectable gain at 1x, 10x or 200x(1)
  - Byte-oriented Two-wire Serial Interface
  - Two Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated RC Oscillator
  - External and Internal Interrupt Sources
  - Six Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, Standby and Extended Standby
- I/O and Packages
  - 32 Programmable I/O Lines
  - 44-lead TQFP, and 44-pad QFN/MLF
- Operating Voltages
  - 2.7 - 5.5V for ATmega164P/324P/644P
- Speed Grades
  - ATmega164P/324P/644P: 0 - 8MHz @ 2.7 - 5.5V, 0 - 16MHz @ 4.5 - 5.5V
- Power Consumption at 8 MHz, 5V, 25°C for ATmega644P
  - Active mode: 8 mA
  - Idle mode: 2.4 mA
  - Power-down Mode: 0.8 µA

Notes: 1. Differential Mode is not recommended above 85°C.

8-bit AVR® Microcontroller with 16/32/64K Bytes In-System Programmable Flash

ATmega164P
ATmega324P
ATmega644P

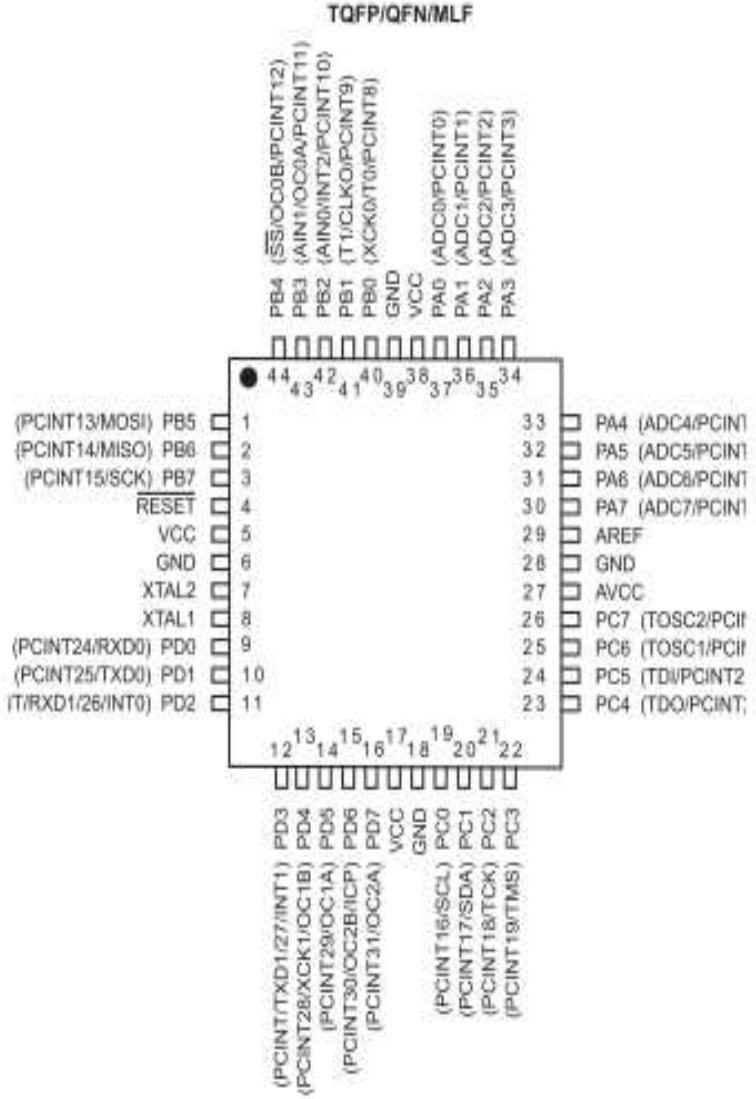
Automotive

Summary

7674DS–AVR–08/08

1. Pin Configurations

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



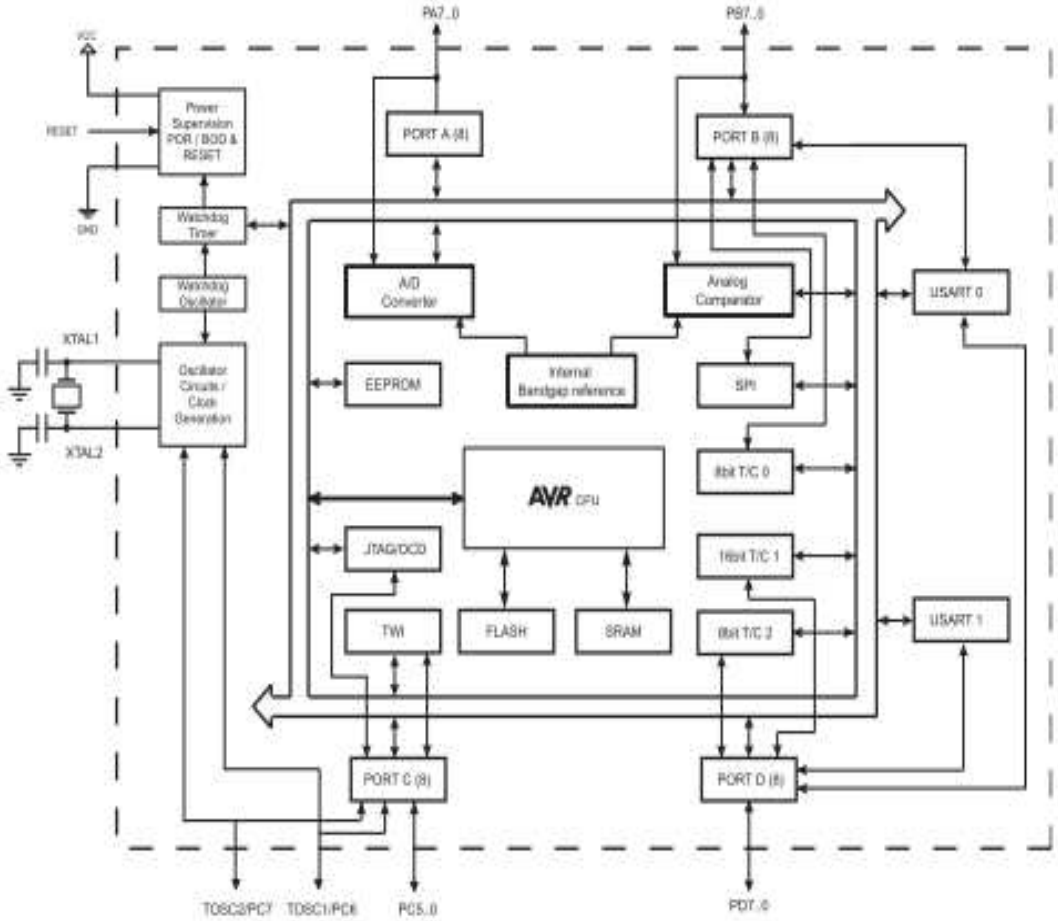
Note: The large center pad underneath the QFN/MLF package should be soldered to ground on the board to ensure good mechanical stability.

2. Overview

The ATmega164P/324P/644P is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega164P/324P/644P achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

2.1 Block Diagram

Figure 2-1. Block Diagram



The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega164P/324P/644P provides the following features: 16/32/64K bytes of In-System Programmable Flash with Read-While-Write capabilities, 512B/1K/2K bytes EEPROM, 1/2/4K 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

Automotive Quality Grade

The ATmega164P/324P/644P have been developed and manufactured according to the most stringent requirements of the international standard ISO-TS-16949. This data sheet contains limit values extracted from the results of extensive characterization (Temperature and Voltage). The quality and reliability of the ATmega164P/324P/644P have been verified during regular product qualification as per AEC-Q100 grade 1.

As indicated in the ordering information paragraph, the products are available in three different temperature grades, but with equivalent quality and reliability objectives. Different temperature identifiers have been defined as listed in Table 1.

Table 1. Temperature Grade Identification for Automotive Products

Temperature	Temperature Identifier	Comments
-40 ; +85	T	Similar to Industrial Temperature Grade but with Automotive Quality
-40 ; +105	T1	Reduced Automotive Temperature Range
-40 ; +125	Z	Full AutomotiveTemperature Range

3. Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xFF)	Reserved	-	-	-	-	-	-	-	-	
(0xFE)	Reserved	-	-	-	-	-	-	-	-	
(0xFD)	Reserved	-	-	-	-	-	-	-	-	
(0xFC)	Reserved	-	-	-	-	-	-	-	-	
(0xFB)	Reserved	-	-	-	-	-	-	-	-	
(0xFA)	Reserved	-	-	-	-	-	-	-	-	
(0xF9)	Reserved	-	-	-	-	-	-	-	-	
(0xF8)	Reserved	-	-	-	-	-	-	-	-	
(0xF7)	Reserved	-	-	-	-	-	-	-	-	
(0xF6)	Reserved	-	-	-	-	-	-	-	-	
(0xF5)	Reserved	-	-	-	-	-	-	-	-	
(0xF4)	Reserved	-	-	-	-	-	-	-	-	
(0xF3)	Reserved	-	-	-	-	-	-	-	-	
(0xF2)	Reserved	-	-	-	-	-	-	-	-	
(0xF1)	Reserved	-	-	-	-	-	-	-	-	
(0xF0)	Reserved	-	-	-	-	-	-	-	-	
(0xEF)	Reserved	-	-	-	-	-	-	-	-	
(0xEE)	Reserved	-	-	-	-	-	-	-	-	
(0xED)	Reserved	-	-	-	-	-	-	-	-	
(0xEC)	Reserved	-	-	-	-	-	-	-	-	
(0xEB)	Reserved	-	-	-	-	-	-	-	-	
(0xEA)	Reserved	-	-	-	-	-	-	-	-	
(0xE9)	Reserved	-	-	-	-	-	-	-	-	
(0xE8)	Reserved	-	-	-	-	-	-	-	-	
(0xE7)	Reserved	-	-	-	-	-	-	-	-	
(0xE6)	Reserved	-	-	-	-	-	-	-	-	
(0xE5)	Reserved	-	-	-	-	-	-	-	-	
(0xE4)	Reserved	-	-	-	-	-	-	-	-	
(0xE3)	Reserved	-	-	-	-	-	-	-	-	
(0xE2)	Reserved	-	-	-	-	-	-	-	-	
(0xE1)	Reserved	-	-	-	-	-	-	-	-	
(0xE0)	Reserved	-	-	-	-	-	-	-	-	
(0xDF)	Reserved	-	-	-	-	-	-	-	-	
(0xDE)	Reserved	-	-	-	-	-	-	-	-	
(0xDD)	Reserved	-	-	-	-	-	-	-	-	
(0xDC)	Reserved	-	-	-	-	-	-	-	-	
(0xDB)	Reserved	-	-	-	-	-	-	-	-	
(0xDA)	Reserved	-	-	-	-	-	-	-	-	
(0xD9)	Reserved	-	-	-	-	-	-	-	-	
(0xD8)	Reserved	-	-	-	-	-	-	-	-	
(0xD7)	Reserved	-	-	-	-	-	-	-	-	
(0xD6)	Reserved	-	-	-	-	-	-	-	-	
(0xD5)	Reserved	-	-	-	-	-	-	-	-	
(0xD4)	Reserved	-	-	-	-	-	-	-	-	
(0xD3)	Reserved	-	-	-	-	-	-	-	-	
(0xD2)	Reserved	-	-	-	-	-	-	-	-	
(0xD1)	Reserved	-	-	-	-	-	-	-	-	
(0xD0)	Reserved	-	-	-	-	-	-	-	-	
(0xCF)	Reserved	-	-	-	-	-	-	-	-	
(0xCE)	UDR1	USART1 I/O Data Register								188
(0xCD)	UBRR1H	-	-	-	-	USART1 Baud Rate Register High Byte				192/205
(0xCC)	UBRR1L	USART1 Baud Rate Register Low Byte								192/205
(0xCB)	Reserved	-	-	-	-	-	-	-	-	
(0xCA)	UCSR1C	UMSEL11	UMSEL10	UPM11	UPM10	USBS1	UCSZ11	UCSZ10	UCPOL1	190/204
(0xC9)	UCSR1B	RXCIE1	TXCIE1	UDRIE1	RXEN1	TXEN1	UCSZ12	RXB81	TXB81	189/203
(0xC8)	UCSR1A	RXC1	TXC1	UDRE1	FE1	DOR1	UPE1	U2X1	MPCM1	188/203
(0xC7)	Reserved	-	-	-	-	-	-	-	-	
(0xC6)	UDR0	USART0 I/O Data Register								188
(0xC5)	UBRR0H	-	-	-	-	USART0 Baud Rate Register High Byte				192/205
(0xC4)	UBRR0L	USART0 Baud Rate Register Low Byte								192/205
(0xC3)	Reserved	-	-	-	-	-	-	-	-	
(0xC2)	UCSR0C	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01	UCSZ00	UCPOL0	190/204
(0xC1)	UCSR0B	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80	189/203
(0xC0)	UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0	188/203

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xBF)	Reserved	-	-	-	-	-	-	-	-	
(0xBE)	Reserved	-	-	-	-	-	-	-	-	
(0xBD)	TWAMR	TWAM6	TWAM5	TWAM4	TWAM3	TWAM2	TWAM1	TWAM0	-	234
(0xBC)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	-	TWIE	231
(0xBB)	TWDR	2-wire Serial Interface Data Register								233
(0xBA)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	234
(0xB9)	TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	-	TWPS1	TWPS0	233
(0xB8)	TWBR	2-wire Serial Interface Bit Rate Register								231
(0xB7)	Reserved	-	-	-	-	-	-	-	-	
(0xB6)	ASSR	-	EXCLK	AS2	TCN2UB	OCR2AUB	OCR2BUB	TCR2AUB	TCR2BUB	157
(0xB5)	Reserved	-	-	-	-	-	-	-	-	
(0xB4)	OCR2B	Timer/Counter2 Output Compare Register B								157
(0xB3)	OCR2A	Timer/Counter2 Output Compare Register A								157
(0xB2)	TCNT2	Timer/Counter2 (8 Bit)								156
(0xB1)	TCCR2B	FOC2A	FOC2B	-	-	WGM22	CS22	CS21	CS20	155
(0xB0)	TCCR2A	COM2A1	COM2A0	COM2B1	COM2B0	-	-	WGM21	WGM20	152
(0xAF)	Reserved	-	-	-	-	-	-	-	-	
(0xAE)	Reserved	-	-	-	-	-	-	-	-	
(0xAD)	Reserved	-	-	-	-	-	-	-	-	
(0xAC)	Reserved	-	-	-	-	-	-	-	-	
(0xAB)	Reserved	-	-	-	-	-	-	-	-	
(0xAA)	Reserved	-	-	-	-	-	-	-	-	
(0xA9)	Reserved	-	-	-	-	-	-	-	-	
(0xA8)	Reserved	-	-	-	-	-	-	-	-	
(0xA7)	Reserved	-	-	-	-	-	-	-	-	
(0xA6)	Reserved	-	-	-	-	-	-	-	-	
(0xA5)	Reserved	-	-	-	-	-	-	-	-	
(0xA4)	Reserved	-	-	-	-	-	-	-	-	
(0xA3)	Reserved	-	-	-	-	-	-	-	-	
(0xA2)	Reserved	-	-	-	-	-	-	-	-	
(0xA1)	Reserved	-	-	-	-	-	-	-	-	
(0xA0)	Reserved	-	-	-	-	-	-	-	-	
(0x9F)	Reserved	-	-	-	-	-	-	-	-	
(0x9E)	Reserved	-	-	-	-	-	-	-	-	
(0x9D)	Reserved	-	-	-	-	-	-	-	-	
(0x9C)	Reserved	-	-	-	-	-	-	-	-	
(0x9B)	Reserved	-	-	-	-	-	-	-	-	
(0x9A)	Reserved	-	-	-	-	-	-	-	-	
(0x99)	Reserved	-	-	-	-	-	-	-	-	
(0x98)	Reserved	-	-	-	-	-	-	-	-	
(0x97)	Reserved	-	-	-	-	-	-	-	-	
(0x96)	Reserved	-	-	-	-	-	-	-	-	
(0x95)	Reserved	-	-	-	-	-	-	-	-	
(0x94)	Reserved	-	-	-	-	-	-	-	-	
(0x93)	Reserved	-	-	-	-	-	-	-	-	
(0x92)	Reserved	-	-	-	-	-	-	-	-	
(0x91)	Reserved	-	-	-	-	-	-	-	-	
(0x90)	Reserved	-	-	-	-	-	-	-	-	
(0x8F)	Reserved	-	-	-	-	-	-	-	-	
(0x8E)	Reserved	-	-	-	-	-	-	-	-	
(0x8D)	Reserved	-	-	-	-	-	-	-	-	
(0x8C)	Reserved	-	-	-	-	-	-	-	-	
(0x8B)	OCR1BH	Timer/Counter1 - Output Compare Register B High Byte								136
(0x8A)	OCR1BL	Timer/Counter1 - Output Compare Register B Low Byte								136
(0x89)	OCR1AH	Timer/Counter1 - Output Compare Register A High Byte								136
(0x88)	OCR1AL	Timer/Counter1 - Output Compare Register A Low Byte								136
(0x87)	ICR1H	Timer/Counter1 - Input Capture Register High Byte								137
(0x86)	ICR1L	Timer/Counter1 - Input Capture Register Low Byte								137
(0x85)	TCNT1H	Timer/Counter1 - Counter Register High Byte								136
(0x84)	TCNT1L	Timer/Counter1 - Counter Register Low Byte								136
(0x83)	Reserved	-	-	-	-	-	-	-	-	
(0x82)	TCCR1C	FOC1A	FOC1B	-	-	-	-	-	-	135
(0x81)	TCCR1B	ICNC1	ICES1	-	WGM13	WGM12	CS12	CS11	CS10	134
(0x80)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	-	-	WGM11	WGM10	132
(0x7F)	DIDR1	-	-	-	-	-	-	AIN1D	AIN0D	238
(0x7E)	DIDR0	ADC7D	ADC6D	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	258

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0x7D)	Reserved	-	-	-	-	-	-	-	-	
(0x7C)	ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	254
(0x7B)	ADCSRB	-	ACME	-	-	-	ADTS2	ADTS1	ADTS0	237
(0x7A)	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	256
(0x79)	ADCH	ADC Data Register High byte								257
(0x78)	ADCL	ADC Data Register Low byte								257
(0x77)	Reserved	-	-	-	-	-	-	-	-	
(0x76)	Reserved	-	-	-	-	-	-	-	-	
(0x75)	Reserved	-	-	-	-	-	-	-	-	
(0x74)	Reserved	-	-	-	-	-	-	-	-	
(0x73)	PCMSK3	PCINT31	PCINT30	PCINT29	PCINT28	PCINT27	PCINT26	PCINT25	PCINT24	70
(0x72)	Reserved	-	-	-	-	-	-	-	-	
(0x71)	Reserved	-	-	-	-	-	-	-	-	
(0x70)	TIMSK2	-	-	-	-	-	OCIE2B	OCIE2A	TOIE2	156
(0x6F)	TIMSK1	-	-	ICIE1	-	-	OCIE1B	OCIE1A	TOIE1	137
(0x6E)	TIMSK0	-	-	-	-	-	OCIE0B	OCIE0A	TOIE0	109
(0x6D)	PCMSK2	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	70
(0x6C)	PCMSK1	PCINT15	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	70
(0x6B)	PCMSK0	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	71
(0x6A)	Reserved	-	-	-	-	-	-	-	-	
(0x69)	EICRA	-	-	ISC21	ISC20	ISC11	ISC10	ISC01	ISC00	67
(0x68)	PCICR	-	-	-	-	PCIE3	PCIE2	PCIE1	PCIE0	69
(0x67)	Reserved	-	-	-	-	-	-	-	-	
(0x66)	OSCCAL	Oscillator Calibration Register								40
(0x65)	Reserved	-	-	-	-	-	-	-	-	
(0x64)	PRR	PRTWI	PRTIM2	PRTIM0	PRUSART1	PRTIM1	PRSPI	PRUSART0	PRADC	48
(0x63)	Reserved	-	-	-	-	-	-	-	-	
(0x62)	Reserved	-	-	-	-	-	-	-	-	
(0x61)	CLKPR	CLKPCE	-	-	-	CLKPS3	CLKPS2	CLKPS1	CLKPS0	40
(0x60)	WDTCSR	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	60
0x3F (0x5F)	SREG	I	T	H	S	V	N	Z	C	10
0x3E (0x5E)	SPH	SP15	SP14	SP13	SP12	SP11	SP10	SP9	SP8	11
0x3D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	11
0x3C (0x5C)	Reserved	-	-	-	-	-	-	-	-	
0x3B (0x5B)	Reserved	-	-	-	-	-	-	-	-	
0x3A (0x5A)	Reserved	-	-	-	-	-	-	-	-	
0x39 (0x59)	Reserved	-	-	-	-	-	-	-	-	
0x38 (0x58)	Reserved	-	-	-	-	-	-	-	-	
0x37 (0x57)	SPMCSR	SPMIE	RWWSB	SIGRD	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	290
0x36 (0x56)	Reserved	-	-	-	-	-	-	-	-	
0x35 (0x55)	MCUCR	JTD	BODS	BODSE	PUD	-	-	IVSEL	IVCE	51/274
0x34 (0x54)	MCUSR	-	-	-	JTRF	WDRF	BORF	EXTRF	PORF	59/274
0x33 (0x53)	SMCR	-	-	-	-	SM2	SM1	SM0	SE	47
0x32 (0x52)	Reserved	-	-	-	-	-	-	-	-	
0x31 (0x51)	OCDR	On-Chip Debug Register								264
0x30 (0x50)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	256
0x2F (0x4F)	Reserved	-	-	-	-	-	-	-	-	
0x2E (0x4E)	SPDR	SPI 0 Data Register								169
0x2D (0x4D)	SPSR	SPIF0	WCOL0	-	-	-	-	-	SPI2X0	168
0x2C (0x4C)	SPCR	SPIE0	SPE0	DORD0	MSTR0	CPOL0	CPHA0	SPR01	SPR00	167
0x2B (0x4B)	GPIOR2	General Purpose I/O Register 2								28
0x2A (0x4A)	GPIOR1	General Purpose I/O Register 1								28
0x29 (0x49)	Reserved	-	-	-	-	-	-	-	-	
0x28 (0x48)	OCR0B	Timer/Counter0 Output Compare Register B								109
0x27 (0x47)	OCR0A	Timer/Counter0 Output Compare Register A								109
0x26 (0x46)	TCNT0	Timer/Counter0 (8 Bit)								109
0x25 (0x45)	TCCR0B	FOC0A	FOC0B	-	-	WGM02	CS02	CS01	CS00	107
0x24 (0x44)	TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	-	-	WGM01	WGM00	109
0x23 (0x43)	GTCCR	TSM	-	-	-	-	-	PSR2	PSR10/310	159
0x22 (0x42)	EEARH	-	-	-	-	EEPROM Address Register High Byte				23
0x21 (0x41)	EEARL	EEPROM Address Register Low Byte								23
0x20 (0x40)	EEDR	EEPROM Data Register								23
0x1F (0x3F)	EECR	-	-	EEPM1	EEPM0	EERIE	EEMWE	EEWE	EERE	23
0x1E (0x3E)	GPIOR0	General Purpose I/O Register 0								28
0x1D (0x3D)	EIMSK	-	-	-	-	-	INT2	INT1	INT0	68
0x1C (0x3C)	EIFR	-	-	-	-	-	INTF2	INTF1	INTF0	68

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x1B (0x3B)	PCIFR	-	-	-	-	PCIF3	PCIF2	PCIF1	PCIF0	69
0x1A (0x3A)	Reserved	-	-	-	-	-	-	-	-	
0x19 (0x39)	Reserved	-	-	-	-	-	-	-	-	
0x18 (0x38)	Reserved	-	-	-	-	-	-	-	-	
0x17 (0x37)	TIFR2	-	-	-	-	-	OCF2B	OCF2A	TOV2	109
0x16 (0x36)	TIFR1	-	-	ICF1	-	-	OCF1B	OCF1A	TOV1	138
0x15 (0x35)	TIFR0	-	-	-	-	-	OCF0B	OCF0A	TOV0	109
0x14 (0x34)	Reserved	-	-	-	-	-	-	-	-	
0x13 (0x33)	Reserved	-	-	-	-	-	-	-	-	
0x12 (0x32)	Reserved	-	-	-	-	-	-	-	-	
0x11 (0x31)	Reserved	-	-	-	-	-	-	-	-	
0x10 (0x30)	Reserved	-	-	-	-	-	-	-	-	
0x0F (0x2F)	Reserved	-	-	-	-	-	-	-	-	
0x0E (0x2E)	Reserved	-	-	-	-	-	-	-	-	
0x0D (0x2D)	Reserved	-	-	-	-	-	-	-	-	
0x0C (0x2C)	Reserved	-	-	-	-	-	-	-	-	
0x0B (0x2B)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	82
0x0A (0x2A)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	82
0x09 (0x29)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	82
0x08 (0x28)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	82
0x07 (0x27)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	82
0x06 (0x26)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	82
0x05 (0x25)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	81
0x04 (0x24)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	81
0x03 (0x23)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	81
0x02 (0x22)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	81
0x01 (0x21)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	81
0x00 (0x20)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	81

Notes:
1. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
2. I/O registers within the address range $00 - $1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions.
3. Some of the status flags are cleared by writing a logical one to them. Note that the CBI and SBI instructions will operate on all bits in the I/O register, writing a one back into any flag read as set, thus clearing the flag. The CBI and SBI instructions work with registers 0x00 to 0x1F only.
4. When using the I/O specific commands IN and OUT, the I/O addresses $00 - $3F must be used. When addressing I/O registers as data space using LD and ST instructions, $20 must be added to these addresses. The ATmega164P/324P/644P is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from $60 - $FF, only the ST/STS/STD and LD/LDS/LDD instructions can be used.

4. Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl,K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	Rd ← Rd - K - C	Z,C,N,V,H	1
SBIW	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	Rd ← Rd • Rr	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	Rd ← Rd • K	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	One's Complement	Rd ← 0xFF - Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← 0x00 - Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • (0xFF - K)	Z,N,V	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd - 1	Z,N,V	1
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V	1
SER	Rd	Set Register	Rd ← 0xFF	None	1
MUL	Rd, Rr	Multiply Unsigned	R1:R0 ← Rd x Rr	Z,C	2
MULS	Rd, Rr	Multiply Signed	R1:R0 ← Rd x Rr	Z,C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	R1:R0 ← Rd x Rr	Z,C	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
FMULS	Rd, Rr	Fractional Multiply Signed	R1:R0 ← (Rd x Rr) << 1	Z,C	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
BRANCH INSTRUCTIONS					
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC ← Z	None	2
JMP	k	Direct Jump	PC ← k	None	3
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	4
ICALL		Indirect Call to (Z)	PC ← Z	None	4
CALL	k	Direct Subroutine Call	PC ← k	None	5
RET		Subroutine Return	PC ← STACK	None	5
RETI		Interrupt Return	PC ← STACK	I	5
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1/2/3
CP	Rd,Rr	Compare	Rd - Rr	Z, N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	Rd - Rr - C	Z, N,V,C,H	1
CPI	Rd,K	Compare Register with Immediate	Rd - K	Z, N,V,C,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBRS	Rr, b	Skip if Bit in Register is Set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1/2/3
SBIC	P, b	Skip if Bit in I/O Register Cleared	if (P(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBIS	P, b	Skip if Bit in I/O Register is Set	if (P(b)=1) PC ← PC + 2 or 3	None	1/2/3
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC←PC+k + 1	None	1/2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC←PC+k + 1	None	1/2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1/2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1/2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1/2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1/2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1/2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1/2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1/2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1/2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1/2
BRLT	k	Branch if Less Than Zero, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1/2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1/2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1/2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1/2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1/2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1/2

Mnemonics	Operands	Description	Operation	Flags	#Clocks
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1/2
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1/2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1/2
BIT AND BIT-TEST INSTRUCTIONS					
SBI	P,b	Set Bit in I/O Register	I/O(P,b) ← 1	None	2
CBI	P,b	Clear Bit in I/O Register	I/O(P,b) ← 0	None	2
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)← Rd(n),C←Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(3..0)←Rd(7..4),Rd(7..4)←Rd(3..0)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1
CLC		Clear Carry	C ← 0	C	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	I ← 1	I	1
CLI		Global Interrupt Disable	I ← 0	I	1
SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV		Set Twos Complement Overflow.	V ← 1	V	1
CLV		Clear Twos Complement Overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1
CLH		Clear Half Carry Flag in SREG	H ← 0	H	1
DATA TRANSFER INSTRUCTIONS					
MOV	Rd, Rr	Move Between Registers	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy Register Word	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LD	Rd, X	Load Indirect	Rd ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-Inc.	Rd ← (X), X ← X + 1	None	2
LD	Rd, - X	Load Indirect and Pre-Dec.	X ← X - 1, Rd ← (X)	None	2
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-Inc.	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, - Y	Load Indirect and Pre-Dec.	Y ← Y - 1, Rd ← (Y)	None	2
LDD	Rd,Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load Indirect and Post-Inc.	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load Indirect and Pre-Dec.	Z ← Z - 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2
LDS	Rd, k	Load Direct from SRAM	Rd ← (k)	None	2
ST	X, Rr	Store Indirect	(X) ← Rr	None	2
ST	X+, Rr	Store Indirect and Post-Inc.	(X) ← Rr, X ← X + 1	None	2
ST	- X, Rr	Store Indirect and Pre-Dec.	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store Indirect and Post-Inc.	(Y) ← Rr, Y ← Y + 1	None	2
ST	- Y, Rr	Store Indirect and Pre-Dec.	Y ← Y - 1, (Y) ← Rr	None	2
STD	Y+q,Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store Indirect and Post-Inc.	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store Indirect and Pre-Dec.	Z ← Z - 1, (Z) ← Rr	None	2
STD	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2
STS	k, Rr	Store Direct to SRAM	(k) ← Rr	None	2
LPM		Load Program Memory	R0 ← (Z)	None	3
LPM	Rd, Z	Load Program Memory	Rd ← (Z)	None	3
LPM	Rd, Z+	Load Program Memory and Post-Inc	Rd ← (Z), Z ← Z+1	None	3
SPM		Store Program Memory	(Z) ← R1:R0	None	-
IN	Rd, P	In Port	Rd ← P	None	1
OUT	P, Rr	Out Port	P ← Rr	None	1

Mnemonics	Operands	Description	Operation	Flags	#Clocks
PUSH	Rr	Push Register on Stack	STACK ← Rr	None	2
POP	Rd	Pop Register from Stack	Rd ← STACK	None	2
MCU CONTROL INSTRUCTIONS					
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep function)	None	1
WDR		Watchdog Reset	(see specific descr. for WDR/timer)	None	1
BREAK		Break	For On-chip Debug Only	None	N/A

5. Ordering Information

5.1 ATmega164P

Speed (MHz)[3]	Power Supply	Ordering Code	Package[1]	Operational Range
8-16	2.7 - 5.5V	ATmega164P-15AT[2] ATmega164P-15AT1[2] ATmega164P-15AZ[2]	ML	-40°C to +85°C -40°C to +105°C -40°C to +125°C
8-16	2.7 - 5.5V	ATmega164P-15MT[2] ATmega164P-15MT1[2] ATmega164P-15MZ[2]	PW	-40°C to +85°C -40°C to +105°C -40°C to +125°C

Notes: 1. Green and Rohs packaging
2. Tape & Reel with Dry-pack delivery
3. For Speed vs. V_{CC} see "Speed Grades" on page 326

Package Type	
ML	44-lead, Thin (1.0 mm) Plastic Gull Wing Quad Flat Package (TQFP)
PW	44-pad, 7 x 7 x 1.0 mm body, lead pitch 0.50 mm, Quad Flat No-Lead/Micro Lead Frame Package (QFN/MLF)

5.2 ATmega324P

Speed (MHz)[3]	Power Supply	Ordering Code	Package[1]	Operational Range
8-16	2.7 - 5.5V	ATmega324P-15AT[2] ATmega324P-15AT1[2] ATmega324P-15AZ[2]	ML	-40°C to +85°C -40°C to +105°C -40°C to +125°C
8-16	2.7 - 5.5V	ATmega324P-15MT[2] ATmega324P-15MT1[2] ATmega324P-15MZ[2]	PW	-40°C to +85°C -40°C to +105°C -40°C to +125°C

Notes: 1. Green and Rohs packaging
2. Tape & Reel with Dry-pack delivery
3. For Speed vs. V_{CC} see "Speed Grades" on page 326

Package Type	
ML	44-lead, Thin (1.0 mm) Plastic Gull Wing Quad Flat Package (TQFP)
PW	44-pad, 7 x 7 x 1.0 mm body, lead pitch 0.50 mm, Quad Flat No-Lead/Micro Lead Frame Package (QFN/MLF)

5.3 ATmega644P

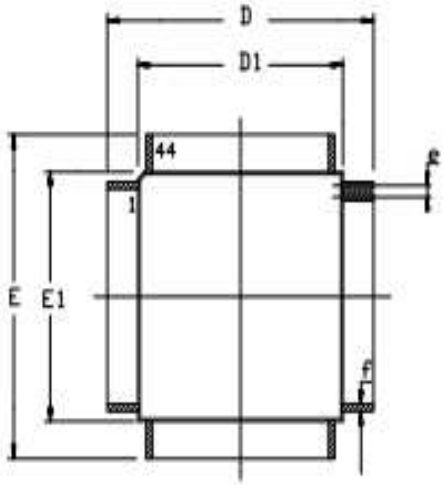
Speed (MHz)[3]	Power Supply	Ordering Code	Package[1]	Operational Range
8-16	2.7 - 5.5V	ATmega644P-15AT [2] ATmega644P-15AT1[2] ATmega644P-15AZ [2]	ML	-40°C to +85°C -40°C to +105°C -40°C to +125°C
8-16	2.7 - 5.5V	ATmega644P-15MT [2] ATmega644P-15MT1[2] ATmega644P-15MZ [2]	PW	-40°C to +85°C -40°C to +105°C -40°C to +125°C

Notes: 1. Green and Rohs packaging
2. Tape & Reel with Dry-pack delivery
3. For Speed vs. V_{CC} see "Speed Grades" on page 326.

Package Type	
ML	44-lead, Thin (1.0 mm) Plastic Gull Wing Quad Flat Package (TQFP)
PW	44-pad, 7 x 7 x 1.0 mm body, lead pitch 0.50 mm, Quad Flat No-Lead/Micro Lead Frame Package (QFN/MLF)

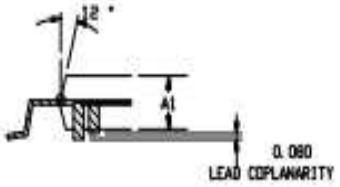
6. Packaging Information

6.1 ML



COMMON DIMENSIONS IN MM

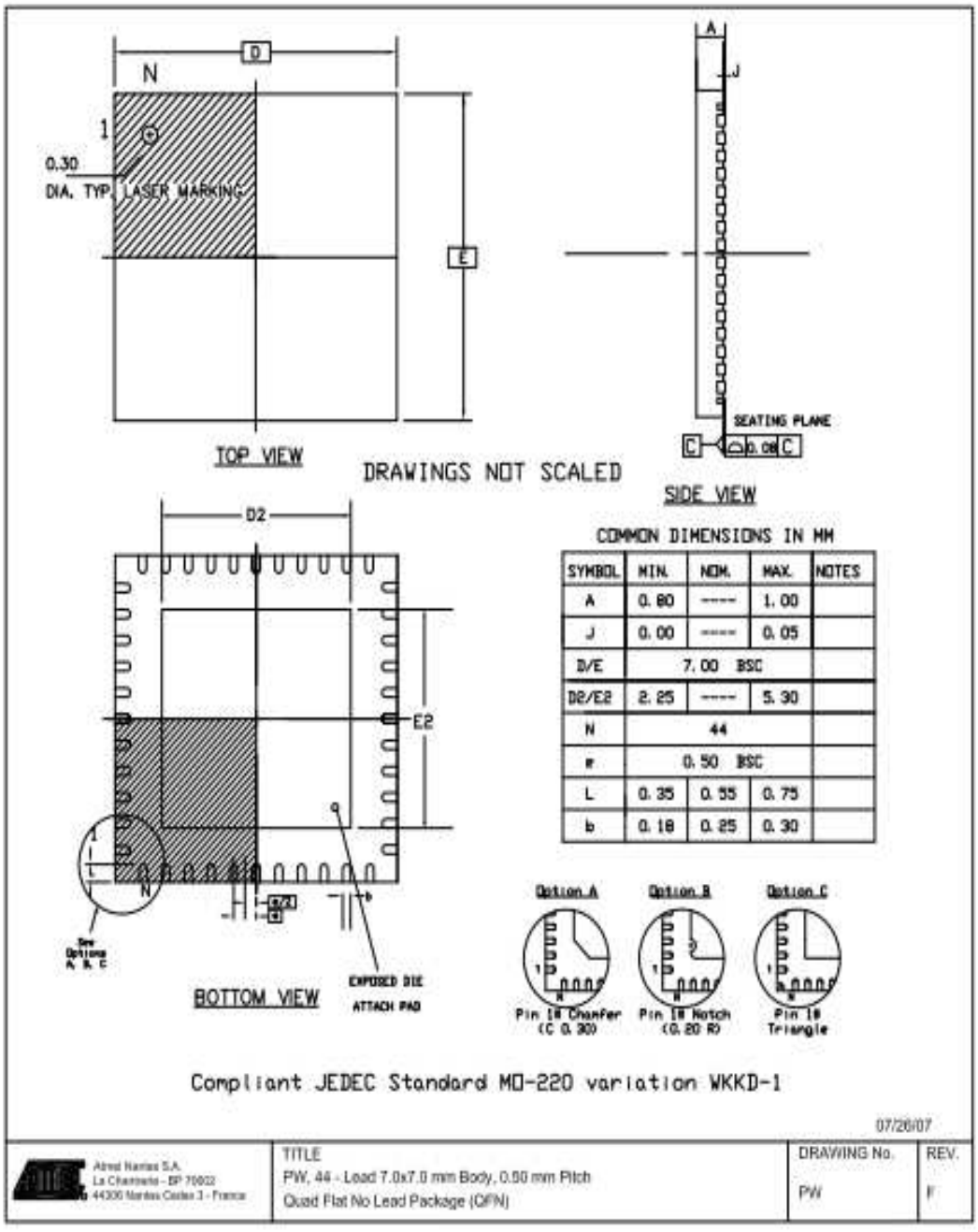
SYMBOL	Min	Max	NOTES
A	----	1.20	
A1	0.95	1.05	
C	0.09	0.20	
D	12.00 BSC		
D1	10.00 BSC		
E	12.00 BSC		
E1	10.00 BSC		
J	0.05	0.15	
L	0.45	0.75	
e	0.80 BSC		
f	0.30	0.45	



07/27/07

Atmel Nantes S.A. Le Chantenay - BP 70602 44306 Nantes Cedex 3 - France	TITLE ML, 44 - Lead, 10x10 mm Body Size, 1.0 mm Body Thickness 0.8 mm Lead Pitch, Thin Profile Plastic Quad Flat Package (TQFP)	DRAWING No. ML	REV. G

6.2 PW



COMMON DIMENSIONS IN MM

SYMBOL	MIN.	NOM.	MAX.	NOTES
A	0.80	-----	1.00	
J	0.00	-----	0.05	
D/E	7.00 BSC			
D2/E2	2.25	-----	5.30	
N	44			
e	0.50 BSC			
L	0.35	0.55	0.75	
b	0.18	0.25	0.30	

Compliant JEDEC Standard MO-220 variation WKKD-1

07/26/07

	TITLE	DRAWING No.	REV.
Atmel Nantes S.A. La Chantrerie - BP 70602 44306 Nantes Cedex 3 - France	PW, 44 - Lead 7.0x7.0 mm Body, 0.50 mm Pitch Quad Flat No Lead Package (QFN)	PW	F

NOTES: QFN STANDARD NOTES

1. DIMENSIONING & TOLERANCING CONFORM TO ASME Y14.5M. – 1994.

2. DIMENSION b APPLIES TO METALLIZED TERMINAL AND IS MEASURED BETWEEN 0.15 AND 0.30 mm FROM TERMINAL TIP. IF THE TERMINAL HAS THE OPTIONAL RADIUS ON THE OTHER END OF THE TERMINAL, THE DIMENSION b SHOULD NOT BE MEASURED IN THAT RADIUS AREA.

3. MAX. PACKAGE WARPAGE IS 0.05mm.

4. MAXIMUM ALLOWABLE BURRS IS 0.076 mm IN ALL DIRECTIONS.

5. PIN #1 ID ON TOP WILL BE LASER MARKED.

6. THIS DRAWING CONFORMES TO JEDEC REGISTERED OUTLINE MO-220.

7. A MAXIMUM 0.15mm PULL BACK (L1) MAY BE PRESENT.
L MINUS L1 TO BE EQUAL TO OR GREATER THAN 0.30 mm

8. THE TERMINAL #1 IDENTIFIER ARE OPTIONAL BUT MUST BE LOCATED WITHIN THE ZONE INDICATEI THE TERMINAL #1 IDENTIFIER BE EITHER A MOLD OR MARKED FEATURE

7. Errata

7.1 ATmega164P Rev. A

No known Errata.

7.2 ATmega324P Rev. A

No known Errata.

7.3 ATmega644P Rev. A

No known Errata.

8. Datasheet Revision History

Please note that the referring page numbers in this section are referred to this document. The referring revision in this section are referring to the document revision.

8.1 Rev. 7674C -05/08

1. VIL reset pin update. Section 26.1 on page 325.
2. Updated EEPROM endurance. See "Features" on page 1.

8.2 Rev. 7674B -01/08

1. Update to electrical characteristics after product characterization.

8.3 Rev. 7674A - 04/07

1. Initial Automotive revision
2. Insertion of specific § for automative quality references
3. DC and Frequency adapted to Automotive temperature range
4. Part numbering and package selection according to Automotive rules
5. Current Consumption adapted based on Industrial electrical characterization.

9. Datasheet Revision History

Please note that the referring page numbers in this section are referred to this document. The referring revision in this section are referring to the document revision.

9.1 Rev. 7674D -08/08

1. Added ADC differential mode electrical characteristics for ATMega164 Table 26-10 on page 334.
2. Removed Ramp Z register.

9.2 Rev. 7674C -05/08

1. VIL reset pin update. Section 26.1 on page 325.
2. Updated EEPROM endurance. See "Features" on page 1.

9.3 Rev. 7674B -01/08

1. Update to electrical characteristics after product characterization.

9.4 Rev. 7674A - 04/07

1. Initial Automotive revision
2. Insertion of specific § for automative quality references
3. DC and Frequency adapted to Automotive temperature range
4. Part numbering and package selection according to Automotive rules
5. Current Consumption adapted based on Industrial electrical characterization.

Headquarters

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

International

Atmel Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Atmel Europe
Le Krebs
8, Rue Jean-Pierre Timbaud
BP 309
78054 Saint-Quentin-en-
Yvelines Cedex
France
Tel: (33) 1-30-60-70-00
Fax: (33) 1-30-60-71-11

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Product Contact

Web Site
www.atmel.com

Technical Support
avr@atmel.com

Sales Contact
www.atmel.com/contacts

Literature Requests
www.atmel.com/literature

7674DS–AVR–08/08

24LC512

MICROCHIP **24AA512/24LC512/24FC512**

512K I²C™ CMOS Serial EEPROM

Device Selection Table

Part Number	Vcc Range	Max. Clock Frequency	Temp. Ranges
24AA512	1.8-5.5V	400 kHz(1)	I
24LC512	2.5-5.5V	400 kHz	I, E
24FC512	2.5-5.5V	1 MHz	I

Note 1: 100 kHz for Vcc < 2.5V

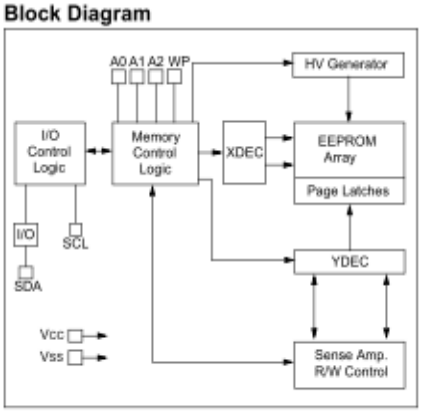
Features:

- Low-power CMOS technology:
  - Maximum write current 5 mA at 5.5V
  - Maximum read current 400 µA at 5.5V
  - Standby current 100 nA, typical at 5.5V
- 2-wire serial interface bus, I²C™ compatible
- Cascadable for up to eight devices
- Self-timed erase/write cycle
- 128-byte Page Write mode available
- 5 ms max. write cycle time
- Hardware write-protect for entire array
- Schmitt Trigger inputs for noise suppression
- 1,000,000 erase/write cycles
- Electrostatic discharge protection > 4000V
- Data retention > 200 years
- 8-pin PDIP, SOIC (208 mil), and DFN packages
- 14-lead TSSOP package
- Pb-free finishes available
- Temperature ranges:
  - Industrial (I): -40°C to +85°C
  - Automotive (E): -40°C to +125°C

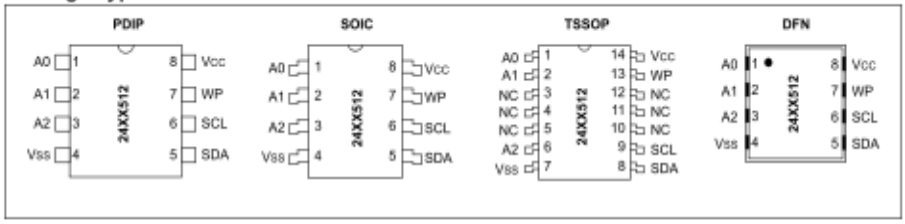
Description:

The Microchip Technology Inc. 24AA512/24LC512/24FC512 (24XX512*) is a 64K x 8 (512 Kbit) Serial Electrically Erasable PROM, capable of operation across a broad voltage range (1.8V to 5.5V). It has been developed for advanced, low-power applications such as personal communications and data acquisition. This device also has a page write capability of up to 128 bytes of data. This device is capable of both random and sequential reads up to the 512K boundary. Functional address lines allow up to eight devices on the same bus, for up to 4 Mbit address space. This device is available in the standard 8-pin plastic DIP, SOIC, DFN and 14-lead TSSOP packages.

Block Diagram



Package Type



* 24XX512 is used in this document as a generic part number for the 24AA512/24LC512/24FC512 devices.

DS 1307

DALLAS SEMICONDUCTOR

DS1307
64 x 8 Serial Real-Time Clock

www.maxim-ic.com

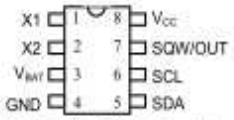
FEATURES

- 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

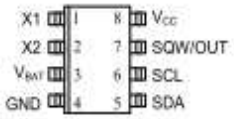
ORDERING INFORMATION

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

PIN ASSIGNMENT



DS1307 8-Pin DIP (300-mil)



DS1307 8-Pin SOIC (150-mil)

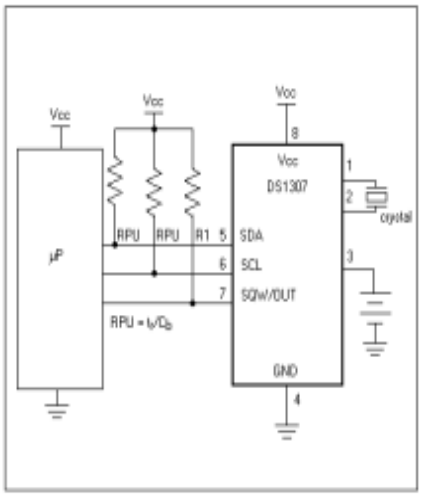
PIN DESCRIPTION

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.

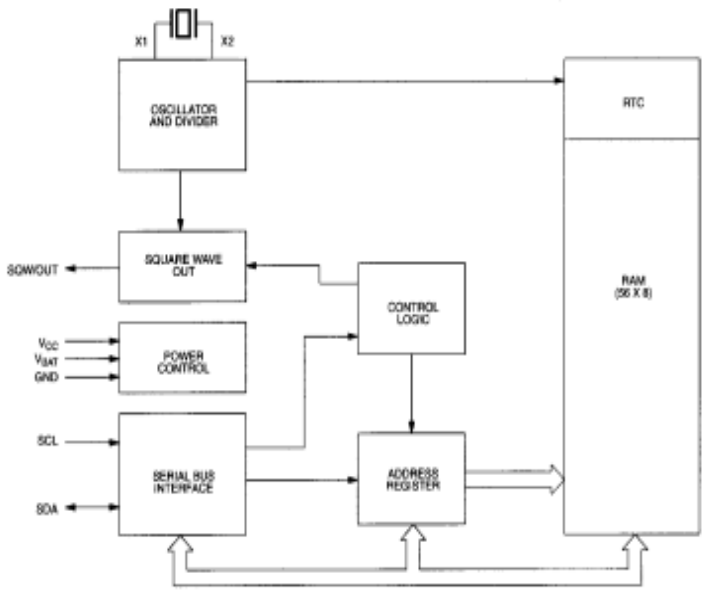
TYPICAL OPERATING CIRCUIT



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.

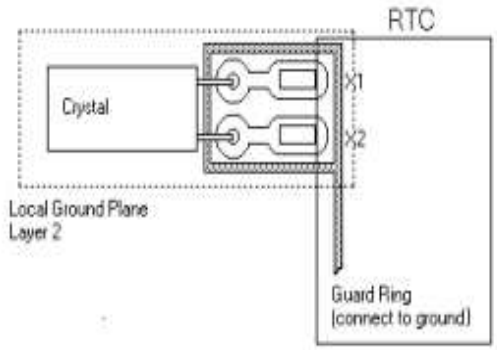
SDA (Serial Data Input/Output) – SDA is the input/output pin for the 2-wire serial interface. The SDA pin is open drain which requires an external pullup resistor.

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

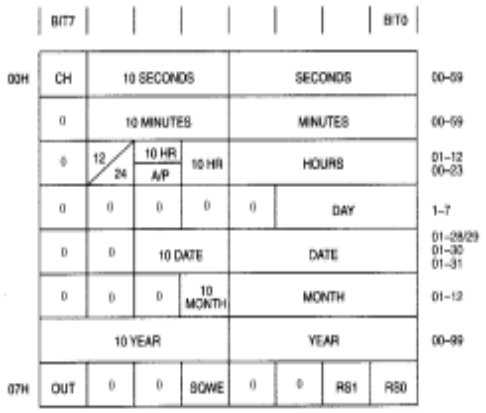
The time and calendar information is obtained by reading the appropriate register bytes. The RTC registers are illustrated in Figure 3. The time and calendar are set or initialized by writing the appropriate register bytes. The contents of the time and calendar registers are in the BCD format. Bit 7 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.

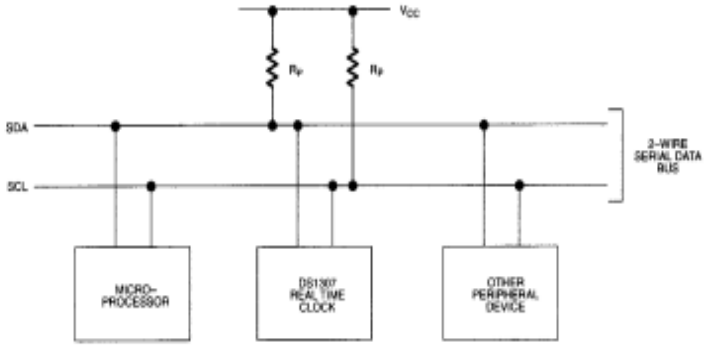
SQUAREWAVE OUTPUT FREQUENCY Table 1

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

2-WIRE SERIAL DATA BUS

The DS1307 supports a bi-directional, 2-wire bus and data transmission protocol. A device that sends data onto the bus is defined as a transmitter and a device receiving data as a receiver. The device that controls the message is called a master. The devices that are controlled by the master are referred to as slaves. The bus must be controlled by a master device that generates the serial clock (SCL), controls the bus access, and generates the START and STOP conditions. The DS1307 operates as a slave on the 2-wire bus. A typical bus configuration using this 2-wire protocol is show in Figure 4.

TYPICAL 2-WIRE BUS CONFIGURATION Figure 4



Figures 5, 6, and 7 detail how data is transferred on the 2-wire bus.

- Data transfer may be initiated only when the bus is not busy.
- During data transfer, the data line must remain stable whenever the clock line is HIGH. Changes in the data line while the clock line is high will be interpreted as control signals.

Accordingly, the following bus conditions have been defined:

Bus not busy: Both data and clock lines remain HIGH.

Start data transfer: A change in the state of the data line, from HIGH to LOW, while the clock is HIGH, defines a START condition.

Stop data transfer: A change in the state of the data line, from LOW to HIGH, while the clock line is HIGH, defines the STOP condition.

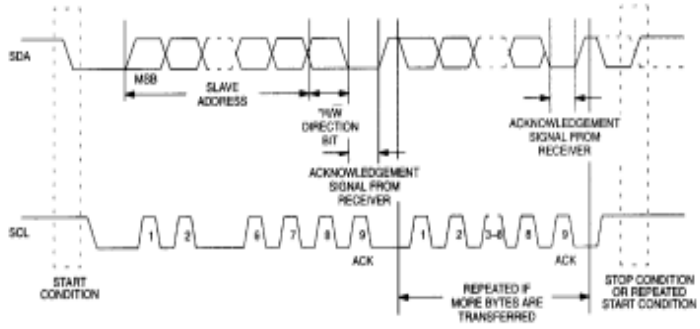
Data valid: The state of the data line represents valid data when, after a START condition, the data line is stable for the duration of the HIGH period of the clock signal. The data on the line must be changed during the LOW period of the clock signal. There is one clock pulse per bit of data.

Each data transfer is initiated with a START condition and terminated with a STOP condition. The number of data bytes transferred between START and STOP conditions is not limited, and is determined by the master device. The information is transferred byte-wise and each receiver acknowledges with a ninth bit. Within the 2-wire bus specifications a regular mode (100kHz clock rate) and a fast mode (400kHz clock rate) are defined. The DS1307 operates in the regular mode (100kHz) only.

Acknowledge: Each receiving device, when addressed, is obliged to generate an acknowledge after the reception of each byte. The master device must generate an extra clock pulse which is associated with this acknowledge bit.

A device that acknowledges must pull down the SDA line during the acknowledge clock pulse in such a way that the SDA line is stable LOW during the HIGH period of the acknowledge related clock pulse. Of course, setup and hold times must be taken into account. A master must signal an end of data to the slave by not generating an acknowledge bit on the last byte that has been clocked out of the slave. In this case, the slave must leave the data line HIGH to enable the master to generate the STOP condition.

DATA TRANSFER ON 2-WIRE SERIAL BUS Figure 5



Depending upon the state of the R/$\overline{W}$ bit, two types of data transfer are possible:

1. **Data transfer from a master transmitter to a slave receiver.** The first byte transmitted by the master is the slave address. Next follows a number of data bytes. The slave returns an acknowledge bit after each received byte. Data is transferred with the most significant bit (MSB) first.

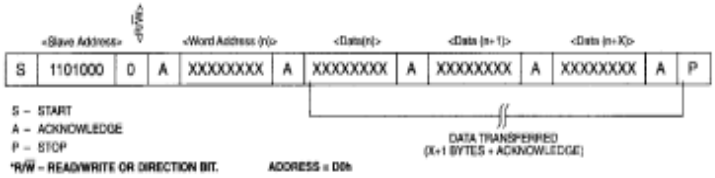
2. **Data transfer from a slave transmitter to a master receiver.** The first byte (the slave address) is transmitted by the master. The slave then returns an acknowledge bit. This is followed by the slave transmitting a number of data bytes. The master returns an acknowledge bit after all received bytes other than the last byte. At the end of the last received byte, a "not acknowledge" is returned.

The master device generates all of the serial clock pulses and the START and STOP conditions. A transfer is ended with a STOP condition or with a repeated START condition. Since a repeated START condition is also the beginning of the next serial transfer, the bus will not be released. Data is transferred with the most significant bit (MSB) first.

The DS1307 may operate in the following two modes:

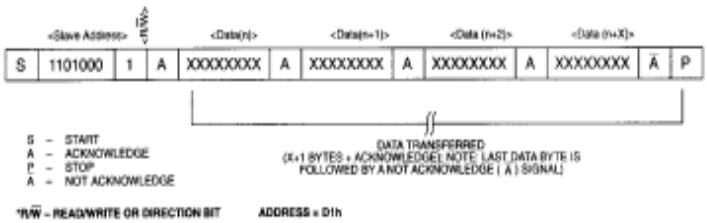
1. **Slave receiver mode (DS1307 write mode):** Serial data and clock are received through SDA and SCL. After each byte is received an acknowledge bit is transmitted. START and STOP conditions are recognized as the beginning and end of a serial transfer. Address recognition is performed by hardware after reception of the slave address and *direction bit (See Figure 6). The address byte is the first byte received after the start condition is generated by the master. The address byte contains the 7 bit DS1307 address, which is 1101000, followed by the *direction bit (R/$\overline{W}$) which, for a write, is a 0. After receiving and decoding the address byte the device outputs an acknowledge on the SDA line. After the DS1307 acknowledges the slave address + write bit, the master transmits a register address to the DS1307 This will set the register pointer on the DS1307. The master will then begin transmitting each byte of data with the DS1307 acknowledging each byte received. The master will generate a stop condition to terminate the data write.

DATA WRITE – SLAVE RECEIVER MODE Figure 6



2. **Slave transmitter mode (DS1307 read mode):** The first byte is received and handled as in the slave receiver mode. However, in this mode, the *direction bit will indicate that the transfer direction is reversed. Serial data is transmitted on SDA by the DS1307 while the serial clock is input on SCL. START and STOP conditions are recognized as the beginning and end of a serial transfer (See Figure 7). The address byte is the first byte received after the start condition is generated by the master. The address byte contains the 7-bit DS1307 address, which is 1101000, followed by the *direction bit (R/$\overline{W}$) which, for a read, is a 1. After receiving and decoding the address byte the device inputs an acknowledge on the SDA line. The DS1307 then begins to transmit data starting with the register address pointed to by the register pointer. If the register pointer is not written to before the initiation of a read mode the first address that is read is the last one stored in the register pointer. The DS1307 must receive a "not acknowledge" to end a read.

DATA READ – SLAVE TRANSMITTER MODE Figure 7



ABSOLUTE MAXIMUM RATINGS*

Voltage on Any Pin Relative to Ground	-0.5V to +7.0V
Storage Temperature	-55°C to +125°C
Soldering Temperature	260°C for 10 seconds DIP See JPC/JEDEC Standard J-STD-020A for Surface Mount Devices

* This is a stress rating only and functional operation of the device at these or any other conditions above those indicated in the operation sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods of time may affect reliability.

Range	Temperature	V_{CC}
Commercial	0°C to +70°C	4.5V to 5.5V V_{CCI}
Industrial	-40°C to +85°C	4.5V to 5.5V V_{CCI}

RECOMMENDED DC OPERATING CONDITIONS

(Over the operating range*)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Supply Voltage	V_{CC}	4.5	5.0	5.5	V	
Logic 1	V_{IH}	2.2		V_{CC} + 0.3	V	
Logic 0	V_{IL}	-0.5		+0.8	V	
V_{BAT} Battery Voltage	V_{BAT}	2.0		3.5	V	

*Unless otherwise specified.

DC ELECTRICAL CHARACTERISTICS

(Over the operating range*)

PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
Input Leakage (SCL)	I_{LI}			1	µA	
I/O Leakage (SDA & SQW/OUT)	I_{LO}			1	µA	
Logic 0 Output (I_{OL} = 5mA)	V_{OL}			0.4	V	
Active Supply Current	I_{CCA}			1.5	mA	7
Standby Current	I_{CCS}			200	µA	1
Battery Current (OSC ON); SQW/OUT OFF	I_{BAT1}		300	500	nA	2
Battery Current (OSC ON); SQW/OUT ON (32kHz)	I_{BAT2}		480	800	nA	
Power-Fail Voltage	V_{PF}	1.216 x V_{BAT}	1.25 x V_{BAT}	1.284 x V_{BAT}	V	8

*Unless otherwise specified.

AC ELECTRICAL CHARACTERISTICS

(Over the operating range*)

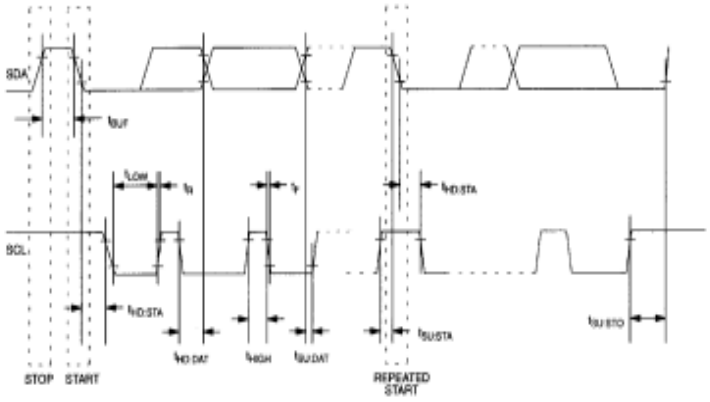
PARAMETER	SYMBOL	MIN	TYP	MAX	UNITS	NOTES
SCL Clock Frequency	f_{SCL}	0		100	kHz	
Bus Free Time Between a STOP and START Condition	t_{BUF}	4.7			µs	
Hold Time (Repeated) START Condition	$t_{HD:STA}$	4.0			µs	3
LOW Period of SCL Clock	t_{LOW}	4.7			µs	
HIGH Period of SCL Clock	t_{HIGH}	4.0			µs	
Set-up Time for a Repeated START Condition	$t_{SU:STA}$	4.7			µs	
Data Hold Time	$t_{HD:DAT}$	0			µs	4,5
Data Set-up Time	$t_{SU:DAT}$	250			ns	
Rise Time of Both SDA and SCL Signals	t_R			1000	ns	
Fall Time of Both SDA and SCL Signals	t_F			300	ns	
Set-up Time for STOP Condition	$t_{SU:STO}$	4.7			µs	
Capacitive Load for each Bus Line	C_B			400	pF	6
I/O Capacitance (T_A = 25ºC)	$C_{I/O}$		10		pF	
Crystal Specified Load Capacitance (T_A = 25ºC)			12.5		pF	

*Unless otherwise specified.

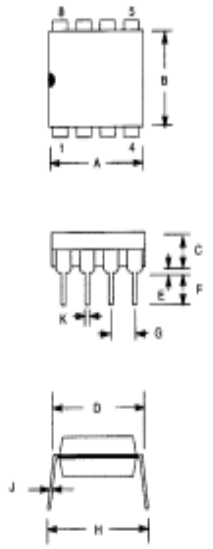
NOTES:

1. I_{CCS} specified with V_{CC} = 5.0V and SDA, SCL = 5.0V.
2. V_{CC} = 0V, V_{BAT} = 3V.
3. After this period, the first clock pulse is generated.
4. A device must internally provide a hold time of at least 300ns for the SDA signal (referred to the V_{IHMIN} of the SCL signal) in order to bridge the undefined region of the falling edge of SCL.
5. The maximum $t_{HD:DAT}$ has only to be met if the device does not stretch the LOW period (t_{LOW}) of the SCL signal.
6. C_B – Total capacitance of one bus line in pF.
7. I_{CCA} – SCL clocking at max frequency = 100kHz.
8. V_{PF} measured at V_{BAT} = 3.0V.

TIMING DIAGRAM Figure 8

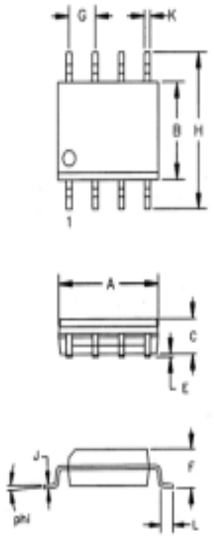


DS1307 64 X 8 SERIAL REAL-TIME CLOCK
8-PIN DIP MECHANICAL DIMENSIONS



PKG	8-PIN	
DIM	MIN	MAX
A IN.	0.360	0.400
MM	9.14	10.16
B IN.	0.240	0.260
MM	6.10	6.60
C IN.	0.120	0.140
MM	3.05	3.56
D IN.	0.300	0.325
MM	7.62	8.26
E IN.	0.015	0.040
MM	0.38	1.02
F IN.	0.120	0.140
MM	3.04	3.56
G IN.	0.090	0.110
MM	2.29	2.79
H IN.	0.320	0.370
MM	8.13	9.40
J IN.	0.008	0.012
MM	0.20	0.30
K IN.	0.015	0.021
MM	0.38	0.53

DS1307Z 64 X 8 SERIAL REAL-TIME CLOCK
8-PIN SOIC (150-MIL) MECHANICAL DIMENSIONS



PKG	8-PIN (150 MIL)	
DIM	MIN	MAX
A IN.	0.188	0.196
MM	4.78	4.98
B IN.	0.150	0.158
MM	3.81	4.01
C IN.	0.048	0.062
MM	1.22	1.57
E IN.	0.004	0.010
MM	0.10	0.25
F IN.	0.053	0.069
MM	1.35	1.75
G IN.	0.050 BSC	
MM	1.27 BSC	
H IN.	0.230	0.244
MM	5.84	6.20
J IN.	0.007	0.011
MM	0.18	0.28
K IN.	0.012	0.020
MM	0.30	0.51
L IN.	0.016	0.050
MM	0.41	1.27
phi	0°	8°

56-G2008-001

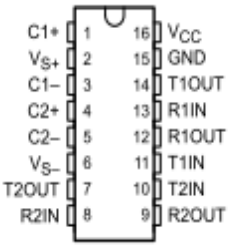
MAX 232

MAX232, MAX232I
DUAL EIA-232 DRIVERS/RECEIVERS

SLLS047I – FEBRUARY 1989 – REVISED OCTOBER 2002

- Meet or Exceed TIA/EIA-232-F and ITU Recommendation V.28
- Operate With Single 5-V Power Supply
- Operate Up to 120 kbit/s
- Two Drivers and Two Receivers
- ±30-V Input Levels
- Low Supply Current . . . 8 mA Typical
- Designed to be Interchangeable With Maxim MAX232
- ESD Protection Exceeds JESD 22 – 2000-V Human-Body Model (A114-A)
- Applications
  - TIA/EIA-232-F
  - Battery-Powered Systems
  - Terminals
  - Modems
  - Computers

MAX232 . . . D, DW, N, OR NS PACKAGE
MAX232I . . . D, DW, OR N PACKAGE
(TOP VIEW)



description/ordering information

The MAX232 is a dual driver/receiver that includes a capacitive voltage generator to supply EIA-232 voltage levels from a single 5-V supply. Each receiver converts EIA-232 inputs to 5-V TTL/CMOS levels. These receivers have a typical threshold of 1.3 V and a typical hysteresis of 0.5 V, and can accept ±30-V inputs. Each driver converts TTL/CMOS input levels into EIA-232 levels. The driver, receiver, and voltage-generator functions are available as cells in the Texas Instruments LinASIC™ library.

ORDERING INFORMATION

T_A	PACKAGE†		ORDERABLE PART NUMBER	TOP-SIDE MARKING
0°C to 70°C	PDIP (N)	Tube	MAX232N	MAX232N
	SOIC (D)	Tube	MAX232D	MAX232
		Tape and reel	MAX232DR	
	SOIC (DW)	Tube	MAX232DW	MAX232
		Tape and reel	MAX232DWR	
	SOP (NS)	Tape and reel	MAX232NSR	MAX232
–40°C to 85°C	PDIP (N)	Tube	MAX232IN	MAX232IN
	SOIC (D)	Tube	MAX232ID	MAX232I
		Tape and reel	MAX232IDR	
	SOIC (DW)	Tube	MAX232IDW	MAX232I
		Tape and reel	MAX232IDWR	

† Package drawings, standard packing quantities, thermal data, symbolization, and PCB design guidelines are available at www.ti.com/sc/package.

Function Tables

EACH DRIVER

INPUT TIN	OUTPUT TOUT
L	H
H	L

H = high level, L = low level

EACH RECEIVER

INPUT RIN	OUTPUT ROUT
L	H
H	L

H = high level, L = low level

logic diagram (positive logic)