ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

CONSTRUCCIÓN DE UN CONTROL DE UNA CARTELERA DIGITAL PARA UN BUS INTERPROVINCIAL

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

LUIS FERNANDO JARAMILLO JARAMILLO

luchifer_reloaded@hotmail.com

DIRECTOR: Ing. Pablo W. López M.

pwlopezm@hotmail.com

Quito, Marzo 2012

DECLARACIÓN

LUIS FERNANDO JARAMILLO JARAMILLO, declaro bajo juramento que el trabajo aquí descrito es de mí autoría; que no ha sido previamente presentado para ningún grado o calificación profesional; y que consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondiente a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la ley de Propiedad Intelectual, por su reglamento y por la normativa institucional vigente.

_____

Luis Fernando Jaramillo Jaramillo

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Luis Fernando Jaramillo Jaramillo, bajo mi supervisión.

Ing. Pablo López
DIRECTOR DEL PROYECTO

RESUMEN

El presente proyecto de titulación tiene como objetivo la construcción de un sistema de control y programación para una matriz de led's de 80 x 7 de alta luminosidad, para indicar el destino, la hora de salida y la hora en tiempo real de un bus interprovincial, y está estructurado en tres capítulos:

Capítulo I: Marco teórico, Capítulo II: Implementación del Sistema, Capítulo III Conclusiones y Recomendaciones.

El Capítulo I habla del transistor porque este es la base de todo lo que se refiere a electrónica, también habla acerca de los led's y sus características, y por último habla acerca del micro-controlador Atmel que es el utilizado en este proyecto.

El Capítulo II expone el cómo fueron hechos los circuitos, la programación y en si la implementación de los mismos en la matriz.

Finalmente en el capítulo III están las conclusiones y recomendaciones del proyecto.

Al final del mismo constan los anexos de los programas utilizados.

AGRADECIMIENTO

Primeramente agradezco a Dios por cada día demostrarme que me ama y hacerme saber que él es el camino la verdad y la vida, agradezco a mis padres por apoyar siempre mis decisiones y guiarme en todo momento, al Ingeniero Pablo López por haber dirigido este trabajo con responsabilidad y carisma, a mi amiga Paola y a Maricela que supieron ser de gran ayuda para la culminación de este proyecto, y a todos los profesores que dictan clases en la Escuela de Formación de Tecnólogos por su entrega y dedicación en bien de la educación de los alumnos.

Luis Fernando Jaramillo Jaramillo

DEDICATORIA

Quiero dedicar este trabajo a las personas más importantes en mi vida, mis padres y mis hermanos.

Luis Fernando Jaramillo Jaramillo

CONTENIDO

PRESENTACIÓN

En el actual sistema de transporte de pasajeros de forma interprovincial, los usuarios de estos buses tiene una manera de informarse cual es el destino de cada bus, ya sea con una matriz de led's, o mediante un rótulo impreso.

En este tipo de rótulos no se puede observar ni la hora de salida, peor un reloj en tiempo real, es por esta razón que decidí diseñar el control de una matriz de 80x7 led's marca AESYS, en donde se puede observar estos parámetros, en el diseño del control de esta matriz se utilizó dos micro-controladores atmel uno es el que maneja el circuito de control donde se introducen los mensajes mediante un teclado matricial, con ayuda de un LCD para observar el mensaje introducido y el otro es el que maneja el barrido y la distribución de cada mensaje, los mismos fueron programados en Bascom el cual es un programa parecido al Visual Basic que utiliza un lenguaje de programación de alto nivel, también decidí utilizar los integrados Atmel debido a que estos circuitos tienen una gran capacidad de memoria interna y además admiten muchas líneas de programación. El paso de la información entre la matriz y el módulo de control se lo hace a través de un cable serial, en el cual se utiliza un par para la transición, otro par para la recepción y un último par para la polarización del mismo.

Para poder utilizar la matriz AESYS 80x7 se probó cual era el modo de desplazar los datos para poder diseñar el control de la misma.

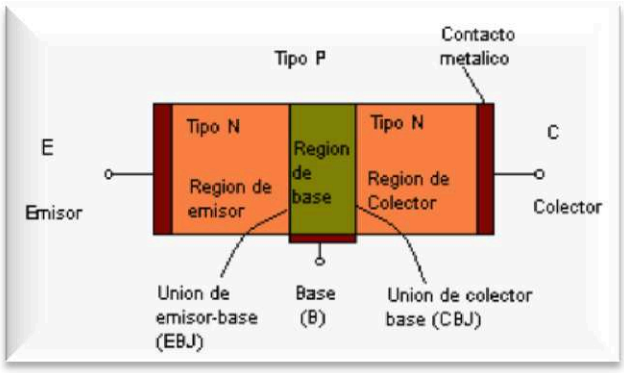
Para finalizar se ensamblo todos los circuitos y se hicieron varias pruebas para el control del correcto funcionamiento.

CAPÍTULO I

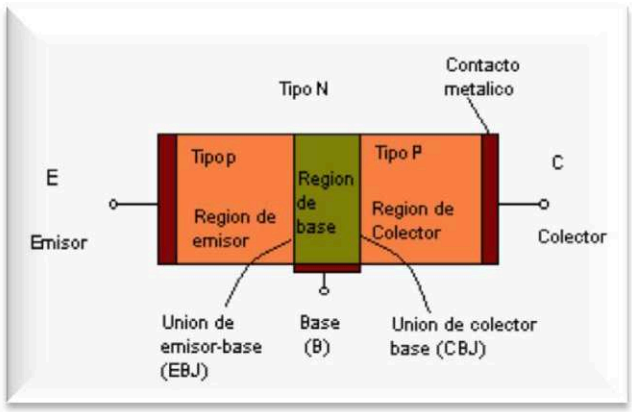
1.-MARCO TEÓRICO

1.1.-EL TRANSISTOR DE UNION BIPOLAR (TBJ)

El transistor TBJ, se utiliza en circuitos discretos y en circuitos integrados, tanto analógicos como digitales, en la Figura 1.1 (a) se muestra un esquema idealizado de un transistor NPN, y la Figura 1.1(b) se muestra un transistor PNP.



(a)



(b)

Figura 1.1 Transistor TBJ; (a) NPN y (b) PNP.[*1]

Cada una de las tres regiones semiconductoras de un transistor tiene una terminal que la conecta al mundo exterior. Las terminales se marcan *emisor (E), base (B) y colector (C).*

[*1] (http://www.solecmexico.com/electronica/transistorbj.pdf)

El transistor TBJ consta de dos junturas PN, la juntura emisor-base (EB) y la juntura colector-base (CB). Según la condición de polarización directa o inversa de cada una de las junturas, se obtienen los modos de operación del TBJ. La tabla 1.1 muestra un cuadro del modo de operación según sea su polarización.

Tabla 1.1 Modos de operación del TBJ

Modo	**Unión EB**	**Unión CB**
Corte	Inverso	Inverso
Activo	Directo	Inverso
Saturación	Directo	Directo

1.1.1.- Zonas de funcionamiento del transistor bipolar:[*2]

1.1.1.1.- Activa Directa:El transistor sólo amplifica en esta zona, y se comporta como una fuente de corriente constante controlada por la intensidad de base (ganancia de corriente).Este parámetro lo suele proporcionar el fabricante dándonos un máximo y un mínimo para una corriente de colector dada (Ic); además de esto, suele presentar una variación acusada con la temperatura y con la corriente de colector, por lo que en principio no podemos conocer su valor. Algunos polímetros son capaces de medir este parámetro pero esta medida hay que tomarla solamente como una indicación, ya que el polímetro mide este parámetro para un valor de corriente de colector distinta a la que circulará por el TBJ una vez en el circuito.

1.1.1.2.-Saturación: En esta zona el transistor es utilizado para aplicaciones de conmutación (potencia, circuitos digitales, etc.), y lo podemos considerar como un cortocircuito entre el colector y el emisor.

1.1.1.3.-Corte: el transistor es utilizado para aplicaciones de conmutación (potencia, circuitos digitales, etc.), y podemos considerar las corrientes que lo atraviesan prácticamente nulas (y en especial Ic).

[*2] http://www.ucm.es/info/electron/laboratorio/componentes/codigos/pag06-03.htm

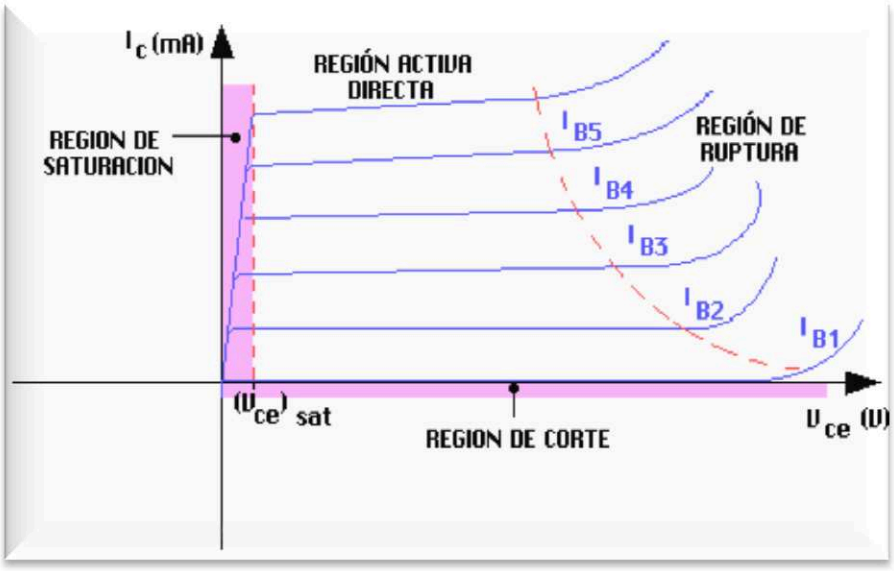


Figura1.2 Regiones de Operación del TBJ

1.2.-DIODO EMISOR DE LUZ [*3]

El diodo **LED** (***L**ight-**E**mitting **D**iode*: Diodo Emisor de Luz), es un dispositivo semiconductor que emite luz incoherente de espectro reducido cuando se polariza de forma directa la unión PN en la cual circula por él una corriente eléctrica . Este fenómeno es una forma de electroluminiscencia, el LED es un tipo especial de diodo que trabaja como un diodo común, pero que al ser atravesado por la corriente eléctrica, emite luz. Este dispositivo semiconductor está comúnmente encapsulado en una cubierta de plástico de mayor resistencia que las de vidrio que usualmente se emplean en las lámparas incandescentes. Aunque el plástico puede estar coloreado, es sólo por razones estéticas, ya que ello no influye en el color de la luz emitida. Usualmente un LED es una fuente de luz compuesta con diferentes partes, razón por la cual el patrón de intensidad de la luz emitida puede ser bastante complejo.

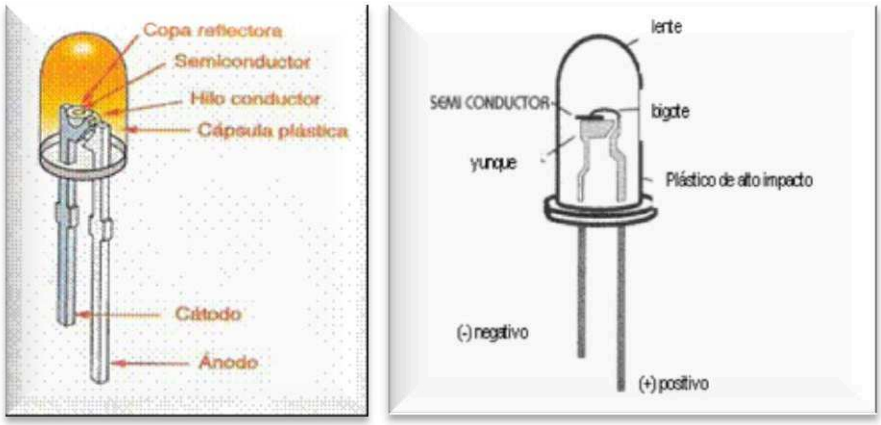
Para obtener una buena intensidad luminosa debe escogerse bien la corriente que atraviesa el LED y evitar que este se pueda dañar; para ello, hay que tener en cuenta que el voltaje de operación va desde 1,8 hasta 3,8 voltios aproximadamente (lo que está relacionado con el material de fabricación y el color de la luz que emite) y la gama de intensidades que debe circular por él varía según su aplicación. Los valores típicos de corriente directa de

[*3] http://www.monografias.com/trabajos60/diodo-led/diodo-led2.shtml

polarización de un LED están comprendidos entre los 10 y 20 miliamperios (mA) en los diodos de color rojo y de entre los 20 y 40 miliamperios (mA) para los otrosLED's. Los LED's tienen enormes ventajas sobre las lámparas indicadoras comunes, como su bajo consumo de energía, su mantenimiento casi nulo y con una vida aproximada de 100,000 horas. Para la protección del LED en caso haya picos inesperados que puedan dañarlo. Se coloca en paralelo y en sentido opuesto un diodo de silicio común

En general, los LED's suelen tener mejor eficiencia cuanto menor es la corriente que circula por ellos, con lo cual, en su operación de forma optimizada, se suele buscar un compromiso entre la intensidad luminosa que producen (mayor cuanto más grande es la intensidad que circula por ellos) y la eficiencia (mayor cuanto menor es la intensidad que circula por ellos). La Figura 1.3 (a) muestra su simbología y la Figura 1.3 (b) muestra su estructura.

***(a)*[4]**



***(b)*[5]**

Figura 1.3 (a) Simbología del LED, (b) Estructura.

[*4] http://es.wikipedia.org/wiki/Archivo:Simbolo_Electrico_diodo_LED.svg
[*5] http://electricidad-viatger.blogspot.com/2008/11/el-diodo.html

1.2.1.- Composición del Led [*6]

Existen **LED's** de varios colores que dependen del material con el cual fueron construidos. Hay de color rojo, verde, amarillo, ámbar, infrarrojo, entre otros. La tabla 1.2 muestra los colores y el tipo de material según el tipo de color.

Tabla 1.2 Material utilizado según el tipo de color del LED.

Material	Longitud de onda	color
Arseniuro de Galio y Zinc (GaAS:Zn)	9000 A	Infra-rojo
Arseniuro de Galio y Aluminio (GaAlAs)	6600 A	Rojo
Fosfuro de Galio y Arsénico (GaAsP.6)	6100 A	Ámbar
Fosfuro de Galio, Arsénico y Nitrógeno (GaAsP.8:N)	5900 A	Amarillo
Fosfuro de Galio y Nitrógeno (GaP:N)	5600 A	Verde

1.2.2.- Funcionamiento del Led.

Un electrón al pasar de la banda de conducción a la de valencia, pierde energía; esta energía perdida se puede manifestar en forma de un fotón desprendido, con una amplitud, una dirección y una fase aleatoria. El que esa energía se manifieste en calor por ejemplo, va a depender principalmente del tipo de material semiconductor. Cuando Al polarizar directamente un diodo LED conseguimos que por la unión PN sean inyectados huecos en el material tipo N y electrones en el material tipo P; O sea los huecos de la zona p se mueven hacia la zona n y los electrones de la zona n hacia la zona p, produciéndose por consiguiente, un intercambio de portadores minoritarios.

Ambos desplazamientos de cargas constituyen la corriente que circula por el diodo. Si los electrones y huecos están en la misma región, pueden recombinarse, es decir, los electrones pueden pasar a "ocupar" los huecos, "cayendo" desde un nivel energético superior a otro inferior más estable. La Figura 1.4 muestra la equivalencia eléctrica de un diodo LED polarizado en forma directa.

[*6] http://www.monografias.com/trabajos60/diodo-led/diodo-led.shtml

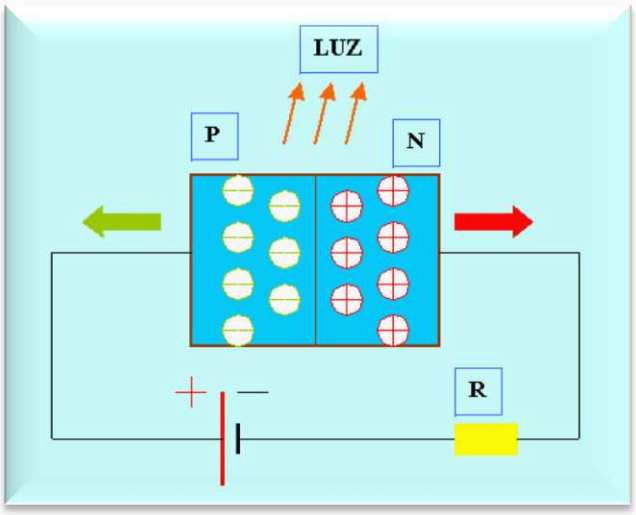


Figura 1.4 Equivalencia de diodo LED polarizado en forma directa.

Cuando estos portadores se recombinan, se produce la liberación de una cantidad de energía proporcional al salto de banda de energía del material semiconductor. Una parte de esta energía se libera en forma de luz, mientras que la parte restante lo hace en forma de calor, estando determinadas las proporciones por la mezcla de los procesos de recombinación que se producen.

La energía contenida en un fotón de luz es proporcional a su frecuencia, es decir, su color. Cuanto mayor sea el salto de banda de energía del material semiconductor que forma el LED, más elevada será la frecuencia de la luz emitida. La Figura 1.5 muestra el diagrama típico para polarizar un diodo LED

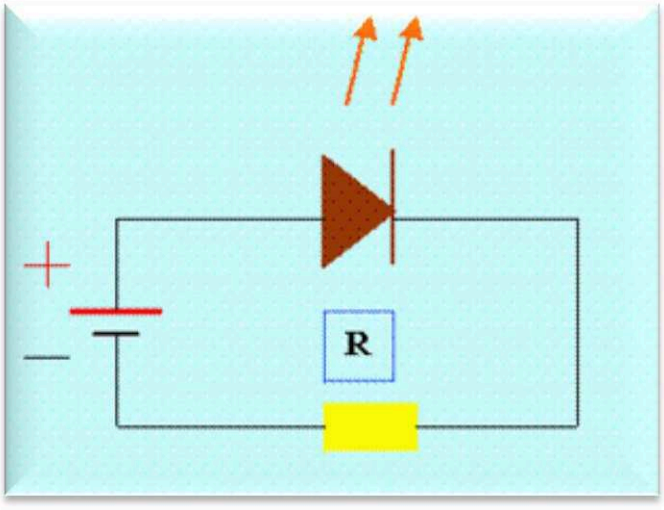


Figura 1.5 Esquema Típico de diodo LED.

1.2.3.-Aplicaciones del Led

Los diodos infrarrojos (IRED) se emplean desde mediados del siglo XX en mandos a distancia de televisores, habiéndose generalizado su uso en otros electrodomésticos como equipos de aire acondicionado, equipos de música, etc. y en general para aplicaciones de control remoto, así como en dispositivos detectores. Los LED se emplean con profusión en todo tipo de indicadores de estado (encendido/apagado) en dispositivos de señalización (de tránsito, de emergencia, etc.) y en paneles informativos. También se emplean en el alumbrado de pantallas de cristal líquido de teléfonos móviles, calculadoras, agendas electrónicas, etc., así como en bicicletas y usos similares. Existen además impresoras LED.

Se utiliza ampliamente en aplicaciones visuales, como indicadoras de cierta situación específica de funcionamiento y desplegar contadores

1.2.4.-Ventajas y desventajas del Led

Las principales ventaja de un LED son fiabilidad, mayor eficiencia energética, mayor resistencia a las vibraciones, mejor visión ante diversas circunstancias de iluminación, menor disipación de energía, menor riesgo para el medio ambiente, capacidad para operar de forma intermitente de modo continuo, respuesta rápida, etc. Además con un LED se pueden producir luces de diferentes colores con un rendimiento luminoso elevado, a diferencia de muchas de las lámparas utilizadas hasta ahora, que tienen filtros para lograr un efecto similar (lo que supone una reducción de su eficiencia energética). Todo ello pone de manifiesto las numerosas ventajas que los LED ofrecen. También se utilizan en la emisión de señales de luz que se trasmiten a través de fibra óptica.

Las desventajas del **LED** son que su potencia de iluminación es tan baja, que su luz es invisible bajo una fuente de luz brillante y que su ángulo de visibilidad está entre los 30° y 60°. Este último problema se corrige con cubiertas difusores de luz.

1.2.5.- Led’s de potencia.

Hoy en día, se están desarrollando y empezando a comercializar led's con prestaciones muy superiores a las de hace unos años y con un futuro prometedor en diversos campos, incluso en aplicaciones generales de iluminación. Como ejemplo, se puede destacar queNichia Corporation ha desarrollado led’s de luz blanca con una eficiencia luminosa de 150 lm/W, utilizando para ello una corriente de polarización directa de 20 miliamperios (mA). Esta eficiencia, comparada con otras fuentes de luz solamente en términos de rendimiento, es aproximadamente 1,7 veces superior a la de la lámpara fluorescente con prestaciones de color altas (90 lm/W) y aproximadamente 11,5 veces la de una lámpara incandescente (13 lm/W).

Su eficiencia es incluso más alta que la de la lámpara de vapor de sodio de alta presión (132 lm/W), que está considerada como una de las fuentes de luz más eficientes.

Son muchos los beneficios de la luz LED de alta potencia, pueden tener una duración de hasta 100.000 horas, ya que no tienen filamentos, este hecho hace que no se fundan como lo hacen las tradicionales bombillas. Estos dispositivos de estado sólido son muy resistentes a los golpes, todo son ventajas.

Día a día, estos emisores de luz o diodos ganan terreno a la iluminación tradicional que todos conocemos, sobre todo porque las ventajas que nos presentan son muchas y entre una de esas, su rendimiento lumínico, que es muy notable.

Estos nuevos LED'S no emiten ultravioletas, ni tampoco infrarrojos y además no calientan la superficie a la que iluminan, pero la mejor ventaja es su reducido consumo y su alta eficiencia. Una bombilla emplea sólo un 10% de cada vatio para iluminar, mientras que el resto es calor pero en los LED'S, es totalmente lo contrario, un 90% de iluminación y un 10% de calor.

Además otra cualidad es que no tiene pérdidas por la reflexión, los sistemas como los dicroicos necesitan de reflectores para concentrar la luz al lugar donde queremos iluminar, lo que supone perder un 60% de

efectividad, mientras que el LED no precisa estos sistemas y la luz puede ser dirigida a la zona que queremos iluminar con una eficiencia del 90%.

1.3.- MICRO-CONTROLADOR ATMEGA16

1.3.1.- Introducción [*7]

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.

Adicionalmente, ATMEL también proporciona en línea el entorno software (AVR estudio) que permite editar, ensamblar y simular el código fuente.

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.

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

Las familias AVR rápidamente han crecido en el mercado y se dispone de las siguientes categorías:

• TINY AVR:son microcontroladores de propósito general con memoria flash hasta 2 Kbytes y 128 bytes de memorias SRAM y EEPROM.

• AVR: Microcontroladores de propósito general con 8 Kbytes de memoria flash y 512 bytes de memoria SRAM y EEPROM.

• Mega AVR Memoria flash hasta 256 Kbytes, 4 Kbytes de memoria EEPROM y SRAM

Los tipos de encapsulado del microcontrolador del ATMEGA presenta desde 28 pines hasta 100 pines en la forma de DIP, TQFP y MLF y su voltaje de alimentación está en el rango de 1.8 a 5.5 voltios. Se presenta en la tabla 1.3sus características principales.

Tabla 1.3 Características del microcontrolador ATMEGA AVR.

Product	Flash (KB)	EEPROM (Bytes)	RAM (Bytes)	I/O	SPI	USART	USI	TWI	PWM	On-Chip Debug		10-bit ADC	LCD
										JTAG	debugWire		
megaAVR													
ATmega 48	4	256	512	23	1	1	–	1	5	–	Y	8	–
ATmega8	8	512	1K	23	1	1	–	1	3	–	–	8	–
ATmega88	8	512	1K	23	1	1	–	1	5	–	Y	8	–
ATmega8515	8	512	512	35	1	1	–	–	3	–	–	–	–
ATmega8535	8	512	512	32	1	1	–	1	4	–	–	8	–
ATmega16	16	512	1K	32	1	1	–	1	4	Y	–	8	–
ATmega162	16	512	1K	35	1	2	–	–	6	Y	–	–	–
ATmega168	16	512	1K	23	1	1	–	1	5	–	Y	8	–
ATmega32	32	1K	2K	32	1	1	–	1	4	Y	–	8	–
ATmega64	64	2K	4K	53	1	2	–	1	8	Y	–	8	–
ATmega128	128	4K	4K	53	1	2	–	1	8	Y	–	8	–
ATmega256	256	4K	8K	53	1	2	–	1	16	–	Y	8	–
LCD AVR													
ATmega169	16	512	1K	53	1	1	Y	–	4	Y	–	8	Y
ATmega329	32	1K	2K	53	1	1	Y	–	4	Y	–	8	Y

El fabricante ATMEL produce micro-controladores de gama alta, media y baja. Para el caso del presente proyecto se utilizó el micro-controlador ATMEGA16 de gama baja,el cual posee las siguientes características:

- Tecnología RISC (conjunto de instrucciones reducidas).

- Arquitectura HARVARD (bus de datos de memoria de datos y programa difierentes).

- Micro-controlador de 8 bits (programa y datos).

- 32 Registros de propósito general, los cuales se acceden mediante operaciones de carga y almacenamiento.

- Los micro-controladores AVR, tienen una cañería (PIPELINE) con dos etapas (traer y ejecutar), que les permite utilizar un ciclo de reloj en la mayoría de instrucciones.

- Voltaje de funcionamiento 4.5 voltios y una corriente máxima de 200 mA en las terminales VCC y GND.

- Porticos de entrada y salida programables (Porticos A, B, C y D).

- Convertidores Analógicos/Digital de 8 y 10 bits, 8 canales de un solo terminal, 2 canales diferenciales con ganacia programable de x1, x10, x200.

- Interfaces seriales. SPI, I2C, UART.

- Memoria EEPROM de 256 bytes.

- Memoria de programa de 16Kbytes.

- Memoria de datos de 1Kbyte.

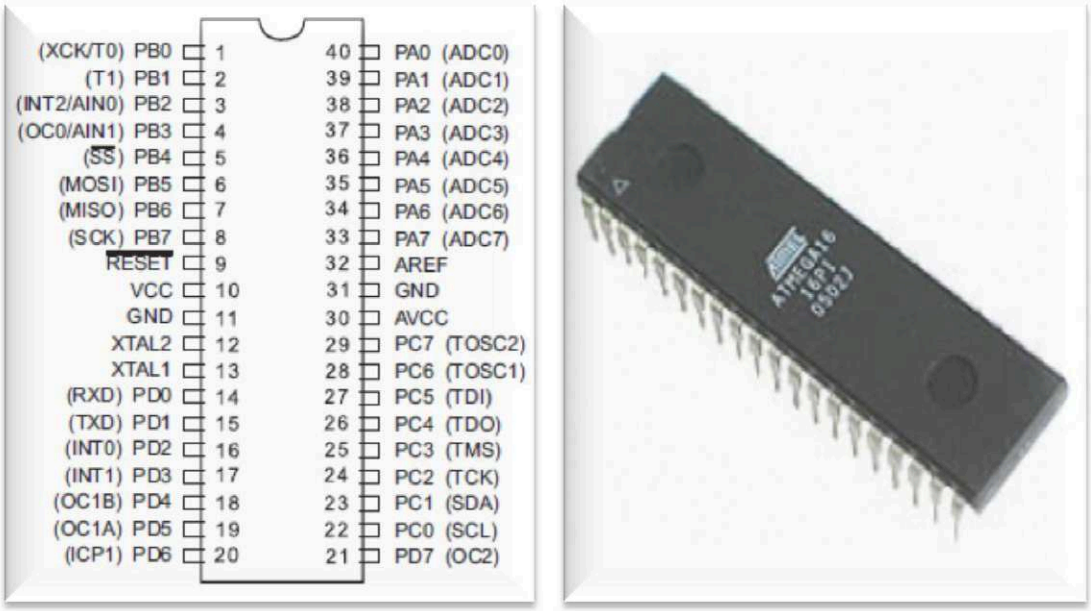
- Oscilador interno programable hasta 8MHz.

- Timers programables con fuente de interrupción.

- Detector programable de bajo voltaje.

✪ Canales PWM.

La Figura 1.6 muestra la distribución de pines del micro-controlador del ATMEGA16para el encapsulado tipo DIP.



***(a)*[8]** ***(b)*[9]**

Figura 1.6 (a)Distribución de pines del ATMEGA16, (b)Encapsulado tipo DIP

1.4.- INTERFAZ RS485 [(*10)].

La interfaz RS485 es un sistema de interconexión para transmisión de datos a grandes distancias y apto para operar en ámbitos eléctricamente ruidosos. El IC comúnmente utilizado es el SN76156.

La Figura 1.7 muestra el diagrama de conexión típica con varios terminales y un computador personal.

[*8] http://loja.multcomercial.com.br/ecommerce_site/produto_8421_4689_Microcontrolador-ATMEGA16-16PU-Atmel
[*9] http://www.voltix.com.mx/openvoltix/ATMEGA16-DIP
[*10]http://www.neoteo.com/rs485-domotica-al-alcance-de-tu-mano-15810

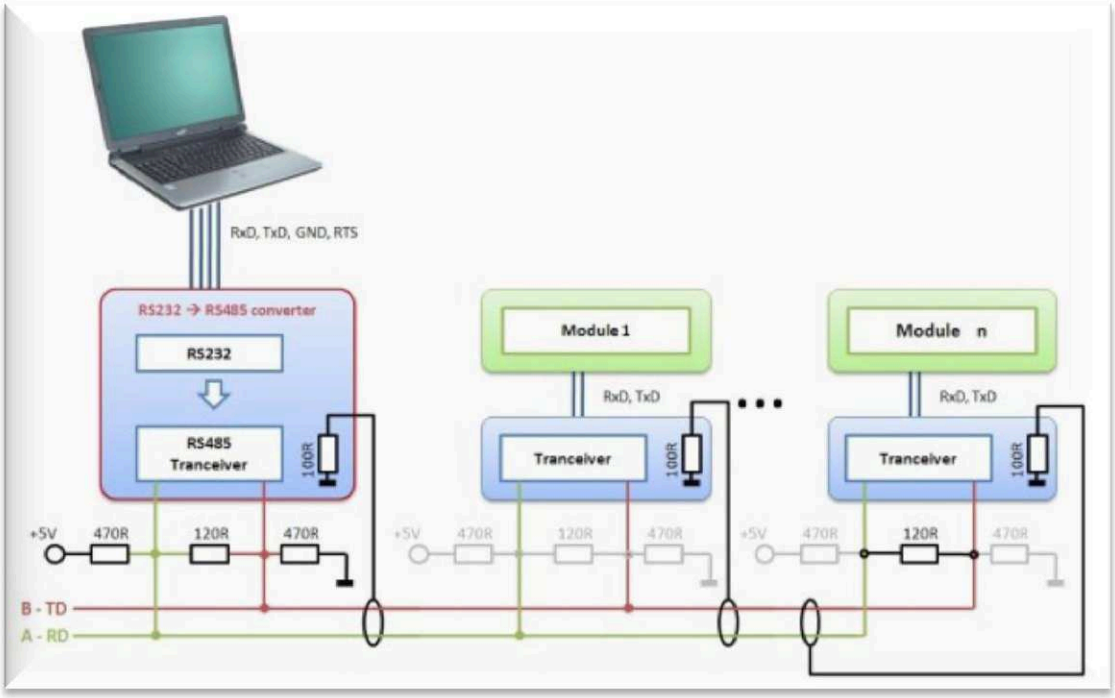
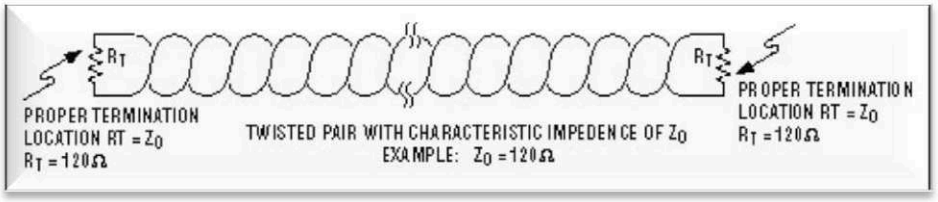


Figura 1.7Topología de un Bus RS48

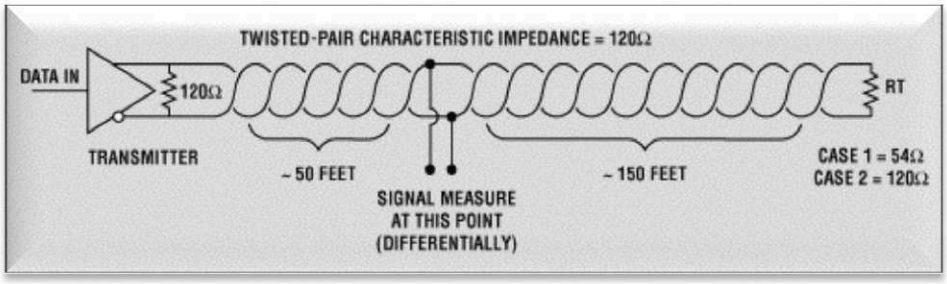
El bus permite una velocidad de datos de 10 y hasta 20 Mbps (a 12 metros de distancia), y de 100 Kbps cuando se conectan terminales o módulos separados 1200 metros entre sí. El sistema permite utilizar hasta 32 terminales. Para la conexión es preferible que el par de cables que transporta la información sea blindado ya que ayudaría a atenuar los ruidos eléctricos que pueden filtrarse entre los datos del sistema diferencial que utiliza el estándar RS485.

Las especificaciones del estándar RS485 (cuyo nombre oficial es TIA/EIA–485-A) no determinan claramente cómo debe ser el correcto cableado de una red, sin embargo, algunas recomendaciones pueden interpretarse dentro del texto de la norma y han sido estudiadas y ensayadas por ingenieros, tanto en forma conceptual como en función del método de prueba y error.

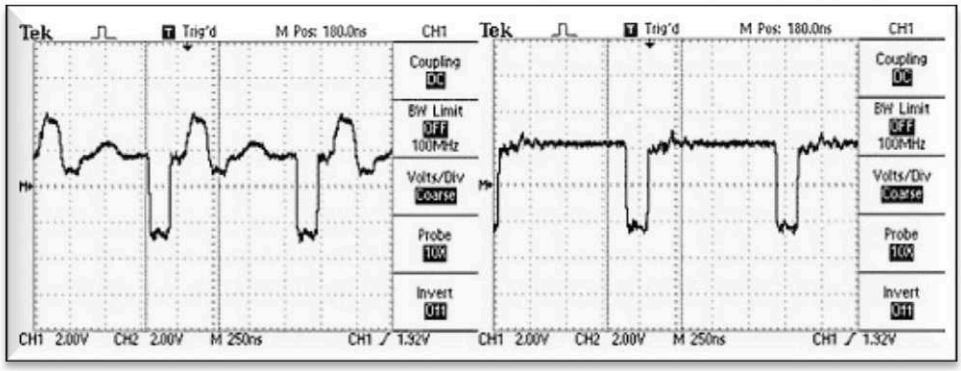
Debido a que altas frecuencias intervienen en el intercambio de datos, que las distancias entre las terminales siempre son inciertas, y que los cables apropiados a utilizar no se determinan en el estándar, se acepta el uso de un par de cables trenzados comunes que tienen una impedancia aproximada de 120 Ohms. Las terminaciones inapropiadas de la línea a utilizar se traducen en reflexiones no deseadas de la señal, tal como muestran los ejemplos en la Figura 1.8.



(a)



(b)



(c)

Figura 1.8 Terminación extremo Interfaz RS485, (a) Correcta, (b) Incorrecta, (c) Formas de onda

En las imágenes (extraídas de la Web oficial de MAXIM) de la Figura 1.8 (c) se puede apreciar claramente la distorsión sufrida en la señal, cuando el final de una línea no tiene una terminación adecuada. La reflexión ocasionada puede llevar a distorsionar y perder por completo los datos transmitidos.

La base del sistema se fundamenta en la transmisión de datos en forma diferencial. Es decir, por ambos cables viaja la misma información, pero desfasada 180° en un cable respecto al otro. De esta forma, cualquier interferencia que pueda introducirse en el cableado lo hará en ambos hilos por igual, con la misma polaridad y amplitud. En el destino de la terminal, sea en el

ordenador o en el dispositivo colocado a la distancia, las señales se restituyen en polaridad y los picos de ruidos que se habían introducido con la misma polaridad en ambos cables, al invertirse las señales, se neutralizan y eliminan entre sí, y se recupera de esta forma la señal útil que se desea transmitir.

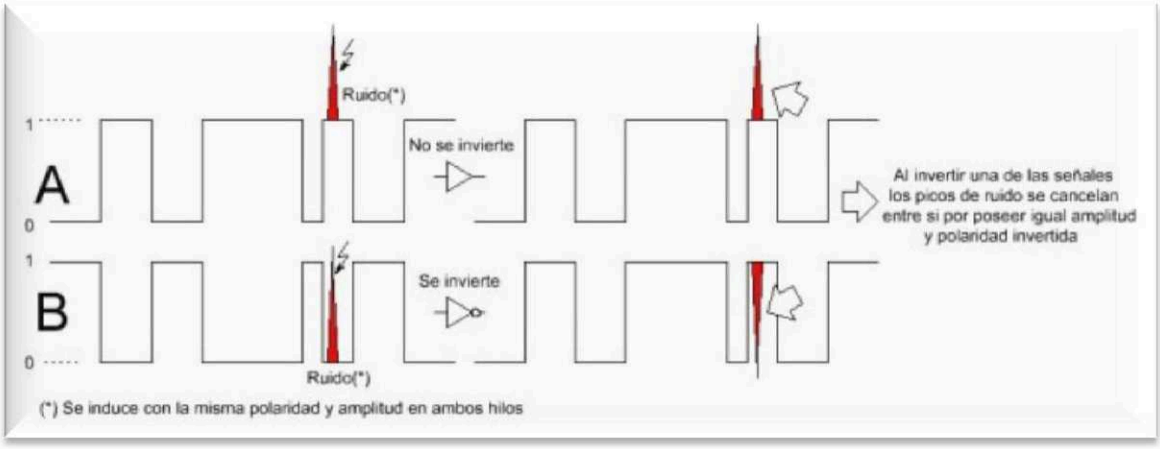


Figura 1.9 Forma de onda de la interfaz RS485, incluido ruido

Cuando el cableado recorre un ambiente ruidoso y hostil, como puede suceder en una instalación industrial, el tercer cable que oficia de tierra o GND también se transforma en un elemento que recepciona y lleva hasta las terminales o módulos los ruidos inducidos en él. Por este motivo siempre es recomendable colocar una resistencia de 100 Ohms en la conexión a GND en cada uno de los circuitos de las terminales.

Entre las múltiples diferencias fundamentales que existen respecto al estándar RS232 es que el RS485 se maneja con niveles TTL de tensión, mientras que el RS232 maneja tensiones de ambas polaridades con valores absolutos de 3 a 15 Volts. RS232 permite comunicaciones "*full-duplex*" (ambos terminales transmiten y reciben datos en forma simultánea), pero su distancia de trabajo es de tan sólo 12 metros; además, se requieren al menos 8 cables para una comunicación full y es muy propenso a ser afectado por el ruido eléctrico.

1.5.- INTERFAZ I2C [*11]

Muchos de los equipos electrónicos incluyen circuitos integrados que utilizan el bus I2C, como por ejemplo, las memorias 24Cxx, los procesadores de señal o

[*11] http://www.comunidadelectronicos.com/articulos/i2c.htm

"jungla" en televisores (LA7610, TA1223, DTC810,...), codificadores de video de reproductores de DVD (SAA 7128, TC 90A32F,...), preamplificadores de video en monitores (KB 2502), etc.

A continuación se presenta las características más relevantes del bus i2c:

- ✪ Se necesitan solamente dos líneas, la de datos (SDA) y la de reloj (SCL).
- ✪ Cada dispositivo conectado al bus tiene un código de dirección seleccionable mediante software. Habiendo permanentemente una relación Master/ Slave entre el micro y los dispositivos conectados
- ✪ El bus permite la conexión de varios Masters, ya que incluye un detector de colisiones.
- ✪ El protocolo de transferencia de datos y direcciones posibilita diseñar sistemas completamente definidos por software.
- ✪ Los datos y direcciones se transmiten con palabras de 8 bits.

1.5.1.-Funcionamiento del bus I2C

Las líneas SDA y SCL transportan información entre los dispositivos conectados al bus lo que se observa en la Figura 1.10. Cada dispositivo es reconocido por su código (dirección) y puede operar como transmisor o receptor de datos. Además, cada dispositivo puede ser considerado como Master o Slave.

El Master es el dispositivo que inicia la transferencia en el bus y genera la señal de reloj (Clock), el Slave (esclavo) es el dispositivo direccionado.

Las líneas SDA (serial Data) y SCL (serial Clock) son bidireccionales, conectadas al positivo de la alimentación a través de las resistencias de pull-up (valores de resistencia entre 1k-10k). Cuando el bus está libre, ambas líneas están en nivel alto.

La transmisión bidireccional serie (8-bits) de datos puede realizarse a 100Kbits/s en el modo standard o 400 Kbits/s en el modo rápido. La cantidad de dispositivos que se pueden conectar al bus está limitada, solamente, por la máxima capacidad permitida: 400 pF.

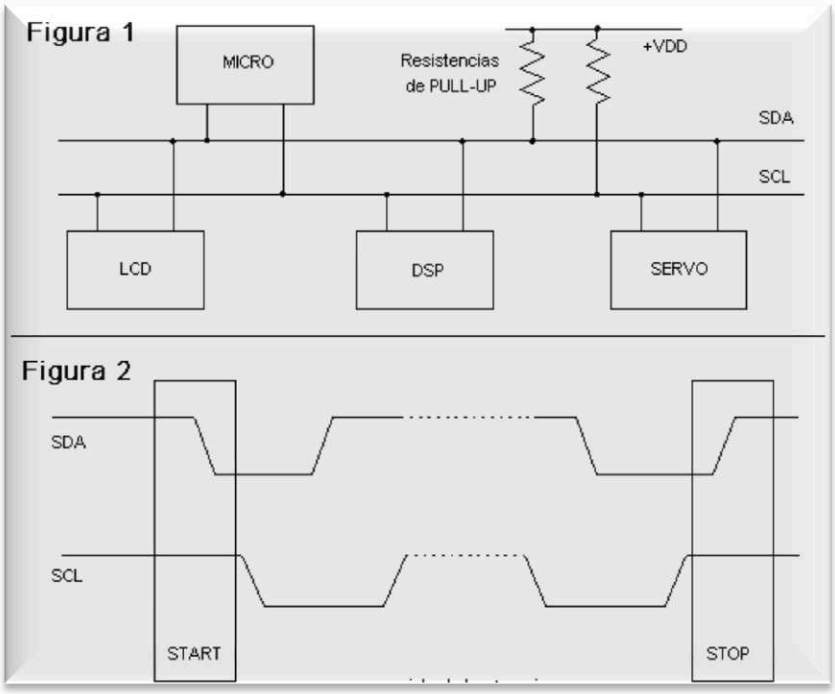


Figura 1.10 Conexión interfaz I2C, (a) Conexión física, (b) Niveles de voltaje

1.5.1.1.- Condiciones de START y STOP

Antes de que se establezca un intercambio de datos entre el circuito Master y los Esclavos, el Master debe informar el comienzo de la comunicación (condición de Start): la línea SDA cae a cero mientras SCL permanece en nivel alto. A partir de este momento comienza la transferencia de datos. Una vez finalizada la comunicación se debe informar de esta situación (condición de Stop). La línea SDA pasa a nivel alto mientras SCL permanece en estado alto. Ver Figura 1.10

1.5.1.2.- Transferencia de datos

El maestro genera la condición de Start. Cada palabra puesta en el bus SDA debe tener 8 bits, la primera palabra transferida contiene la dirección del Esclavo seleccionado.

Luego el master lee el estado de la línea SDA, si vale 0 (impuesto por el esclavo), el proceso de transferencia continúa. Si vale 1, indica que el circuito

direccionado no valida la comunicación, entonces, el Maestro genera un bit de stop para liberar el bus I2C.

Este acuse de recibo se denomina ACK (acknowledge) y es una parte importante del protocolo I2C.Al final de la transmisión, el Maestro genera la condición de Stop y libera el bus I2C, las líneas SDA y SCL pasan a estado alto.

1.6.- MATRIZ DE LED'S.

Una matriz de LED's consiste en un arreglo de LED's que pueden ser encendidos y apagados individualmente desde un microcontrolador. Se puede considerar como una pantalla de pocos pixeles en los cuales se puede presentar gráficos y textos, tanto estáticos como en movimiento.(*12)

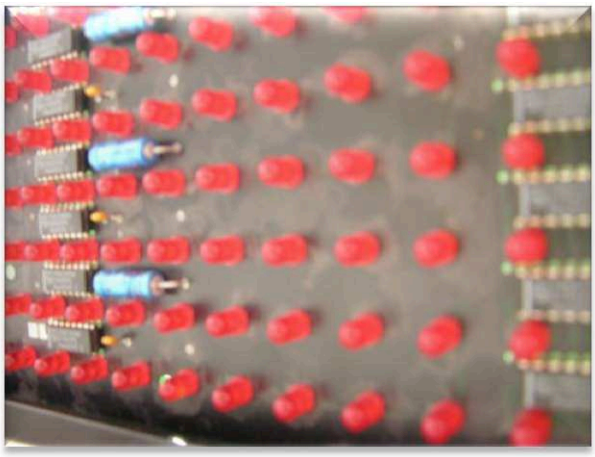


Figura1.11 Matriz de Led's

La pantalla (cartelera digital) está formada por una serie de filas y columnas. La intersección entre ambas contiene un LED. Para que este encienda, tiene que recibir simultáneamente un "0" en la fila, y un "1" en la columna. Cuando se dan estas condiciones, la electrónica de la placa se encarga del encendido del LED en cuestión.

*12http://galaxi0.wordpress.com/about/salidas-y-entradas-digitales/matriz-de-leds/

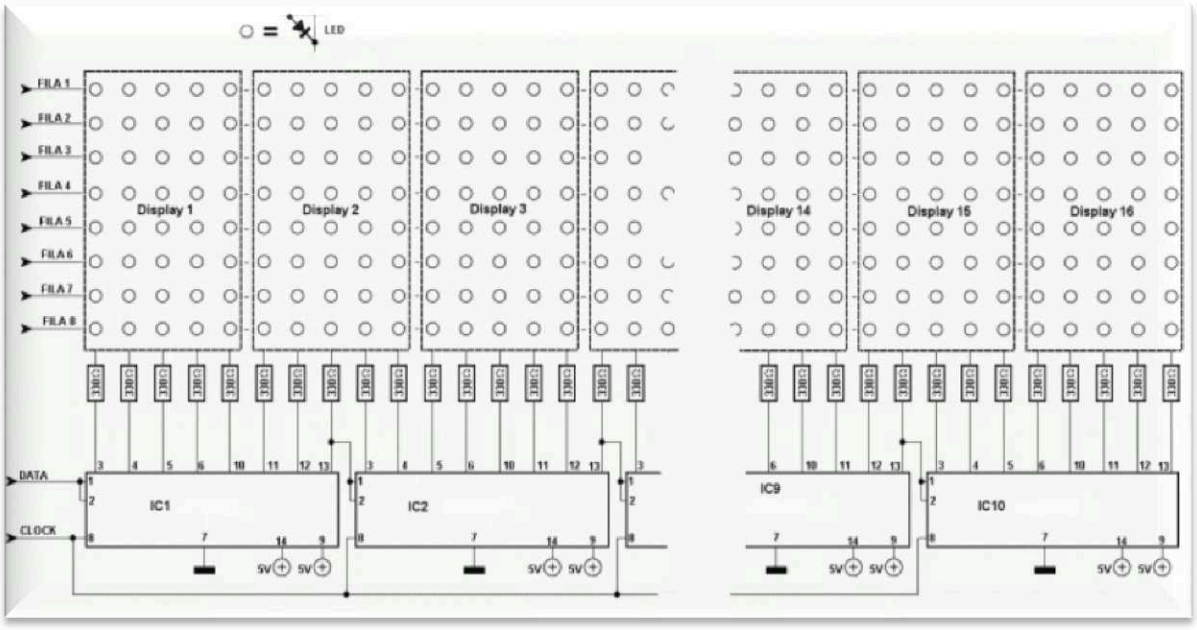


Figura1.12 diagrama de la matriz de led's

La forma de generar un mensaje sobre la matriz es relativamente sencilla, si nos atenemos al siguiente algoritmo:

1) Apagar todas las filas.

2) Escribir los valores correspondientes a la primer fila en el registro de desplazamiento, teniendo en cuenta que el primer digito binario colocado corresponde al último LED de la fila, y el ultimo en poner al de la primer columna.

3) Encenderla primer fila, esperar un tiempo, y volver a apagarla.

4) Repetir los pasos 2 y 3 para las filas restantes.

El tiempo de la demora debe ser tal que permita una visualización correcta, sin molestos parpadeos y con los LED'S brillantes. Hay que tener en cuenta que si utilizamos tiempos mayores para el encendido de cada fila, el brillo de los LED'S será mayor, pero también aumentará el parpadeo. La forma de transformar este algoritmo en un programa funcional depende de cada programador, y puede ser más o menos complejo según se permitan diferentes tipos de caracteres, animaciones, etc. [*13]

[*13] http://www.ucontrol.com.ar/wiki/index.php?title=Funcionamiento_de_una_matriz_de_LEDs

Es importante tomar en cuenta los siguientes parámetros:

1.6.1.-Resolución de la matriz

La matriz es el parámetro descriptivo más importante de un letrero electrónico.

Matriz es la cantidad de puntos luminosos que componen el cartel. Se la expresa en cantidad de filas por cantidad de columnas. De la matriz dependen la visibilidad del cartel, la calidad de su tipografía y la cantidad de caracteres visibles.

En este caso se utiliza una matriz de 7x80 led's.

1.6.2.-Diámetro de cada LED o Punto

Los puntos luminosos de estos carteles se forman con un componente denominado LED. Por eso, a la matriz anterior se le dice matriz de LED.

Normalmente cada punto se hace con un solo LED, que generalmente es de 5 mm de diámetro. Hay carteles que para mejorar la visibilidad usan varios LED's por punto, o de LED's de diámetro mayor que 5 mm. Es por lo tanto útil saber cuántos LED's por punto se emplean y de qué diámetro.

1.6.3.-Visibilidad

La visibilidad o alcance visual hace referencia al rango de distancias desde donde el cartel puede ser leído por una persona con vista normal.

Un cartel de LED se lee desde más lejos que uno convencional (impreso) con la misma altura de letra, porque las letras son luminosas. Si la matriz es buena, un letrero luminoso de 5 cm de altura de letra se lee perfectamente desde 25 metros.

El tamaño influye en la visibilidad de lejos, pero no tanto como la matriz y el diámetro de cada punto luminoso. De lejos, un display de 5 cm de altura se ve casi tanto como uno de 7 cm, si es que su matriz y los diámetros de LED son iguales.

Sin embargo, para leer a poca distancia, es mejor el cartel de 5 cm, porque los LED’s dispersos son incómodos para leer de cerca.

1.6.4.-Brillo

El brillo de los carteles luminosos se clasifica generalmente en:

Brillo estándar: Se usan poco, en carteles de bajo costo tales como los importados más económicos.

Alto brillo: Son los más empleados.

Híper/Súper brillo: Se usan exclusivamente para carteles para exteriores que tengan incidencia directa de la luz solar.

El brillo más adecuado dependerá de la intensidad de la luz que incida sobre el letrero. Por ejemplo, si va a recibir luz solar directa durante la mayor parte de la jornada, convendrá utilizar un letrero de súper/híper brillo, los cuales se especifican como aptos para tal aplicación.

Un brillo excesivo hace incómoda la lectura, así que para estudiar la necesidad de un cartel para sol directo hay que analizar la proporción del tiempo que vaya a estar recibiendo sol directo sobre el total.

Además, los LED's de mucho brillo generalmente concentran la luz, empeorando así un parámetro importante que hace a la visibilidad del letrero, llamado ángulo de visualización. Por lo tanto, carteles con mayor brillo que el necesario no son convenientes.

1.6.5.-Angulo de Visualización

Este parámetro nos indica que tan "de lado" podemos colocarnos sin dejar de poder leer el cartel. En algunos modelos es muy amplio, mientras que en otros necesitamos estar parados prácticamente de frente para poder leerlos.

A mayor ángulo de visualización, mayor área de captación. Por ejemplo, el área de cobertura de un cartel con ángulo de visualización de 60º es de solo un tercio de la de uno con ángulo de 180º. Eso quiere decir que el segundo transmite mensajes a tres veces más público.

1.6.6.-Cantidad de Caracteres

La máxima cantidad de caracteres que puede mostrar un display en un mismo instante nos da pautas sobre que tan destacados serán los mensajes en movimiento y qué tanto se va a poder aprovechar los efectos de texto fijo.

Esta cantidad depende fundamentalmente de cuántas columnas tenga la matriz. Es otro ejemplo de la importancia de la resolución de la matriz, por eso decíamos que éste es el parámetro más importante.

Hay también técnicas para aumentar la cantidad de caracteres, tales como las tipografías de ancho variable, en las cuales por ejemplo una "i" ocupa menos que una "m". Estas, además de maximizar la cantidad de caracteres, le dan un aspecto más elegante al texto.

1.6.7.-Memoria

Muchas aplicaciones requieren carteles con memorias que conserven su contenido aunque estén apagados. Esto se solía implementar con pilas recargables, que estaban soldadas dentro del cartel. El problema con esas pilas es que su vida útil es corta (3 a 5 años). Entonces, los carteles requerían mantenimiento a los pocos años de uso.

Este cartel, en cambio, utiliza la memoria EEPROM para prescindir de pilas, disminuyendo así radicalmente el tiempo entre mantenimientos. [*14]

1.7.- MEMORIA EEPROM

EEPROM son las siglas de *Electrically Erasable Programmable Read-Only Memory* (ROM programable y borrable eléctricamente). Es un tipo de memoria ROM que puede ser programada, borrada y reprogramada eléctricamente, a diferencia de la EPROMque ha de borrarse mediante un aparato que emite rayos ultravioletas. Son memorias no volátiles.

[*14] http://www.indicart.com.ar/letreros-electronicos.htm

Las celdas de memoria de una EEPROM están constituidas por un transistor MOS, que tiene una compuerta flotante, su estado normal esta cortado y la salida proporciona un 1 lógico.

Aunque una EEPROM puede ser leída un número ilimitado de veces, sólo puede ser borrada y reprogramada entre 100.000 y un millón de veces. [*15]

1.8.- RELOJ EN TIEMPO REAL [*16]

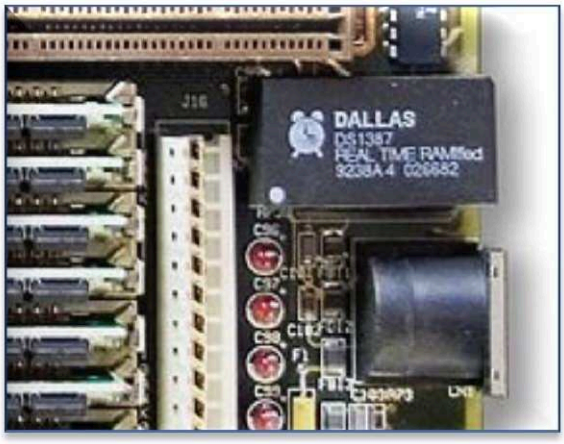


Figura1.13 Reloj en tiempo real Dallas

Un reloj en tiempo real (en inglés, real-time clock, RTC), es un reloj de un ordenador, incluido en un circuito integrado, que mantiene la hora actual. Aunque el término normalmente se refiere a dispositivos en ordenadores personales, servidores y sistemas embebidos, los RTCs están presentes en la mayoría de los aparatos electrónicos que necesitan el tiempo exacto.

El término se usa para evitar la confusión con los relojes hardware ordinarios que sólo son señales que dirigen circuitos digitales, y no cuentan el tiempo en unidades humanas. Los RTC no deben ser confundidos con la computación en tiempo real (en inglés, real-time computing), que comparte su acrónimo de tres letras, pero que no se refiere directamente al tiempo del día.

[*15] http://es.wikipedia.org/wiki/EEPROM
[*16] http://es.wikipedia.org/wiki/Reloj_en_tiempo_real

Las ventajas de usar un reloj en tiempo real son:

- Bajo consumo de energía (importante cuando está funcionando con una pila)
- Libera de trabajo al sistema principal para que pueda dedicarse a tareas más críticas
- Algunas veces más preciso que otros métodos.

1.8.1.- Fuente de alimentación

Los RTCs a menudo tienen una fuente de alimentación alternativa, por lo que pueden seguir midiendo el tiempo mientras la fuente de alimentación principal está apagada o no está disponible. Esta fuente de alimentación alternativa es normalmente una batería de litio en los sistemas antiguos, pero algunos sistemas nuevos usan un supercapacitor, porque son recargables y pueden ser soldados. La fuente de alimentación alternativa también puede suministrar energía a una memoria no volátil.

1.8.1.1.- Batería De Litio [*17]

Figura 1.19 Batería de Litio CR2032 3 Voltios

La batería de iones de litio, también denominada batería Li-Ion, es un dispositivo diseñado para almacenamiento de energía eléctrica que emplea como electrolito, una sal de litio que procura los iones necesarios para la reacción electroquímica reversible que tiene lugar entre el cátodo y el ánodo.

[*17] http://es.wikipedia.org/wiki/Bater%C3%ADa_de_ion_de_litio

Las propiedades de las baterías de Li-ion, como la ligereza de sus componentes, su elevada capacidad energética y resistencia a la descarga, la ausencia de efecto memoria o su capacidad para operar con un elevado número de ciclos de regeneración, han permitido el diseño de acumuladores livianos, de pequeño tamaño y variadas formas, con un alto rendimiento, especialmente adaptados para las aplicaciones de la industria electrónica de gran consumo. Desde la primera comercialización a principios de los años 1990 de un acumulador basado en la tecnología Li-ion, su uso se ha popularizado en aparatos como teléfonos móviles, agendas electrónicas, ordenadores portátiles y lectores de música.

1.8.2.- Medición del tiempo

La mayoría de los RTCs usan un oscilador de cristal, pero algunos usan la frecuencia de la fuente de alimentación. En muchos casos la frecuencia del oscilador es 32.768 kHz. Ésta es la misma frecuencia usada en los relojes de cuarzo, y por las mismas razones, que la frecuencia es exactamente 2^{15} ciclos por segundo, que es un ratio muy práctico para usar con circuitos de contadores binarios simples.

1.8.3.- DS1307 [*18]

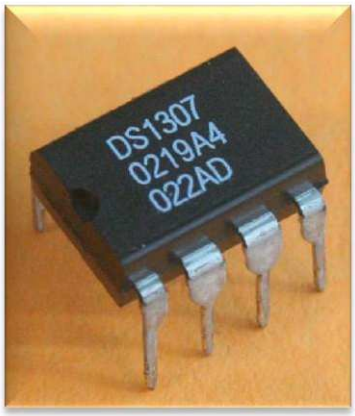


Figura1.14 Circuito DS1307

El DS1307 de Dallas Semiconductor (Maxim) es un dispositivo que se conoce como “Reloj de Tiempo Real” (Real Time Clock – RTC) que opera a través del

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

bus I2C y que, además de brindarnos la hora con minutos y segundos, posee un calendario que contempla los años bisiestos hasta fin de siglo, es decir, hasta el año 2100.

Entre las características destacadas, posee una salida (configurable por software), y la posibilidad de trabajar con una pequeña batería para almacenar los datos mientras el sistema se encuentra desconectado de la alimentación. Además, esta pequeña alimentación de respaldo permite mantener funcionando el oscilador maestro del reloj con un consumo ínfimo de 300nA, según su hoja de datos. Una simple batería de Litio CR2032 puede brindarnos un funcionamiento satisfactorio durante 10 años.

1.9.- LCD

Una pantalla de cristal líquido o *LCD* (sigla del inglés *liquid crystal display*) es una pantalla delgada y plana formada por un número de píxeles en color o monocromos colocados delante de una fuente de luz o reflectora. A menudo se utiliza en dispositivos electrónicos de pilas, ya que utiliza cantidades muy pequeñas de energía eléctrica. [*19]

En la figura 1.15 se muestra las partes y componentes de una pantalla LCD

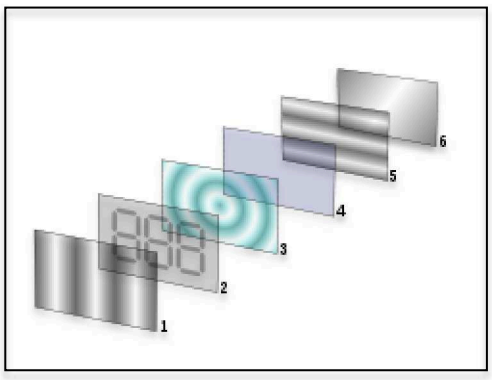


Figura 1.15 Partes de la pantalla del LCD

1. Film de filtro vertical para polarizar la luz que entra.
2. Sustrato de vidrio con electrodos de Óxido de Indio ITO. Las formas de los electrodos determinan las formas negras que aparecen

[*19] http://es.wikipedia.org/wiki/Pantalla_de_cristal_l%C3%ADquido

cuando la pantalla se enciende y apaga. Los cantos verticales de la superficie son suaves.

3. Cristales líquidos "Twisted Nematic" (TN).
4. Sustrato de vidrio con film electrodo común (ITO) con los cantos horizontales para alinearse con el filtro horizontal.
5. Film de filtro horizontal para bloquear/permitir el paso de luz.
6. Superficie reflectante para enviar devolver la luz al espectador. En un LCD retroiluminado, esta capa es remplazada por una fuente luminosa.

1.9.1.- Descripción.

Un LCD es un dispositivo Micro-Controlado de visualización grafica para la presentación de caracteres, símbolos o incluso dibujos (en algunos modelos)

Tiene 2 filas de 16 caracteres cada una y cada carácter dispone de una matriz de 5x7 puntos (pixels), aunque los hay de otro número de filas y caracteres.

Este dispositivo es manejado internamente por un microcontrolador Hitachi 44780 el cual regula todos los parámetros de presentación

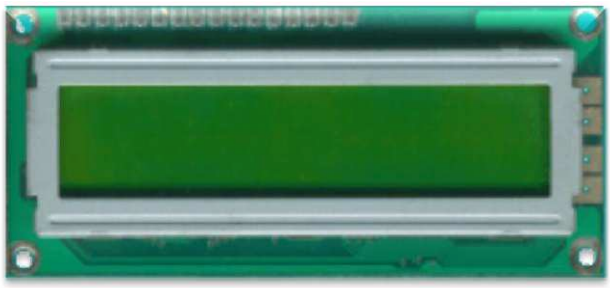


Figura1.15 LCD

Este display fue utilizado en el circuito de configuración de texto y hora.

1.9.2.- Características del LCD:

- Pantalla de caracteres ASCII, además de los caracteres Kanji y Griegos.
- Desplazamiento de los caracteres hacia la izquierda o la derecha.
- Proporciona la dirección de la posición absoluta o relativa del carácter.
- Memoria de 40 caracteres por línea de pantalla.

- Movimiento del cursor y cambio de su aspecto.
- Permite que el usuario pueda programar 8 caracteres.
- Conexión a un procesador usando un interfaz de 4 u 8 bits. (*20)

1.9.3.- Distribución de Pines

Tabla 1.4 Distribución de pines del LCD

PIN	Nombre	Nivel	Descripción
1	Vss	0 V	Tierra
2	Vcc	5 V	Alimentación
3	Vee	Pot	Potenciómetro (Contraste)
4	RS	Lógico	0L instrucción, 1L dato
5	R/W	Lógico	1L lee, 0L escribe en el LCD
6	E	Lógico	Pulso de habilitación
7 a 14	DB0-DB7	Lógico	BUS de datos

1.10.- PULSADORES (*21)

Figura 1.16 Pulsadores

Un botón o pulsador es un dispositivo utilizado para activar alguna función. Los botones son de diversa forma y tamaño y se encuentran en todo tipo de dispositivos, aunque principalmente en aparatos eléctricos o electrónicos. Los

*20 Apuntes de Control con Microprocesadores, Ingeniero Alcívar Costales
*21 http://es.wikipedia.org/wiki/Bot%C3%B3n_(dispositivo)

botones son por lo general activados al ser pulsados, normalmente con un dedo.

Un botón de un dispositivo electrónico, funciona por lo general como un interruptor eléctrico, es decir en su interior tiene dos contactos, que son: NA (normalmente abierto) o NC (normalmente cerrado), con lo que al pulsarlo se activará la función inversa de la que en ese momento este realizando.

Hay que tener en cuenta, a la hora de diseñar circuitos electrónicos, que la excesiva acumulación de botones, puede confundir al usuario, por lo que se tenderá a su uso más imprescindible.

También existen "botones virtuales", cuyo funcionamiento debe ser igual al de los "físicos"; su uso queda restringido para pantallas táctiles o gobernadas por otros dispositivos electrónicos

1.11.- TECLADO MATRICIAL [*22]

Figura 1.17 Teclado matricial 4x4

Un teclado matricial es un simple arreglo de botones conectados en filas y columnas, de modo que se pueden leer varios botones con el mínimo número de pines requeridos. Un teclado matricial 4×4 solamente ocupa 4 líneas de un puerto para las filas y otras 4 líneas para las columnas, de este modo se

[*22] http://www.rockbotica.com/esp/index/item/92/teclado-matricial-4x4-de-membrana

pueden leer 16 teclas utilizando solamente 8 líneas de un microcontrolador. Si asumimos que todas las columnas y filas inicialmente están en alto (1 lógico), la pulsación de un botón se puede detectar al poner cada fila a en bajo (0 lógico) y monitorear cada columna en busca de un cero, si ninguna columna está en bajo entonces el 0 de las filas se recorre hacia la siguiente y así secuencialmente.

1.11.1.- Funcionamiento.

El principio de funcionamiento se basa en enviar mediante 4 pines del puerto (los cuatro primeros o los cuatro últimos) una combinación y leer los siguientes 4 pines para ver si hubo algún cambio en la lectura, si no lo hubo nos indica que no se presionó tecla alguna, pero si hubo un cambio hay que rastrear la tecla que se presionó mediante un "código".

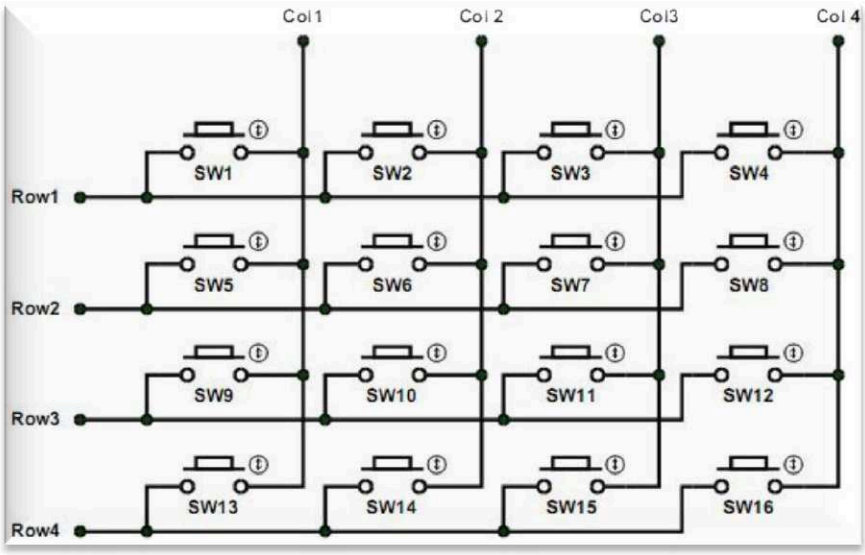


Figura 1.18 Diagrama del teclado matricial 4x4

Por ejemplo primero colocamos el valor 1111 en "Row" y luego colocamos el valor 0111 en "Col", si no existiese ninguna tecla pulsada la lectura que obtenemos en "Row" es 1111 pero si se pulsará por ejemplo la tecla SW12 el valor de la lectura en "Row" sería 1011 y de esta forma podemos obtener el código 01111011 que le correspondería a la tecla SW12.

1.12.-LENGUAJE DE PROGRAMACIÓN BASCOM

El lenguaje de programación BASCOM, es un lenguaje similar al Basic, es un lenguaje de alto nivel, el cual posee estructuras de programación for, while, do, loop, if, then, else, select, case, creación de variables con signo y sin signo, de longitud tipo byte, word, dword, precisión simple y flotante, además posee una interfaz para la simulación del programa realizado.

1.12.1.-Instrucciones más utilizadas del Bascom

✪ **Selección del micro controlador**

$regfile = "m16def.dat"

✪ **Stack**

Definición del STACK (PILA) hardware y software, que define el nivel de anidamiento de subrutinas e interrupciones.

$hwstack = 50
$swstack = 50
$framesize = 50

✪ **Definición del valor del cristal interno o externo utilizado.**

$crystal = 8000000

✪ **Configuración de un LCD alfanumérico a 4 bits**

Config Lcdpin = Pin, Db4 = Portc.4, Db5 = Portc.5, Db6 = Portc.6, Db7 = Portc.7, E = Portc.3, Rs = Portc.1, Wr = Portc.2

Config Lcd = 16 * 2

Para la configuración de un lcd alfanumérico se selecciona los pines conectados del micro-controlador y datos del lcd (DB4 a DB7 en la interfaz a 4 bits), y los pines de control del lcd (Rs, WR, En) asociados con el micro-controlador.

✪ **Variables**

Dim Nombre variable(x) As tipo

Para definición de variables se debe seleccionar un nombre y el tipo de la variable (bit, byte, Word, etc) y además de ser el caso la longitud x si se trata de un arreglo.

✪ **Alias**

Nombre alias puerto o variable (F1 Alias Pina.7)

Los alias sirven para facilitar la legibilidad del programa y facilitar el cambio del código en caso de que el hardware se encuentre distribuido o asignado a otros pines o variables.

✪ **Constantes**

Const Nombre de la constante=valor (Const Asterisco = 10)

Para la declaración de constantes se debe anteponer const luego seleccionar el nombre de la constante y luego escoger el valor deseado.

✪ **Interrupciones.**

Definición de interrupciones, para definir interrupciones se debe seleccionar la fuente de interrupción, el nombre de la etiqueta para el salto de la interrupción.

```
Config Timer0 = Timer, Prescale = 64
On Ovf0 Tim1_isr
……..
……..
Código
……..
…….
Tim1_isr:
        Código de la interrupción
        …….
        ……
return
```

En presencia de una interrupción del timer0 este ejecutara el código encontrado después de la etiqueta Tim1_isr:

✪ **Subrutinas**

Para utilizar subrutinas se debe realizar los siguientes:

```
Declare Sub Nombre (var1, var2,…)
…….
……
……
……
Sub Nombre (var1, var2,…)
        Código
        ……..
End Sub
```

Para la utilización de subrutinas se debe primero declarar la subrutina por facilidad en la parte superior del programa con la palabra declare.

✪ **Funciones**

La diferencia con las subrutinas para este caso las funciones devuelven un valor x especificado y su formato es el siguiente:

```
Declare Function Nombre (var1, var2,…) as tipo
…….
……
……
……
Declare Function Nombre (var1, var2,…) as tipo
        Código
        ……..
        Nombre=x ‘(valor de retorno)
End function
```

Para la función se debe seleccionar el tipo de variable de retorno (bit, byte, Word, etc.)

1.12.2.-Compilación.

Una vez realizado se procede a compilar el programa realizado presionando la tecla F7 y de no existir ningún error generara el código en formato .bin para ser cargado en el micro-controlador. La Figura 1.20 muestra el progreso de compilación.

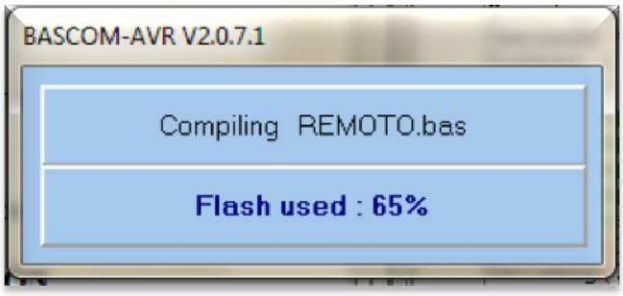


Figura 1.20Progreso de compilación

1.12.3.-Simulación.

Una vez corregido y compilado el programa, para depurar el programa se puede utilizar el simulador del Bascom presionando la tecla de función F2. La Figura 1.21 muestra la interfaz para la simulación.

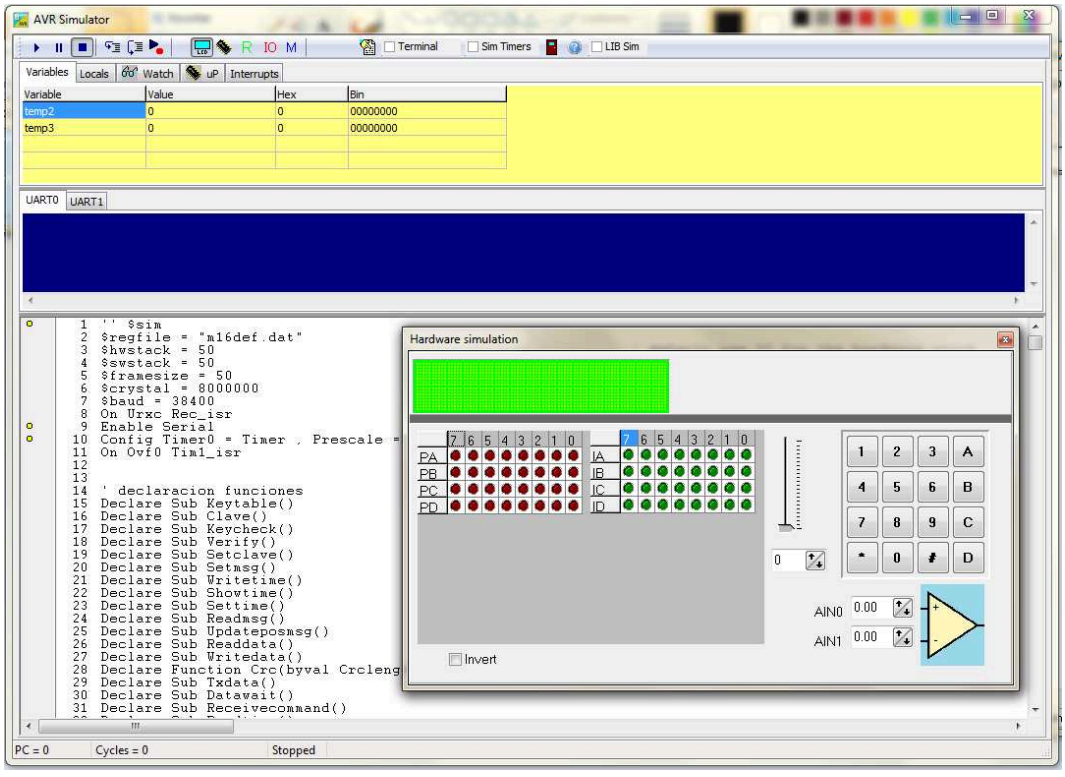


Figura 1.21 Pantalla de simulación.

Para depurar el programa paso a paso se debe presionar la tecla de función F8 y para ejecutar el programa por encima de subrutinas se presiona las teclas SHIFT+F8.

CAPÍTULO II

2.1.- DESCRIPCIÓN DEL SISTEMA.

2.1.1.- INTRODUCCIÓN

El siguiente capítulo explica y describe los componentes de cadacircuito que forman parte del sistema.

El sistema recibe alimentación de un banco de baterías de 24 voltios provenientes del sistema eléctrico del bus, la matriz de led's tiene incorporado un regulador de voltaje de 5 voltios que sirve para la alimentación de todos los circuitos del sistema.

El circuito configurador de texto y hora cuenta con un micro-controlador, el ATMEGA16, el mismo que se encarga de guardar en memoria los mensajes que se puedan ingresar a través de un teclado matricial, el texto será modificado a voluntad del usuario con la ayuda de una pantalla LCD 2X16 en el que se podrá visualizar los cambios efectuados al texto y a la hora. Con ayuda de pulsadores ubicados de manera estratégica (teclas de navegación) será posible cambiar de un mensaje a otro y moverse con facilidad dentro del texto.

Mediante el uso de la interfaz podemos comunicar al circuito configurador de texto y hora, con el circuito controlador de la matriz de led's, logrando enviar la información almacenada por el usuario en el micro-controlador a la matriz para que esta sea visible.

En el circuito controlador de la matriz de led's encontramos otro micro-controlador encargado del barrido de led's, y se recibe la información enviada por el circuito de configuración de texto y hora. Se adicinó un reloj en tiempo real para poder mostrar la hora actual en la matriz, la memoria EEPROM almacena los mensajes que se enviarán a la matriz.

En la figura 2.1 se presenta un diagrama de bloquen en el que se detalla la configuración utilizada para el funcionamiento adecuado de los circuitos del sistema.

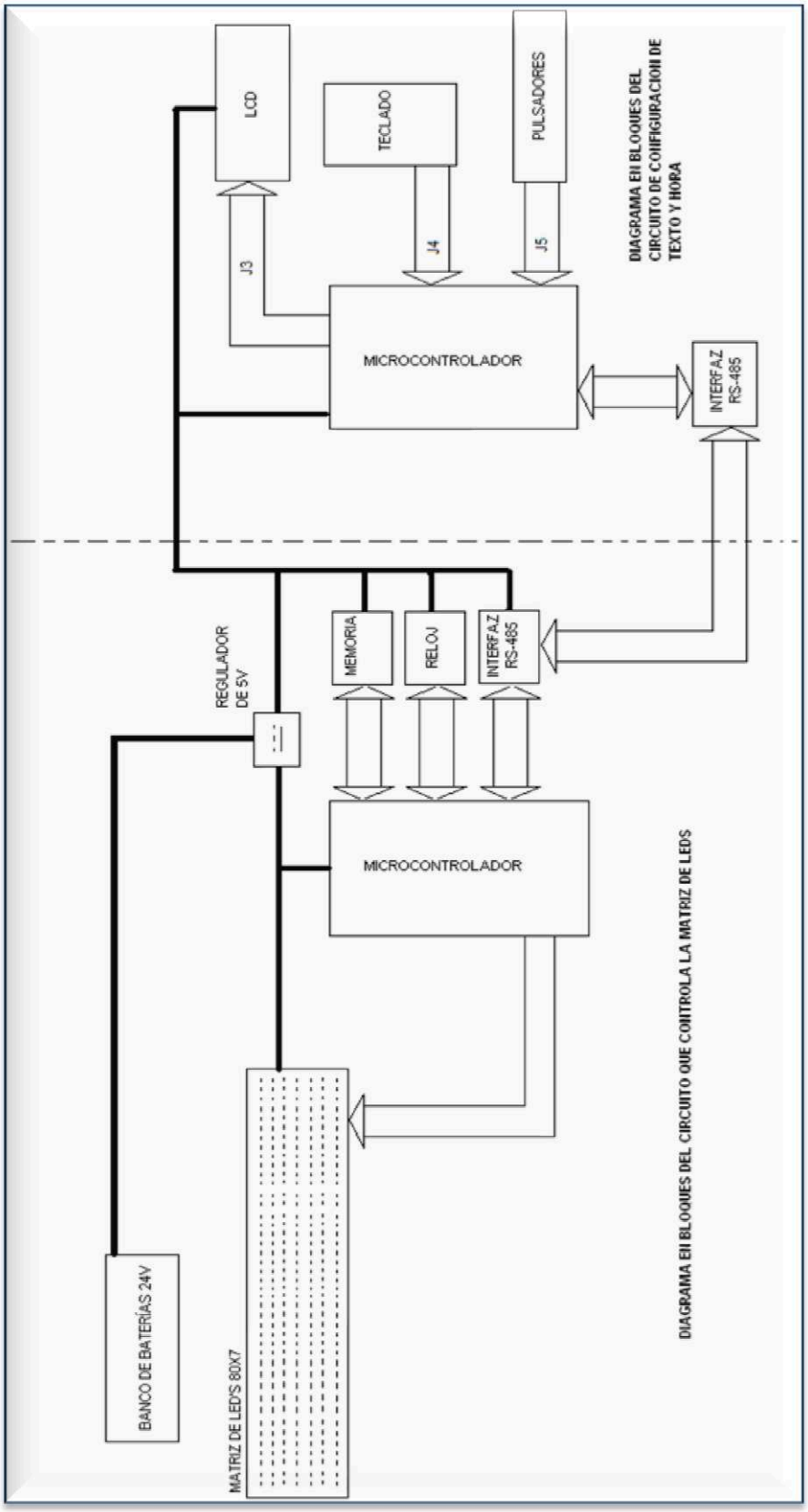


Figura 2.1. Diagrama de bloques del circuito.

Para poder controlar la matriz de led's 7x80 se necesitan 2 circuitos, el primero que controle la matriz de led's y el segundo para la configuración de texto y hora.

2.1.2.- ALIMENTACIÓN DEL CIRCUITO

La alimentación proviene de 2 baterías de 12 voltios conectadas en serie, las cuales suman 24 voltios, las mismas que son conectadas con todo el sistema eléctrico del bus.

La matriz consta de un regulador interno de 5 voltios el mismo que polariza la matriz de led's, el circuito que controla a la matriz de led's y el circuito de configuración de texto y hora.

2.1.3.- Circuito de configuración de texto y hora.

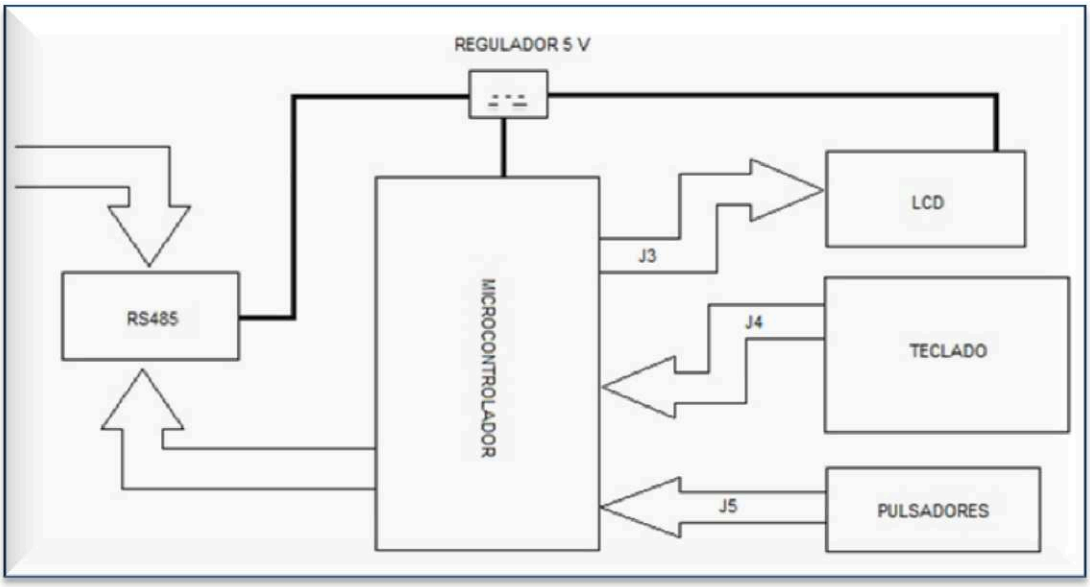


Figura 2.2.-Diagrama de bloques del circuito de configuración de texto y hora.

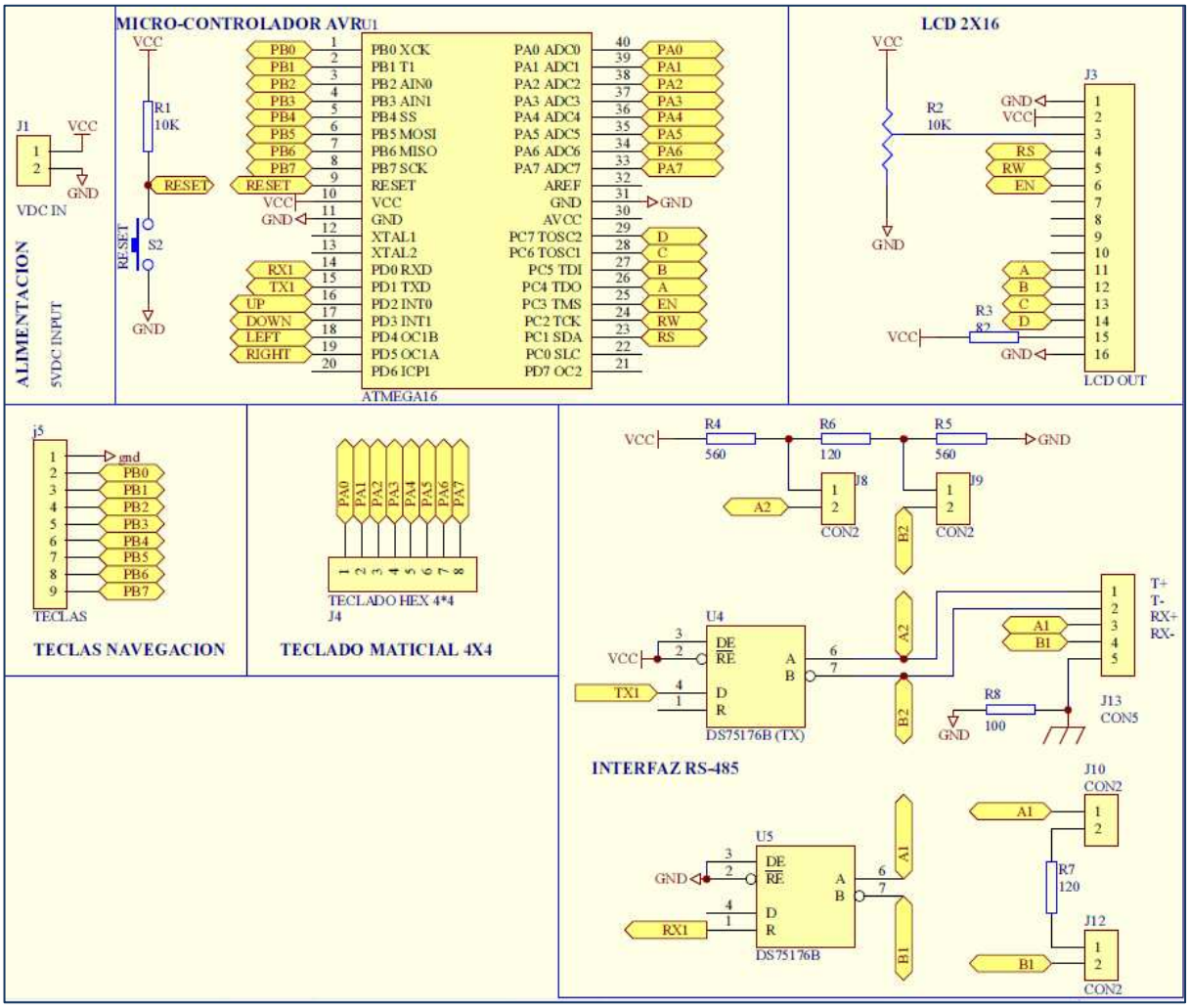


Figura 2.3 Diagrama del circuito de configuración de texto y hora.

El circuito para la configuración de mensajes y hora está constituido por las siguientes partes.

2.1.3.1.- Micro-controlador AVR.

El ATMEGA16 (U1), se encarga de enviar mensajes a la pantalla LCD 2x16, enviar y recibir datos mediante la interfaz RS-485, además de supervisar el estado de los pulsadores de navegación y el estado del teclado matricial 4x4.

2.1.3.2.- Teclas de Navegación.

Las teclas de navegación (pulsadores) deben ir al conector J5, las cuales se utiliza para ir navegando entre el mensaje, hora de salida, el número de mensaje, y enviar el mensaje a la matriz de LEDS para que se almacene en la memoria eeprom. El valor lógico de cada tecla sin presionar es 1L (5voltios) y 0L (0 voltios) cuando estos se encuentran presionados, para lo cual se debe

activar mediante programación las resistencias internas PULL-UPS[23] asociadas al puerto conectado las teclas de navegación.

2.1.3.3.- Teclado Matricial.

El teclado matricial 4x4 utiliza el conector J4, el cual se encuentra asociado al PUERTO A del micro-controlador. La tabla 2.1 muestra las filas y columnas asociadas a dicho puerto.

Tabla 2.1. Asociación teclado matricial al puerto a del ATMEGA16.

Teclado	Puerto PIN	Entrada	Salida
FILA 1	PINA.7	SI	-
FILA 2	PINA.6	SI	-
FILA 3	PINA.5	SI	-
FILA 4	PINA.4	SI	-
COLUMNA 1	PORTA.3	-	SI
COLUMNA 2	PORTA.2	-	SI
COLUMNA 3	PORTA.1	-	SI
COLUMNA 4	PORTA.0	-	SI

Para el caso de las filas el puerto debe configurarse como entradas y además habilitar las resistencias pull-ups. En la tabla 2.2 se puede apreciar las teclas asociadas a las filas y columnas.

Tabla 2.2. Distribución del teclado matricial 4x4 utilizado.

	COLUMNA 1	**COLUMNA 2**	**COLUMNA 3**	**COLUMNA 4**
FILA 1	1	2	3	A
FILA 2	4	5	6	B
FILA 3	7	8	9	C
FILA 4	*	0	#	D

[23] Las resistencias PULL-UPS, son resistencias internas del microcontrolador

2.1.3.4.-Pantalla Alfanumérica LCD 2x16.

La pantalla matricial de cristal líquido (LCD), se utiliza para mostrar el destino, la hora de salida, además de contar con la posibilidad de observar la hora actual.

Para controlar el contraste se utiliza el potenciómetro R2, y para limitar la luz de fondo la resistencia R3.

2.1.3.5.- Interfaz RS-485.

Para el envío y recepción de datos se utiliza la interfaz RS-485 a cuatro hilos mediante los integrados U4 y U5 (DS75176B). La interfaz serial al igual que el caso del circuito que manipula la matriz de LEDS debe poseer las mismas características es decir:

Velocidad de transmisión: 38400
Bits de inicio: 1
Bits de parada: 1
Paridad: Ninguna
Longitud de caracteres: 8.

2.1.4.- Circuito que controla la matriz de led's.

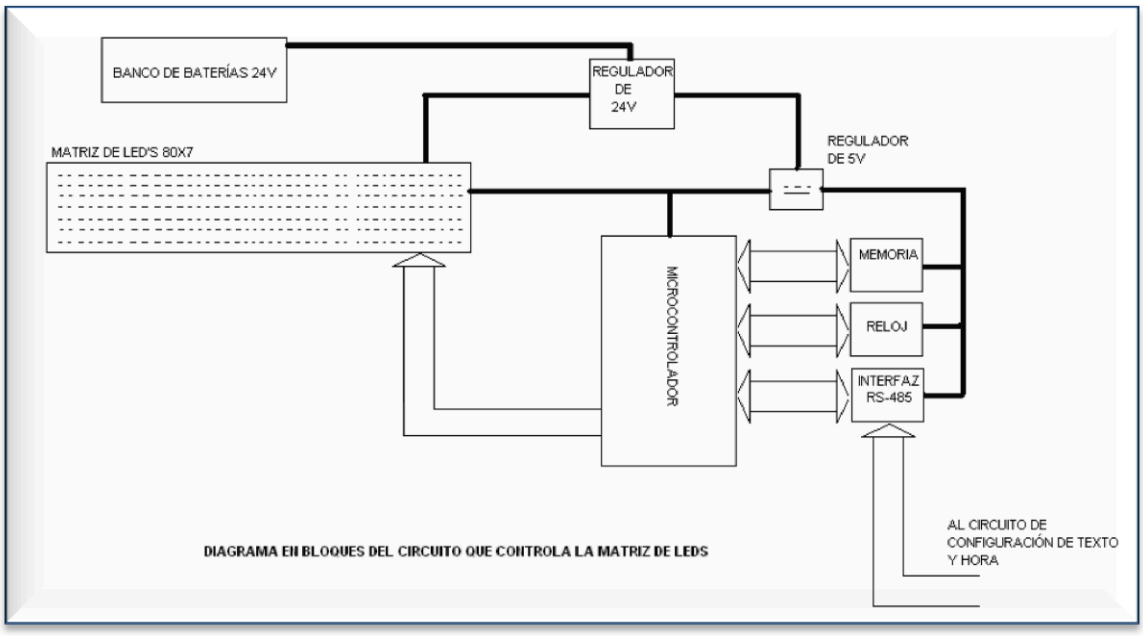


Figura 2.4.- Diagrama en bloques del circuito que controla la matriz de led's.

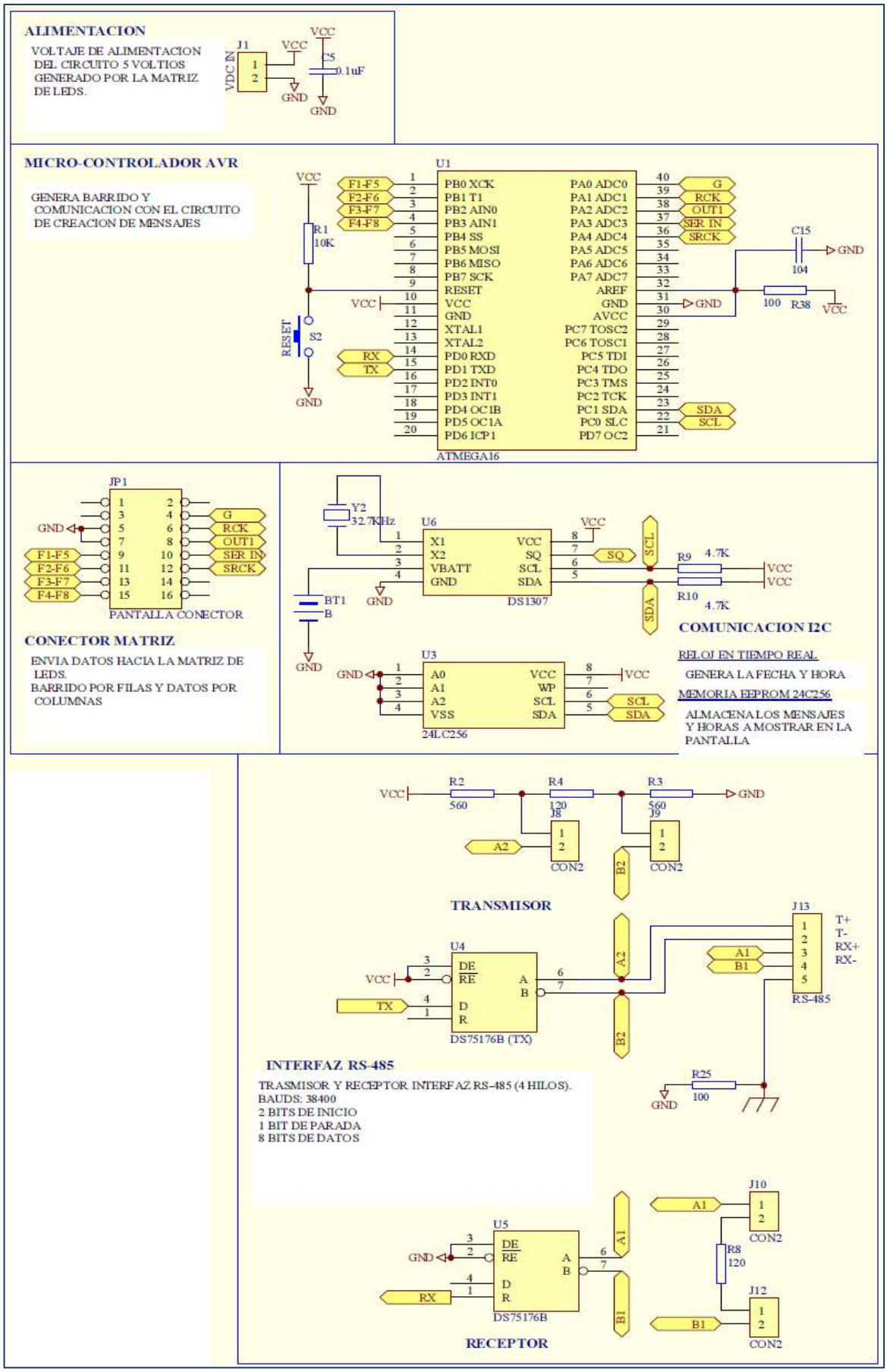


Figura 2.5 Diagrama circuito que controla la matriz de LEDS.

2.1.4.1.- Micro-controlador AVR ATMEGA16.

El micro-controlador se encarga de controlar el barrido por filas y datos de la matriz de leds por columnas, el envió de datos de las columnas se lo realiza mediante integrados “shift-registers” (convertidor de dato serial a paralelo), además se comunica con el circuito de configuración de mensajes y ajuste del reloj mediante una interfaz RS-485 bidireccional

2.1.4.2.- Integrados con Comunicación I2C.

Se utiliza un **reloj en tiempo real** (U6 DS1307) para leer la hora y fecha, además posee una **batería de litio de 3v** (BT1) para que el reloj siga en funcionamiento aún en caso de ausencia de voltaje. Para almacenar los mensajes se utiliza la **memoria eeprom 24c256 (U3).**

2.1.4.3.- Interfaz RS-485.

La comunicación con el módulo de configuración se lo realiza mediante una interfaz RS-485 a 4 hilos (TX+, TX-, RX+, RX-), la cual posee las siguientes características:

Velocidad de transmisión: 38400
Bits de inicio: 1
Bits de parada: 1
Paridad: Ninguna
Longitud de caracteres: 8.

2.2.- PROGRAMACIÓN DE LOS MICROCONTROLADORES.

2.2.1.- PROGRAMACIÓN DEL CIRCUITO DE CONTROL DE LA MATRIZ DE LEDS (VER ANEXO 1).

El programa para el circuito que manipula la matriz de LEDS debe realizar las siguientes acciones:

- ✪ Leer, enviar y procesar los datos provenientes de la interfaz RS-485 (Mensajes y hora).

- ✪ Generar el barrido por filas de la matriz de LEDS.

- ✪ Generar el desplazamiento serial de los datos a mostrar en la matriz de LEDS.

- ✪ Realizar la lectura y escritura (en caso de ser necesario) en la memoria eeprom.

- ✪ Realizar la lectura y escritura (en caso de ser necesario) en el reloj en tiempo real.

- ✪ Realizar la comprobación de la hora de salida con la hora actual.

- ✪ Leer el mensaje actual de la memoria eeprom.

- ✪ Con las acciones requeridas el diagrama de flujo del programa realizado se muestra en la Figura 2.6.

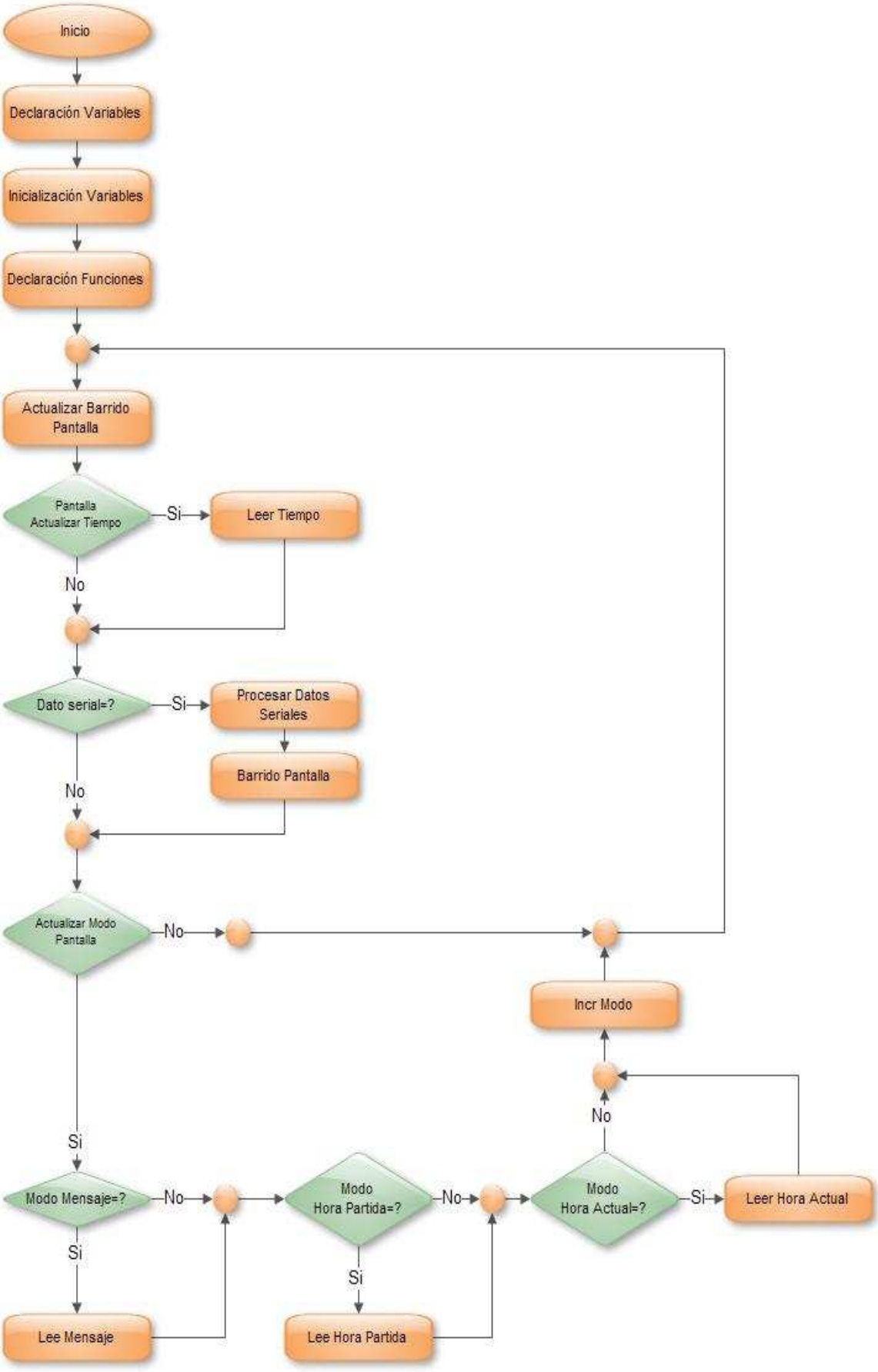


Figura 2.6. Diagrama de flujo del circuito que controla la Matriz de LEDS.

2.2.2.- PROGRAMA DEL CIRCUITO DE CONFIGURACIÓN DE TEXTO Y HORA (VER ANEXO 2).

El programa debe leer, enviar y procesar los datos provenientes del circuito que manipula la matriz de LEDS, es decir:

- ✪ Leer el teclado matricial y teclas de navegación.
- ✪ Mostrar datos en la pantalla LCD según sea el caso
- ✪ Actualizar fecha y hora
- ✪ Actualizar destino y hora de salida
- ✪ Leer y verificar clave (agregado por seguridad).
- ✪ Con lo expuesto el diagrama de flujo del programa realizado es el siguiente.

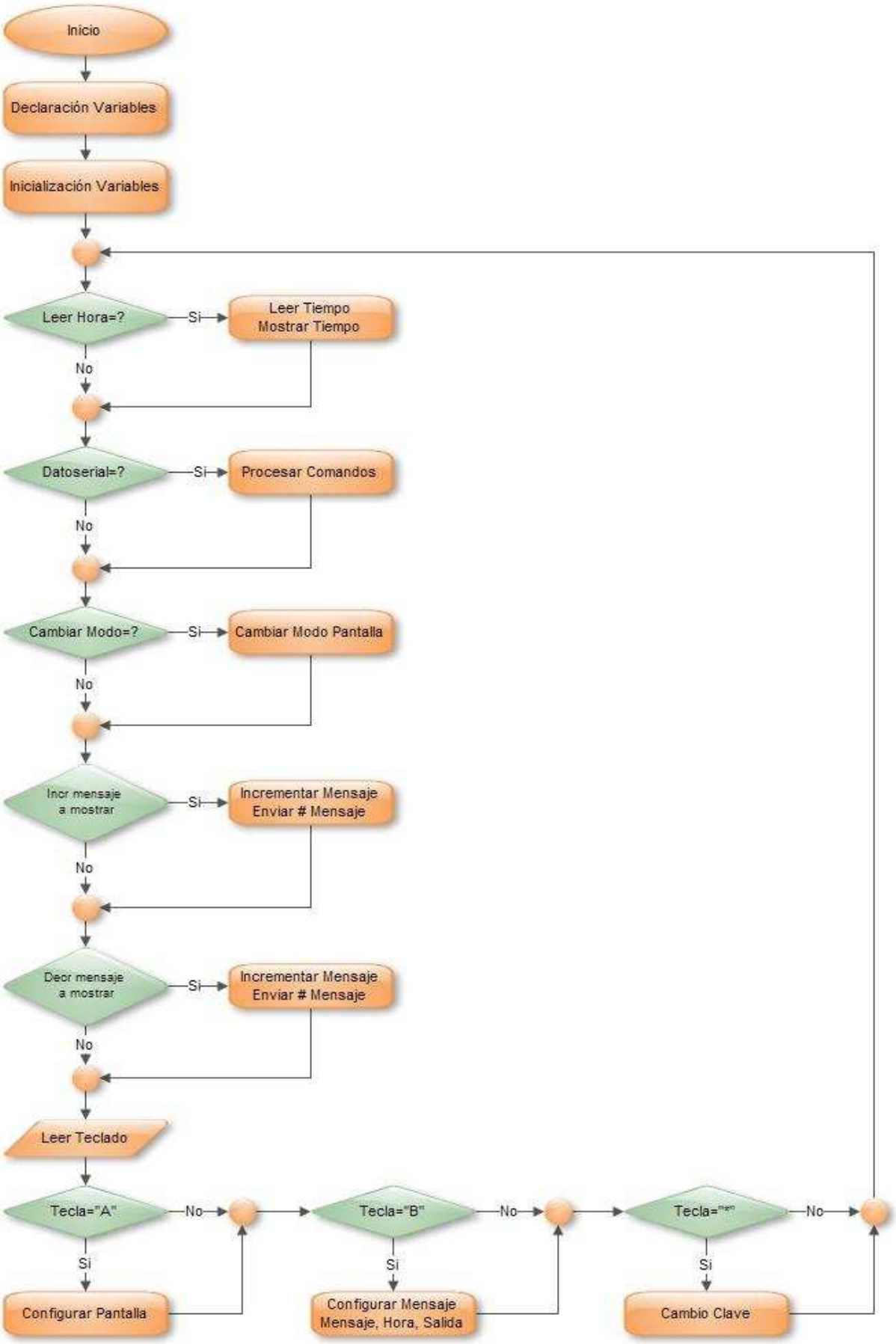
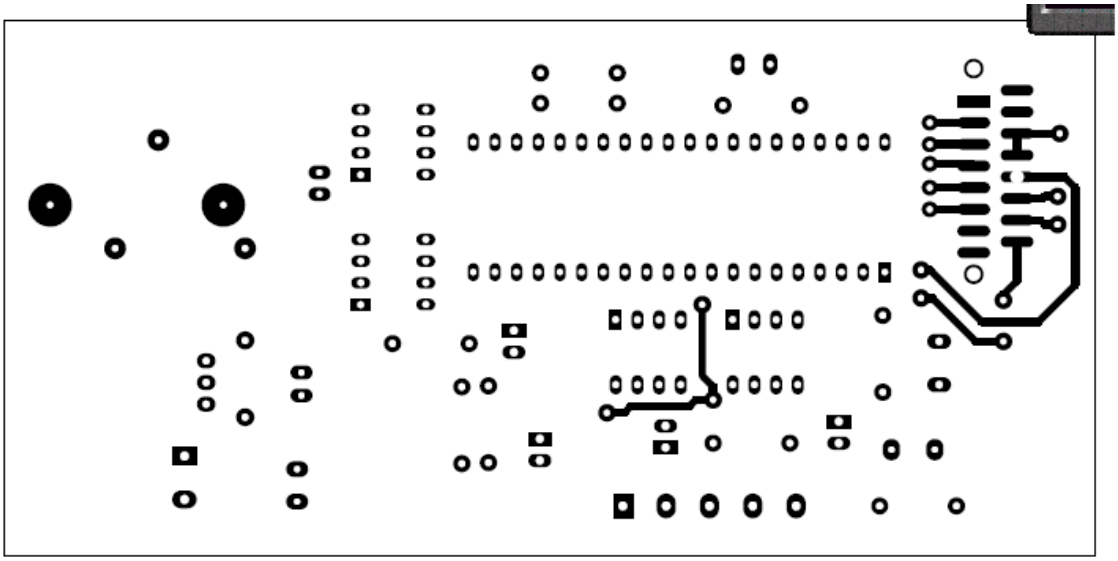


Figura 2.7. Diagrama de flujo para el circuito de configuración de texto y hora.

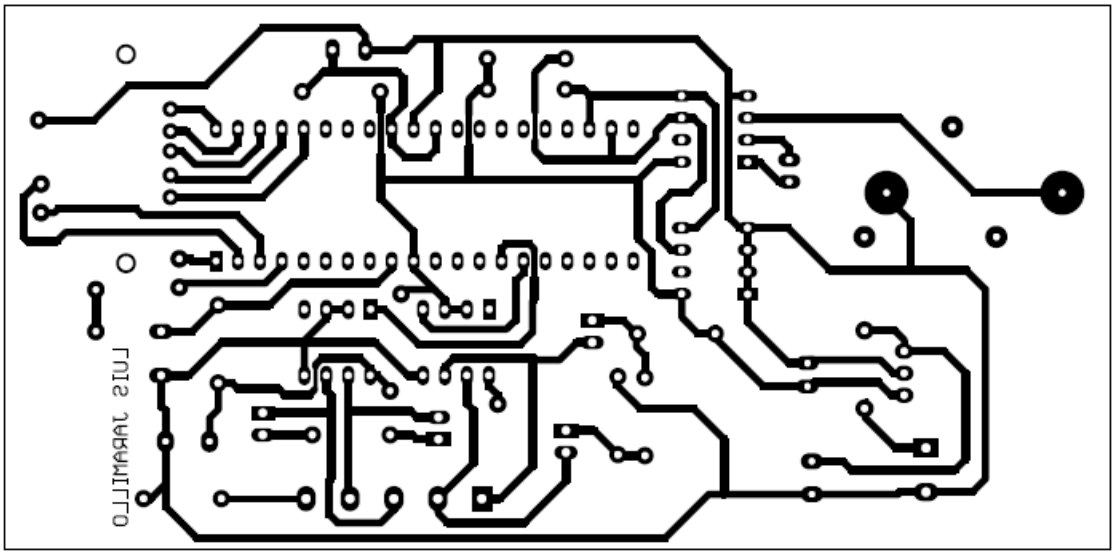
2.3.- ENSAMBLAJE DE LOS CIRCUITOS UTILIZADOS.

2.3.1.- CIRCUITO QUE CONTROLA LA MATRIZ LED'S.

La Figura 2.8 muestra las pistas de la tarjeta diseñada.



(a)



(b)

Figura 2.8. Pistas tarjeta control matriz de LEDS (a). Cara Superior, (b). Cara Inferior.

La Figura 2.9 muestra la posición de los elementos de la tarjeta diseñada.

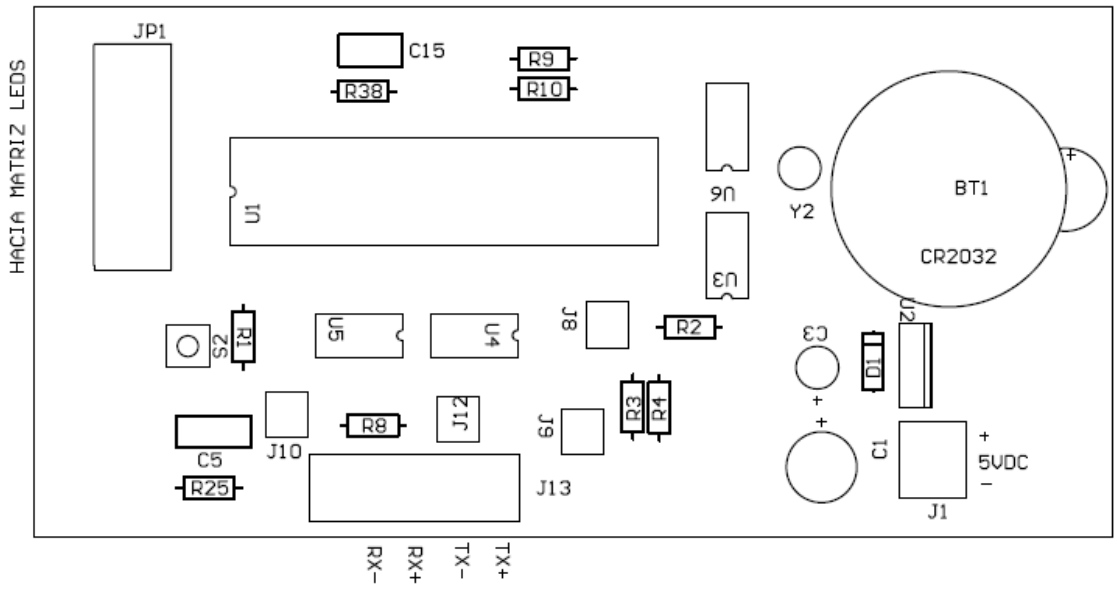


Figura 2.9 Posición elementos de la tarjeta de control de la matriz de leds.

La Figura 2.10 muestra la tarjeta realizada

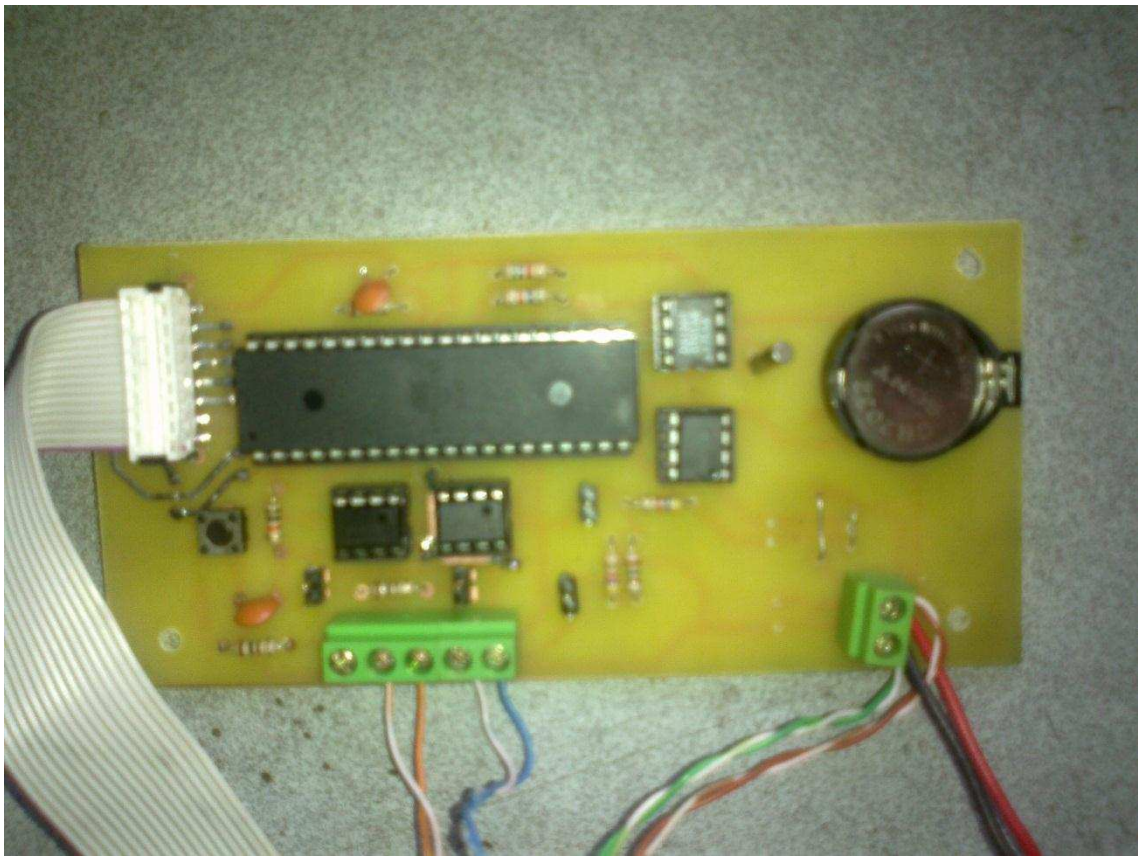


Figura 2.10. Tarjeta control matriz leds.

Con lo que se puede observar en la Figura 2.9 las conexiones son las siguientes:

- ✪ J1: alimentación 5 Vdc
- ✪ J13: interfaz RS-485.
- ✪ JP1: conector hacia la matriz de leds (datos filas y columnas).

2.3.2.- CIRCUITO DE CONFIGURACIÓN DE TEXTO Y HORA.

La Figura 2.11 muestra las pistas de la tarjeta realizada del circuito de configuración.

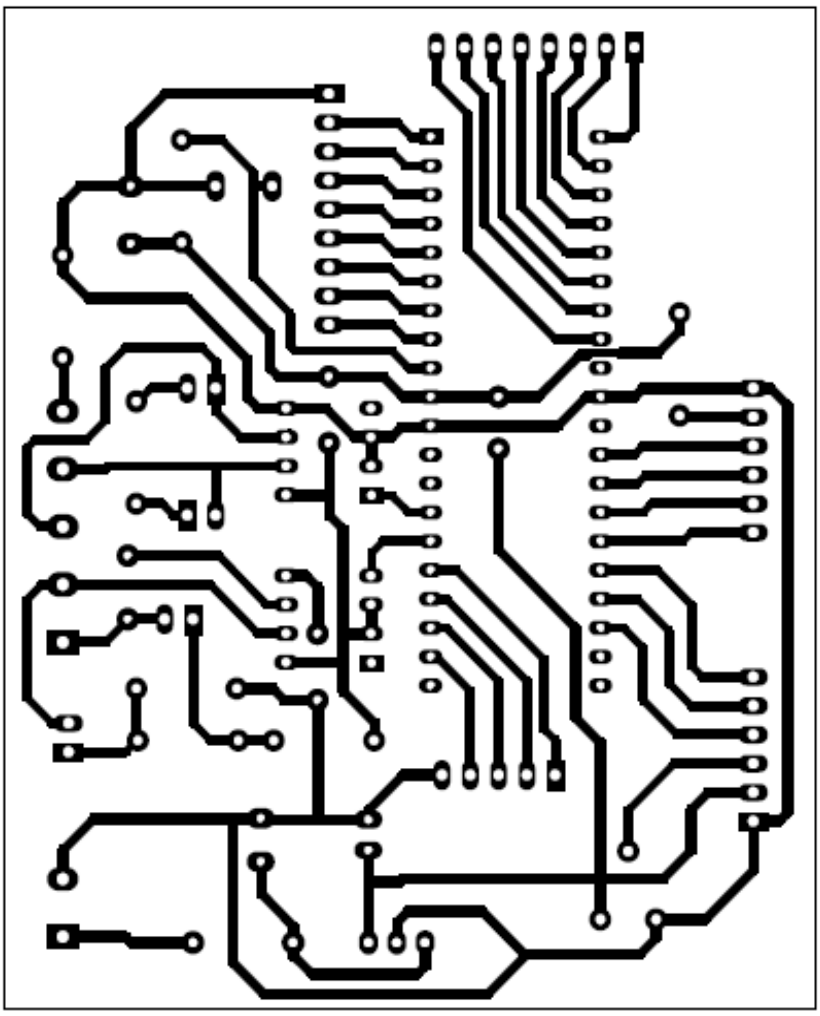


Figura 2.11. Pistas Tarjeta de configuración.

La Figura 2.12 muestra la tarjeta de configuración realizada.

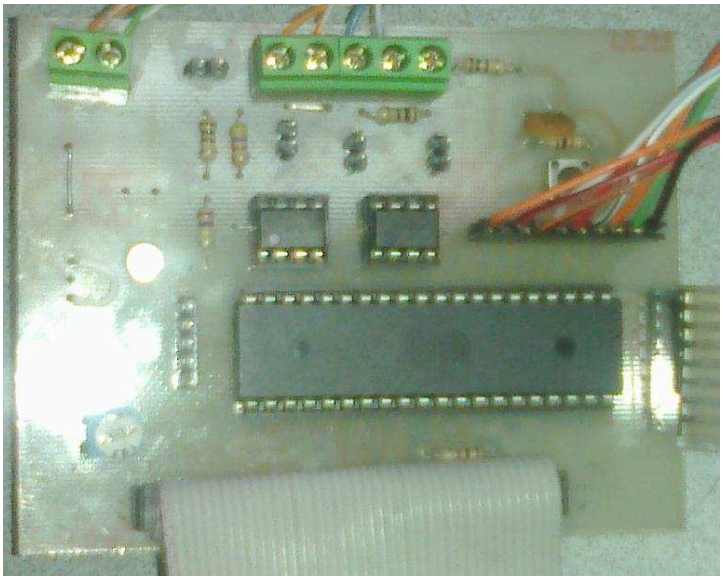


Figura 2.12 Tarjeta realizada del módulo de configuración hora y mensajes.

La Figura 2.13 muestra la posición de los elementos y las conexiones para la pantalla LCD, el teclado y la interfaz RS 485.

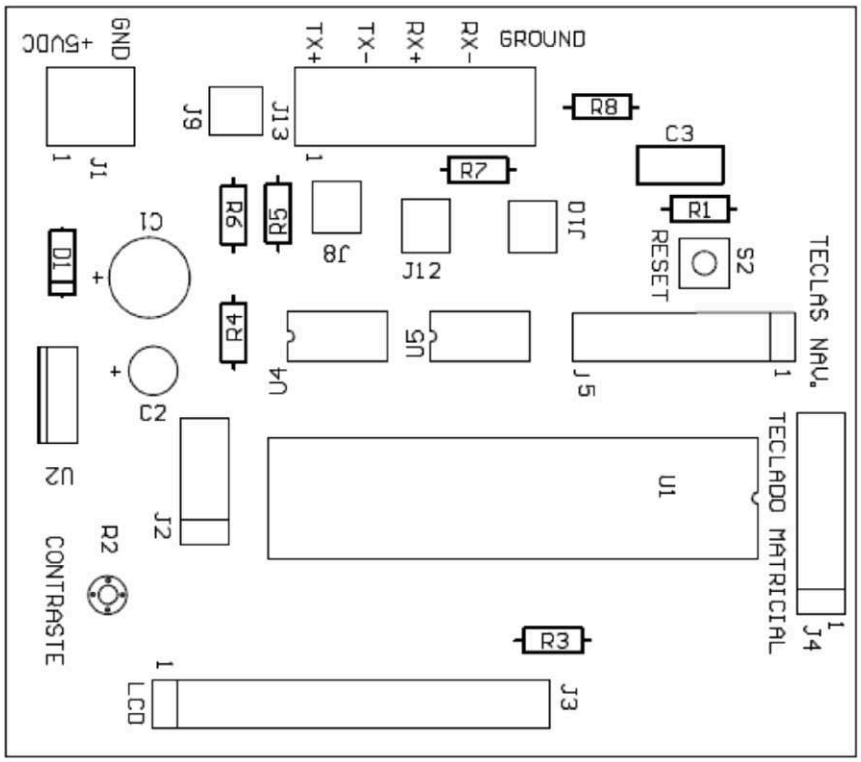


Figura 2.13. Posición elementos y conexionado de la tarjeta de configuración.

Las conexiones par el circuito de configuración son:

- ✪ J1: alimentación 5 Vdc.
- ✪ J13: conexión interfaz RS-485
- ✪ J3: Pantalla LCD 2*16
- ✪ J5: Teclas de navegación
- ✪ J4: Teclado matricial.

2.3.3.- ENSAMBLAJE DE LOS CIRCUITOS UTILIZADOS.

A continuación se detalla el ensamblaje de los circuitos realizados para la matriz de LEDS y para la configuración.

2.3.3.1.- Comunicación Serial (Interfaz RS-485)

Para la comunicación Serial se tiene que realizar la conexión en las borneras J13 entre las dos tarjetas (control matriz-tarjeta configuración), lo cual se muestra en la Tabla 2.3.

Tabla 2.3. Conexión comunicación serial.

Tarjeta Matriz (Conector J13)	**Tarjeta configuración (Conector J13)**
TX+	RX+
TX-	RX-
RX+	TX+
RX-	TX-

2.3.3.2.- Fuente de alimentación.

La fuente de alimentación como explicamos anteriormente es proporcionada por las baterías del bus, la matriz regula los 24V a 5 voltios de corriente continua para alimentar la tarjeta de control de la matriz (conector J1), con su respectiva polaridad indicada en la Figura 2.9, y conector (J1) de la Figura 2.10.

2.3.3.3.- Teclado.

El teclado matricial 4x4 debe conectarse al terminal J4 de la Figura 2.13

2.3.3.4.- Pantalla LCD.

La pantalla alfanumérica LCD 2x16, debe conectarse al terminal J3 de la Figura 2.13, y debe tener especial precaución ya que deben coincidir el pin 1 del conector y de la pantalla LCD, caso contrario puede dañarse la pantalla LCD:

Con lo expuesto anteriormente a continuación se muestra en la Figura 2.14.el ensamblaje de la tarjeta de configuración.

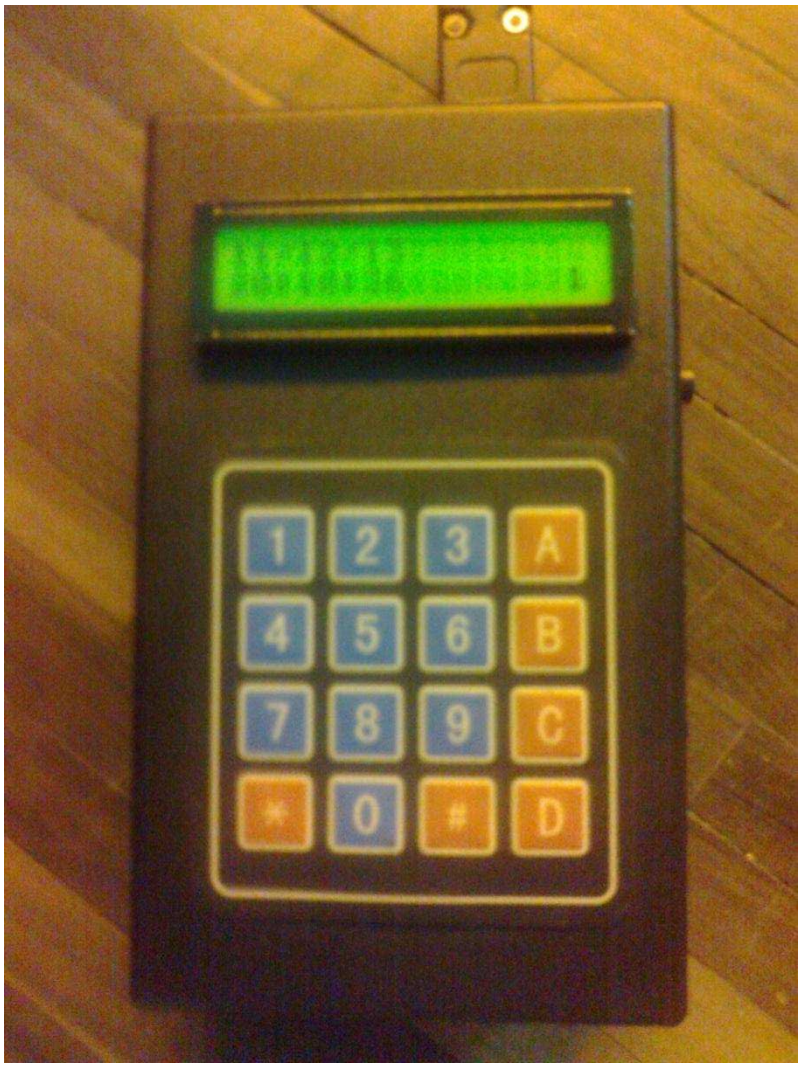


Figura 2.14 Ensamblaje de la tarjeta de configuración.

La tarjeta de control de la matriz de LEDS ensamblada se muestra en la Figura 2.15, la cual se encuentra instalada en el interior de la matriz de LEDS.

Figura 2.15. Ensamblaje Tarjeta de control de la matriz de LEDS.

La Figura 2.16 muestra la construcción final del módulo de configuración y de la matriz de LEDS.

Figura 2.16 Ensamblaje final del proyecto.

2.4.- PRUEBAS REALIZADAS.

A continuación se muestra en la Figura 2.17 el destino en la matriz de LEDS previamente almacenado.

(a)

(b)

(c)

Figura 2.17 (a) Mensaje destino, (b) hora de salida, (C) hora actual

Según lo que se puede observar en la Figura 2.17 la intensidad luminosa irradiada por los LEDS es suficientemente clara para que pueda ser visualizada tanto en la noche como en el día.

2.5.- MANUAL DE USUARIO.

Para manipular, cambiar o almacenar un nuevo mensaje; cambiar la hora de la matriz se lo realiza por medio del módulo de configuración, lo cual se detalla a continuación.

2.5.1.- Cambio de la hora del sistema.

Para realizar el cambio de la hora del sistema se debe presionar la tecla A y luego ingresar la clave de 4 dígitos. Al ingresar la clave correcta aparece la fecha (día/mes/año) y la hora actual (formato 24 horas).

Para incrementar el día presione la tecla A

Para cambiar entre fecha y hora se debe presionar la tecla B

Presione las teclas de 0 a 9 nueve para cambiar la hora o fecha.

Presione la tecla D para actualizar la nueva fecha y hora.

Presione la tecla C para retorna al menú principal sin actualizar la hora y fecha.

2.5.2.- Cambio del mansaje de destino y la hora de salida.

Desde el menú principal presione la tecla B y luego digite la clave de ingreso

Para cambiar entre mensaje, número de mensaje y salida presione las teclas ARRIBA o DERECHA.

2.5.2.1.-Ingreso de texto en la posición de mensaje

Si se encuentra en la posición para cambiar el mensaje presione las teclas de 0 a 9 varias veces hasta alcanzar la letra deseada o numero deseado.

Para cambiar entre letras mayúsculas o minúsculas presione la tecla *

Para moverse por caracteres del mensaje presione la tecla Derecha o izquierda

2.5.2.2.- Ingreso del número de mensaje.

Para cambiar el número de mensaje presione las teclas ARRIBA o ABAJO repetidamente hasta alcanzar la posición de cambio de numero de mensaje, y luego presione la teclas Derecha o izquierda para incrementar o decrementar el número de mensaje.

2.5.2.3.- Ingreso de la hora de salida

Para ir a la posición de hora de salida presione repetidamente las teclas ARRIBA o ABAJO.

Para moverse a través de la posición de la hora de salida presione las teclas DERECHA o IZQUIERDA

Presione las teclas numéricas 0-9 para cambiar el valor de la hora de salida según sea la posición de la hora o minutos de salida.

Para retornar al menú anterior presione la tecla C.

2.5.3.-Cambio de clave.

Para cambiar la calve actual presione la tecla *, luego presione la tecla actual y despuésdigite las teclas de 0-9 de la nueva clave.

CAPÍTULO III

3.1.- CONCLUSIONES.

- Se concluye que con la construcción del control para la cartelera digital facilita a los usuarios de transporte público interprovincial la introducción de mensajes como el destino, hora de salida y la hora actual en un letrero luminoso.

- En conclusión se diseñó un circuito capaz de almacenar en memoria hasta 10 mensajes, fácil de manipular y con capacidad de escribir hasta 16 caracteres.

- Se concluye que el tipo de LED'sutilizados son de alto brillo y un ángulo de dispersión mínima de 60 grados (para evitar la necesidad de poner un difusor), con los cuales se obtiene una muy buena visibilidad tanto en el día como en la noche.

- Se concluye que la utilización de la interfaz RS-485 garantiza una recepción y transmisión de datos a corto y largo alcance sin interferencias ni pérdidas de la información.

- Con la utilización de una batería de litio en el reloj en tiempo real se garantiza con ello que el reloj no se va a desigualar aún si el circuito se encuentra apagado y desconectado

3.2.- RECOMENDACIONES.

- Es recomendable que cuando se des actualice la hora y la fecha revisar la batería de litio ubicada internamente en la caja que contiene el la matriz de led's.

- Se recomienda tener cuidado con el cable de conexión (evitar aplastamientos) entre la matriz de LED's y el módulo de configuración, ya que en éste se encuentra los cables de transmisión de datos y la alimentación de 5v, por lo que al unirse los cables puede provocar un corto circuito, con lo cual se puede dañar los circuitos realizados.

- Es recomendable hacia el futuro reemplazar el modulo de ingreso de los mensajes, por un control remoto, al igual que es recomendable diseñar un circuito que controle la luminosidad de los led´s en el día como en la noche.

ANEXOS

ANEXO 1. PROGRAMA REALIZADO PARA EL CIRCUITO QUE CONTROLA LA MATRIZ DE LEDS.

```
$regfile = "m16def.dat"
$hwstack = 70                              ' default use 32 for the hardware stack
$swstack = 70                              ' default use 10 for the SW stack
$framesize = 70                            ' default use 40 for the frame space
$crystal = 16000000                          ' velocidad transmision datos
$baud = 38400
On Urxc Rec_isr                            ' define serial receive ISR
Enable Serial
Config Timer0 = Timer , Prescale = 64                'configuracion timer 0
On Ovf0 Tim0_isr

' declaracion funciones
Declare Function Crc(byval Crclength As Byte) As Byte
Declare Sub Readmem(byval Memaddress As Word , Byval Memlength As Byte)
Declare Sub Writemem(byval Memaddress As Word , Byval Memlength As Byte)
Declare Sub Writemsg(byval Memnumber As Byte)
Declare Function Readmsg(byval Memnumber As Byte) As Word
Declare Sub Rxprocess()
Declare Sub Txmsg()
Declare Sub Rowscreen()
Declare Sub Updatemsg()
Declare Sub Memtomsg()
Declare Sub Asciidecode(byval Letter As Byte)
Declare Sub Timemsg()
Declare Sub Setmsg()
Declare Sub Setdate()
Declare Sub Settime()
Declare Sub Readtime()
Declare Sub Realtime()
Declare Sub Comparetime()
Declare Sub Updateposmsg()
Declare Sub Screenfreq()

' declaraciones de variables
Dim Timedata(16) As Byte                   ' tiempo para mostrar mensaje
Dim Databuffer(150) As Byte                ' texto del mensaje
Dim Datamem(100) As Byte
Dim Screen(80) As Byte
Dim Datamsg(100) As Byte
Dim Chardecode(5) As Byte
Dim Rxtemp As Byte
Dim Year As Byte
```

```
Dim Month As Byte
Dim Datem As Byte
Dim Hour As Byte
Dim Minutes As Byte
Dim Seconds As Byte
Dim Weekday As Byte
Dim Temp As Word
Dim Temp1 As Word
Dim Temp2 As Byte
Dim Temp3 As Byte
Dim Temp4 As Byte
Dim Temp5 As Byte
Dim Temp6 As Byte
Dim Temp7 As Byte
Dim Key As Byte
Dim J As Byte
Dim Dia As Byte
Dim Hora As Byte
Dim Minutos As Byte
Dim Temptimer1 As Word
Dim Memaddressl As Byte
Dim Memaddressh As Byte
Dim Memvalue As Byte
Dim Memcounter As Word
Dim Ibuffer As Byte
Dim Datasend1 As Dword
Dim Datasend2 As Dword
Dim Datasend3 As Dword
Dim Datasend4 As Dword
Dim Datasend5 As Dword
Dim X As Byte
Dim Updatescreen As Word
Dim Fupdatescreen As Bit
Dim Updatetime As Word
Dim Fupdatetime As Bit
Dim Letterseg As Byte
Dim Letterpos As Byte
Dim Rxlength As Byte
Dim Rxcounter As Byte
Dim Fdata As Bit
Dim Frxdata As Bit
Dim Fscreen As Bit
```

```
Dim Nummsg As Byte
Dim Time1seg As Word
Dim Modescreen As Byte
Dim X7 As Byte
Dim Temps2 As Byte
Dim Temps3 As Byte
Dim Foutmode As Bit
Dim Frealtime As Bit
'declaracion de alias
'salidas
Row1 Alias Portb.0
Row2 Alias Portb.1
Row3 Alias Portb.2
Row4 Alias Portb.3
Row5 Alias Portb.4
Row6 Alias Portb.5
Row7 Alias Portb.6
Rowport Alias Portb
Shiftclk Alias Porta.4
Shiftdata Alias Porta.3
Shiftstrobe Alias Porta.1
Shiftenable Alias Porta.0
'direccion ds1307
Const Ds1307w = &HD0                    ' escritura
Const Ds1307r = &HD1                    ' lectura
Const Memcontrolw = &HA0                ' slave write address
Const Memcontrolr = &HA1                ' slave read address
Const Memoffset = 256
Const Memmaxlength = 56
Const Updatescreenmax = 4000
Const Updatetimemax = 3000
Const Const1seg = 488
Const Const5seg = 6000
'configuracion de los pines scl y sda
Config I2cdelay = 30
Config Sda = Portc.1
Config Scl = Portc.0
I2cinit
' configuracion puertos
Ddra = &B11111011
Porta = &B00000000
Ddrb = &B11111111
```

```
Portb = &B00011111
Ddrc = &B11111111
Portc = &B00000000
Ddrd = &B00010110
Portd = &B11100001
'inicializacion de variables
Time1seg = 0
Modescreen = 1
Hour = 14
Year = 10
Datem = 25
Month = 12
Minutes = 59
Seconds = 38
Dia = 1
Fupdatescreen = 0
Updatescreen = 0
Letterseg = 0
Letterpos = 0
Rxlength = 0
Rxcounter = 1
Fdata = 0
Frxdata = 0
Fupdatetime = 0
Updatetime = 0
Fscreen = 0
Frealtime = 0
X = 0
Fupdatescreen = 1
Foutmode = 0
Cursor Off Noblink
Enable Timer0
Start Timer0
Enable Interrupts
' limpia pantalla
For Temp2 = 1 To 80
Screen(temp2) = &B0000000
Next
'lazo principal
Do
  Call Screenfreq()
  If Fupdatetime = 1 Then
```

```
    Fupdatetime = 0
    Call Readtime()
    If Frealtime = 1 Then
     Call Realtime()
    End If
  End If
  If Fdata = 1 Then
    Fdata = 0
    Call Rxprocess()
    Call Screenfreq()
  End If
  If Fupdatescreen = 1 Then
    Fupdatescreen = 0
    Frealtime = 0
    If Foutmode = 1 Then Modescreen = 1
    If Modescreen = 1 Then
     For X7 = 1 To 100
        Datamem(x7) = 0
        Datamsg(x7) = 0
        Call Screenfreq()
      Next
      For X7 = 1 To 80
        Screen(x7) = 0
      Next
      Call Readmem(1 , 1)                    ' lee mensaje actual
      Temp2 = Datamem(1)
      If Temp2 > 9 Then
Datamem(1) = 1
        Call Writemem(1 , 1)
        Temp2 = 1
      End If
      Temp1 = Readmsg(temp2)
      Call Readmem(temp1 , 56)
      Call Memtomsg()
      Call Screenfreq()
      Call Updatemsg()
    End If
    If Modescreen = 2 Then
      Call Comparetime()
      Call Screenfreq()
      If Foutmode = 0 Then Call Timemsg()
    End If
```

```
      If Modescreen = 3 Then
        Call Comparetime()
        If Foutmode = 0 Then Call Realtime()
        Frealtime = 1
        Call Screenfreq()
      End If
      Incr Modescreen
      If Modescreen > 3 Then Modescreen = 1
    End If
Loop
' subrutinas
' actualiza pantalla
Sub Screenfreq()
If Fscreen = 1 Then
      Fscreen = 0
      Call Rowscreen()
      Shiftstrobe = 0
      Shiftenable = 1
      Shiftout Shiftdata , Shiftclk , Datasend1 , 3
      Shiftout Shiftdata , Shiftclk , Datasend2 , 3
      Shiftout Shiftdata , Shiftclk , Datasend3 , 3
      Shiftout Shiftdata , Shiftclk , Datasend4 , 3
      Shiftout Shiftdata , Shiftclk , Datasend5 , 3
      If X = 0 Then Rowport = &B00000001
      If X = 1 Then Rowport = &B00000010
      If X = 2 Then Rowport = &B00000100
      If X = 3 Then Rowport = &B00001000
      Shiftenable = 0
      Shiftstrobe = 1
      Shiftstrobe = 0
      Incr X
      If X > 3 Then X = 0
End If
End Sub
'compara hora actual con hora de salida
Sub Comparetime()
        Foutmode = 0
        Temp2 = Datamsg(1) - &H30
        Shift Temp2 , Left , 4
        Temp3 = Datamsg(2) - &H30
        Temp4 = Datamsg(3) - &H30
        Shift Temp4 , Left , 4
```

```
        Temp5 = Datamsg(4) - &H30
        Temp6 = Datamsg(5) - &H30
        Shift Temp6 , Left , 4
        Temp7 = Datamsg(6) - &H30
        Temp2 = Temp2 Or Temp3
        Temp3 = Temp4 Or Temp5
        Temp4 = Temp6 Or Temp7
        Temp2 = Makedec(temp2)
        Temp3 = Makedec(temp3)
        Temp4 = Makedec(temp4)
        Temp1 = Temp2 * Temp3
        Temp = Hour * Minutes
        If Temp1 >= Temp Then
            If Temp4 => Seconds Then Foutmode = 1
        End If
End Sub
' copia memoria datamem a datamsg
Sub Memtomsg()
  Local Datalength As Byte
  For Datalength = 1 To Memmaxlength
Datamsg(datalength) = Datamem(datalength)
    If Datamsg(datalength) = 0 Or Datamsg(datalength) = 255 Then Exit For
  Next
Datamsg(datalength) = 0
End Sub
' lee mensaje actual
Function Readmsg(byval Memnumber As Byte) As Word
  Local Mempos As Word
  Mempos = Memnumber - 1
  Mempos = Mempos * Memmaxlength
  Mempos = Mempos + Memoffset
  Readmsg = Mempos
End Function
' escribe mensaje
Sub Writemsg(byval Memnumber As Byte)
  Local Mempos As Word
  Mempos = Memnumber - 1
  Mempos = Mempos * Memmaxlength
  Mempos = Mempos + Memoffset
  Call Writemem(mempos , Memmaxlength)
End Sub
' memoria 24lc256
```

```
' memaddres direccion memoria
' memlength longitud datos
' outmem data mem value
Sub Readmem(byval Memaddress As Word , Byval Memlength As Byte)
  Local Memend As Word
  Ibuffer = 1
  Memend = Memaddress + Memlength
  I2cstop
  For Memcounter = Memaddress To Memend
    Memaddressh = High(memcounter)
    Memaddressl = Low(memcounter)
    I2cstart                          'generate start
    I2cwbyte Memcontrolw                  'slave adsress
    I2cwbyte Memaddressh                   'address of EEPROM
    I2cwbyte Memaddressl                   'address of EEPROM
    I2cstart                          'repeated start
    I2cwbyte Memcontrolr                 'slave address (read)
    I2crbyte Memvalue , Nack               'read byte
    I2cstop
    If Memvalue = 0 Or Memvalue = 255 Then Exit For
Datamem(ibuffer) = Memvalue
    Incr Ibuffer
  Next
End Sub
' escribe dato en la memoria 24c256
Sub Writemem(byval Memaddress As Word , Byval Memlength As Byte)
  Local Memend As Word
  Ibuffer = 1
  Memend = Memaddress + Memlength
  For Memcounter = Memaddress To Memend
    Memaddressh = High(memcounter)
    Memaddressl = Low(memcounter)
    Memvalue = Datamem(ibuffer)
    Incr Ibuffer
    If Memvalue = 0 Then Exit For
    I2cstart                          'generate start
    I2cwbyte Memcontrolw                  'slave adsress
    I2cwbyte Memaddressh                   'address of EEPROM
    I2cwbyte Memaddressl                   'address of EEPROM
    I2cwbyte Memvalue                     'read byte
    I2cstop
    Waitms 10
```

```
  Next
End Sub
' lee hora actual
Sub Readtime()
I2cstart
  I2cwbyte Ds1307w                    ' send address
  I2cwbyte 0                          ' start address in 1307
  I2cstart                            ' Generate start code
  I2cwbyte Ds1307r                    ' send address
  Call Screenfreq()
  I2crbyte Seconds , Ack
  I2crbyte Minutes , Ack              ' MINUTES
  Call Screenfreq()
  I2crbyte Hour , Ack                 ' Hours
  I2crbyte Weekday , Ack              ' Day of Week
  I2crbyte Datem , Ack                ' Day of Month
  I2crbyte Month , Ack                ' Month of Year
  Call Screenfreq()
  I2crbyte Year , Nack                ' Year
  I2cstop
  If Seconds.7 = 0 Then
    Seconds = Makedec(seconds) : Minutes = Makedec(minutes) : Hour = Makedec(hour)
    Datem = Makedec(datem) : Month = Makedec(month) : Year = Makedec(year)
  Else
    Call Setdate()
    Call Settime()
  End If
End Sub
' actualiza fecha
Sub Setdate()
Datem = Makebcd(datem) : Month = Makebcd(month) : Year = Makebcd(year)
 I2cstart                             ' Generate start code
 I2cwbyte Ds1307w                     ' send address
 I2cwbyte 4                           ' starting address in 1307
 I2cwbyte Datem                       ' Send Data to SECONDS
 I2cwbyte Month                       ' MINUTES
 I2cwbyte Year                        ' Hours
 I2cstop
End Sub
' actualiza hora
Sub Settime()
Seconds = Makebcd(seconds) : Minutes = Makebcd(minutes) : Hour = Makebcd(hour)
```

```
  I2cstart                                ' Generate start code
  I2cwbyte Ds1307w                          ' send address
  I2cwbyte 0                              ' starting address in 1307
  I2cwbyte Seconds                          ' Send Data to SECONDS
  I2cwbyte Minutes                        ' MINUTES
  I2cwbyte Hour                           ' Hours
I2cstop
End Sub
' convierte datos para la matriz de leds
Sub Rowscreen()
   Local Datascreen As Dword
   Local Datascreen1 As Dword
   Local Iscreen As Byte
   Local Xrow As Byte
   Local Xrow1 As Byte
   Local X1 As Byte
   Local X2 As Byte
   Xrow = &B00000001
   Xrow1 = &B00010000
   Shift Xrow , Left , X
   Shift Xrow1 , Left , X
   Datasend1 = 0
   Datasend2 = 0
   Datasend3 = 0
   Datasend4 = 0
   Datasend5 = 0
   For Iscreen = 0 To 79
      Temps2 = Screen(iscreen + 1) And Xrow
      Temps3 = Screen(iscreen + 1) And Xrow1
      X1 = X + 4
      Shift Temps2 , Right , X
      Shift Temps3 , Right , X1
      Shift Temps2 , Left , 1
      Temps2 = Temps2 Or Temps3
      Datascreen = Temps2
      Shift Datascreen , Left , X2
      X2 = X2 + 2
      If X2 > 31 Then X2 = 0
      If Iscreen < 16 Then Datasend1 = Datascreen Or Datasend1
      If Iscreen > 15 And Iscreen < 32 Then Datasend2 = Datascreen Or Datasend2
      If Iscreen > 31 And Iscreen < 48 Then Datasend3 = Datascreen Or Datasend3
      If Iscreen > 47 And Iscreen < 64 Then Datasend4 = Datascreen Or Datasend4
```

```
    If Iscreen > 63 And Iscreen < 80 Then Datasend5 = Datascreen Or Datasend5
Next
End Sub
' muestra la hora de salida en la matriz de leds
Sub Timemsg()
  Local Lettermsg As Byte
  Local I As Byte
  Local I1 As Byte
  For I = 1 To 80
Screen(i) = 0
  Next
Timedata(1) = "S"
Timedata(2) = "A"
Timedata(3) = "L"
Timedata(4) = "I"
Timedata(5) = "D"
Timedata(6) = "A"
Timedata(7) = " "
Timedata(8) = Datamsg(1)
Timedata(9) = Datamsg(2)
Timedata(10) = ":"
Timedata(11) = Datamsg(3)
Timedata(12) = Datamsg(4)
Timedata(13) = ":"
Timedata(14) = Datamsg(5)
Timedata(15) = Datamsg(6)
  For I = 0 To 14
    Lettermsg = Timedata(1 + I)
    If Lettermsg = 0 Then Exit For
    Call Asciidecode(lettermsg)
    I1 = I * 5
Screen(i1 + 1) = Chardecode(1)
Screen(i1 + 2) = Chardecode(2)
Screen(i1 + 3) = Chardecode(3)
Screen(i1 + 4) = Chardecode(4)
Screen(i1 + 5) = Chardecode(5)
  Next
End Sub
' muestra la hora actual
Sub Realtime()
Local Lettermsg As Byte
  Local I As Byte
```

```
  Local I1 As Byte
  Local I2 As Byte
  Lettermsg = Makebcd(hour)
Timedata(2) = Lettermsg And &H0F
Timedata(2) = &H30 + Timedata(2)
  Shift Lettermsg , Right , 4
Timedata(1) = Lettermsg + &H30
Timedata(3) = ":"
  Lettermsg = Makebcd(minutes)
Timedata(5) = Lettermsg And &H0F
Timedata(5) = Timedata(5) + &H30
  Shift Lettermsg , Right , 4
Timedata(4) = Lettermsg + &H30
Timedata(6) = ":"
  Lettermsg = Makebcd(seconds)
Timedata(8) = Lettermsg And &H0F
Timedata(8) = Timedata(8) + &H30
  Shift Lettermsg , Right , 4
Timedata(7) = Lettermsg + &H30
  For I = 1 To 80
Screen(i) = 0
  Next
  For I = 0 To 7
    Lettermsg = Timedata(i + 1)
    If Lettermsg = 0 Then Exit For
    Call Asciidecode(lettermsg)
    I1 = I * 5
    I2 = I1 + 20
Screen(i2 + 1) = Chardecode(1)
Screen(i2 + 2) = Chardecode(2)
Screen(i2 + 3) = Chardecode(3)
Screen(i2 + 4) = Chardecode(4)
Screen(i2 + 5) = Chardecode(5)
Next
End Sub
' crea la referencia para la decodificacion de los codigos ascii
Sub Asciidecode(byval Letter As Byte)
  Local Segmentout As Word
  Local Z As Byte
  If Letter = "1" Then Segmentout = Loadlabel(1c)
  If Letter = "2" Then Segmentout = Loadlabel(2c)
  If Letter = "3" Then Segmentout = Loadlabel(3c)
```

```
If Letter = "4" Then Segmentout = Loadlabel(4c)
If Letter = "5" Then Segmentout = Loadlabel(5c)
If Letter = "6" Then Segmentout = Loadlabel(6c)
If Letter = "7" Then Segmentout = Loadlabel(7c)
If Letter = "8" Then Segmentout = Loadlabel(8c)
If Letter = "9" Then Segmentout = Loadlabel(9c)
If Letter = "0" Then Segmentout = Loadlabel(0c)
If Letter = "a" Then Segmentout = Loadlabel(al)
If Letter = "b" Then Segmentout = Loadlabel(bl)
If Letter = "c" Then Segmentout = Loadlabel(cl)
If Letter = "d" Then Segmentout = Loadlabel(dl)
If Letter = "e" Then Segmentout = Loadlabel(el)
If Letter = "f" Then Segmentout = Loadlabel(fl)
If Letter = "g" Then Segmentout = Loadlabel(gl)
If Letter = "h" Then Segmentout = Loadlabel(hl)
If Letter = "i" Then Segmentout = Loadlabel(il)
If Letter = "j" Then Segmentout = Loadlabel(jl)
If Letter = "k" Then Segmentout = Loadlabel(kl)
If Letter = "l" Then Segmentout = Loadlabel(ll)
If Letter = "m" Then Segmentout = Loadlabel(ml)
If Letter = "n" Then Segmentout = Loadlabel(nl)
If Letter = "o" Then Segmentout = Loadlabel(ol)
If Letter = "p" Then Segmentout = Loadlabel(pl)
If Letter = "q" Then Segmentout = Loadlabel(ql)
If Letter = "r" Then Segmentout = Loadlabel(rl)
If Letter = "s" Then Segmentout = Loadlabel(sl)
If Letter = "t" Then Segmentout = Loadlabel(tl)
If Letter = "u" Then Segmentout = Loadlabel(ul)
If Letter = "v" Then Segmentout = Loadlabel(vl)
If Letter = "w" Then Segmentout = Loadlabel(wl)
If Letter = "x" Then Segmentout = Loadlabel(xl)
If Letter = "y" Then Segmentout = Loadlabel(yl)
If Letter = "z" Then Segmentout = Loadlabel(zl)
If Letter = "A" Then Segmentout = Loadlabel(au)
If Letter = "B" Then Segmentout = Loadlabel(bu)
If Letter = "C" Then Segmentout = Loadlabel(cu)
If Letter = "D" Then Segmentout = Loadlabel(du)
If Letter = "E" Then Segmentout = Loadlabel(eu)
If Letter = "F" Then Segmentout = Loadlabel(fu)
If Letter = "G" Then Segmentout = Loadlabel(gu)
If Letter = "H" Then Segmentout = Loadlabel(hu)
If Letter = "I" Then Segmentout = Loadlabel(iu)
```

```
  If Letter = "J" Then Segmentout = Loadlabel(ju)
  If Letter = "K" Then Segmentout = Loadlabel(ku)
  If Letter = "L" Then Segmentout = Loadlabel(lu)
  If Letter = "M" Then Segmentout = Loadlabel(mu)
  If Letter = "N" Then Segmentout = Loadlabel(nu)
  If Letter = "O" Then Segmentout = Loadlabel(ou)
  If Letter = "P" Then Segmentout = Loadlabel(pu)
  If Letter = "Q" Then Segmentout = Loadlabel(qu)
  If Letter = "R" Then Segmentout = Loadlabel(ru)
  If Letter = "S" Then Segmentout = Loadlabel(su)
  If Letter = "T" Then Segmentout = Loadlabel(tu)
  If Letter = "U" Then Segmentout = Loadlabel(uu)
  If Letter = "V" Then Segmentout = Loadlabel(vu)
  If Letter = "W" Then Segmentout = Loadlabel(wu)
  If Letter = "X" Then Segmentout = Loadlabel(xu)
  If Letter = "Y" Then Segmentout = Loadlabel(yu)
  If Letter = "Z" Then Segmentout = Loadlabel(zu)
' SPECIAL CHART
  If Letter = &H20 Then Segmentout = Loadlabel(spacec)
  If Letter = ":" Then Segmentout = Loadlabel(dospuntos)
  If Segmentout > 0 Then
    For Z = 0 To 4
Chardecode(z + 1) = Cpeek(segmentout)
      Incr Segmentout
    Next
  End If
End Sub
' actualiza destino
Sub Updatemsg()
Local Lettermsg As Byte
  Local I As Byte
  Local I1 As Byte
  For I = 0 To 15
    Lettermsg = Datamsg(7 + I)
    If Lettermsg = 0 Then Exit For
    Call Asciidecode(lettermsg)
    I1 = I * 5
Screen(i1 + 1) = Chardecode(1)
Screen(i1 + 2) = Chardecode(2)
Screen(i1 + 3) = Chardecode(3)
Screen(i1 + 4) = Chardecode(4)
Screen(i1 + 5) = Chardecode(5)
```

```
  Next
End Sub
' transmite dato mediante el uart
Sub Txmsg()
    Local Txlength As Byte
    Local Txcounter As Byte
    Txlength = Databuffer(2)
    For Txcounter = 1 To Txlength
      Print Chr(databuffer(txcounter));
    Next
End Sub
Function Crc(byval Crclength As Byte) As Byte
    Local Crccounter As Byte
    Local Crctemp As Byte
    Crctemp = 0
    For Crccounter = 1 To Crclength
      Crctemp = Databuffer(crccounter) Xor Crctemp
    Next
    Crc = Crctemp
End Function
'vector de interrupcion del timer 0
Tim0_isr:
  Incr Updatescreen
  Incr Time1seg
  If Time1seg > Const1seg Then
    Time1seg = 0
    Fupdatetime = 1
    Incr Updatescreen
    If Updatescreen > Const5seg Then
      Updatescreen = 0
      Fupdatescreen = 1
    End If
  End If
Fscreen = 1
Return
'vsubrutina procesa dato recibido por el puerto uart
Sub Rxprocess()
  Local Processcounter As Byte
  Local Processlength As Byte
  If Databuffer(3) = "M" And Databuffer(4) = "S" Then
    Foutmode = 1
  End If
```

```
  If Databuffer(3) = "M" And Databuffer(4) = "V" Then
    Foutmode = 0
  End If
  If Databuffer(3) = "N" And Databuffer(4) = "M" Then      ' MENSAJE ACTUAL A MOSTRAR
Datamem(1) = Databuffer(5) - &H30
    Call Writemem(1 , 1)
  End If
  If Databuffer(3) = "T" And Databuffer(4) = "R" Then      'lectura tiempo
Databuffer(1) = &HFE
Databuffer(2) = 11
Databuffer(3) = "T"
Databuffer(4) = "D"
Databuffer(5) = Year
Databuffer(6) = Month
Databuffer(7) = Datem
Databuffer(8) = Hour
Databuffer(9) = Minutes
Databuffer(10) = Seconds
Databuffer(11) = Crc(10)
    Call Txmsg()
  Elseif Databuffer(3) = "T" And Databuffer(4) = "W" Then
    Year = Databuffer(5)
    Month = Databuffer(6)
    Datem = Databuffer(7)
    Hour = Databuffer(8)
    Minutes = Databuffer(9)
    Seconds = Databuffer(10)
    Call Settime()
    Call Setdate()
  Elseif Databuffer(3) = "M" And Databuffer(4) = "W" Then
    Processlength = Databuffer(2) - 1
    For Processcounter = 1 To 255
Datamem(processcounter) = 0
    Next
    For Processcounter = 6 To Processlength
Datamem(processcounter -5) = Databuffer(processcounter)
    Next
    Call Writemsg(databuffer(5))
  Elseif Databuffer(3) = "M" And Databuffer(4) = "R" Then
    For Processcounter = 1 To 255
Datamem(processcounter) = 0
Next
```

```
Temp1 = Readmsg(databuffer(5))                    ' numero de mensaje
Call Readmem(temp1 , Memmaxlength)
Databuffer(1) = &HFE
Databuffer(2) = Ibuffer + 5
Databuffer(3) = "M"
Databuffer(4) = "D"
Databuffer(5) = Databuffer(5)
    Ibuffer = Ibuffer + 5
    For Processcounter = 6 To Ibuffer
Databuffer(processcounter) = Datamem(processcounter - 5)
    Next
    Ibuffer = Ibuffer - 1
Databuffer(ibuffer + 1) = Crc(ibuffer)
    Call Txmsg()
  End If
End Sub
'vector interrupcion del puerto rx del uart
Rec_isr:
  Rxtemp = Udr
  If Frxdata = 0 Then
    If Rxtemp = &HFE Then
Databuffer(1) = Rxtemp
      Rxcounter = 2
      Frxdata = 1
      Fdata = 0
    End If
  Else
    If Rxcounter = 2 Then Rxlength = Rxtemp
Databuffer(rxcounter) = Rxtemp
    If Rxcounter > 2 And Rxlength = Rxcounter Then
      Fdata = 1
      Frxdata = 0
    End If
    Incr Rxcounter
End If
Return
' valores por defecto de la memoria eeprom del uc
$eeprom
  Ini_eeprom:
' clave address  01 - 04
  Data &H00 ,&H00 , 0 , 0
  Data 0 , 2 , 0 , 0
```

```
  Data 25 , 0 , 0 , 0
  Data 20 , 0 , 40 , 0
  Data 75 , 0 , 100 , 0
  Data 0 , 0 , 0 , 0
  Data 0 , 0 , 0 , 0
  Data 0 , 0 , 0 , 0
Data 0 , 0 , 0 , 0
$data
End
' valores de decodificacion de los datos ascii
Dospuntos:
Data 0 , 0 , 54 , 54 , 0
Spacec:
Data 0 , 0 , 0 , 0 , 0
1c:
Data 1 , 17 , 33 , 127 , 1
2c:
Data 39 , 73 , 73 , 73 , 49
3c:
Data 34 , 73 , 73 , 73 , 54
4c:
Data 12 , 20 , 36 , 127 , 4
5c:
Data 122 , 73 , 73 , 73 , 70
6c:
Data 62 , 73 , 81 , 81 , 46
7c:
Data 64 , 67 , 68 , 72 , 112
8c:
Data 54 , 73 , 73 , 73 , 54
9c:
Data 50 , 73 , 73 , 73 , 62
0c:
Data 62 , 65 , 65 , 62 , 0

' MAYUSCULAS
Au:
Data 31 , 36 , 68 , 36 , 31
Bu:
Data 65 , 127 , 73 , 73 , 54
Cu:
Data 62 , 65 , 65 , 65 , 34
```

Du:
Data 65 , 127 , 65 , 65 , 62
Eu:
Data 127 , 73 , 73 , 65 , 65
Fu:
Data 127 , 72 , 72 , 64 , 64
Gu:
Data 62 , 65 , 73 , 73 , 46
Hu:
Data 127 , 8 , 8 , 8 , 127
Iu:
Data 0 , 65 , 127 , 65 , 0
Ju:
Data 2 , 1 , 65 , 126 , 64
Ku:
Data 127 , 8 , 20 , 34 , 65
Lu:
Data 127 , 1 , 1 , 1 , 1
Mu:
Data 127 , 32 , 24 , 32 , 127
Nu:
Data 127 , 32 , 16 , 8 , 127
Ou:
Data 62 , 65 , 65 , 65 , 62
Pu:
Data 127 , 72 , 72 , 72 , 48
Qu:
Data 62 , 65 , 69 , 67 , 63
Ru:
Data 127 , 72 , 76 , 74 , 49
Su:
Data 50 , 73 , 73 , 73 , 38
Tu:
Data 64 , 64 , 127 , 64 , 64
Uu:
Data 126 , 1 , 1 , 1 , 126
Vu:
Data 112 , 12 , 3 , 12 , 112
Wu:
Data 127 , 2 , 12 , 2 , 127
Xu:
Data 99 , 20 , 8 , 20 , 99

```
Yu:
Data 96 , 16 , 15 , 16 , 96
Zu:
Data 67 , 69 , 73 , 81 , 97
' MINUSCULAS
Al:
Data 2 , 21 , 21 , 21 , 15
Bl:
Data 127 , 10 , 17 , 17 , 14
Cl:
Data 0 , 14 , 17 , 17 , 17
Dl:
Data 14 , 17 , 17 , 10 , 127
El:
Data 14 , 21 , 21 , 21 , 12
Fl:
Data 8 , 63 , 72 , 72 , 32
Gl:
Data 8 , 21 , 21 , 21 , 30
Hl:
Data 127 , 8 , 16 , 16 , 15
Il:
Data 0 , 17 , 95 , 1 , 0
Jl:
Data 2 , 1 , 17 , 94 , 0
Kl:
Data 0 , 127 , 4 , 10 , 17
Ll:
Data 0 , 65 , 127 , 1 , 0
Ml:
Data 31 , 16 , 8 , 16 , 15
Nl:
Data 31 , 8 , 16 , 16 , 15
Ol:
Data 14 , 17 , 17 , 17 , 14
Pl:
Data 31 , 20 , 20 , 20 , 8
Ql:
Data 8 , 20 , 20 , 20 , 31
Rl:
Data 31 , 8 , 16 , 16 , 8
Sl:
```

Data 9 , 21 , 21 , 21 , 18

Tl:

Data 0 , 8 , 126 , 9 , 9

Ul:

Data 30 , 1 , 1 , 2 , 31

Vl:

Data 28 , 2 , 1 , 2 , 28

Wl:

Data 30 , 1 , 6 , 1 , 30

Xl:

Data 17 , 10 , 4 , 10 , 17

Yl:

Data 24 , 5 , 5 , 5 , 30

Zl:

Data 17 , 19 , 21 , 25 , 17

ANEXO 2. PROGRAMA REALIZADO PARA EL MÓDULO DE CONFIGURACIÓN DE TEXTO Y HORA.

```
$regfile = "m16def.dat"
$hwstack = 50                                ' default use 32 for the hardware stack
$swstack = 50                                ' default use 10 for the SW stack
$framesize = 50                               ' default use 40 for the frame space
$crystal = 8000000
$baud = 38400
On Urxc Rec_isr                               ' define serial receive ISR
Enable Serial
Config Timer0 = Timer , Prescale = 64
On Ovf0 Tim1_isr
' declaracion funciones
Declare Sub Keytable()
Declare Sub Clave()
Declare Sub Keycheck()
Declare Sub Verify()
Declare Sub Setclave()
Declare Sub Setmsg()
Declare Sub Writetime()
Declare Sub Showtime()
Declare Sub Settime()
Declare Sub Readmsg()
Declare Sub Updateposmsg()
Declare Sub Readdata()
Declare Sub Writedata()
Declare Function Crc(byval Crclength As Byte) As Byte
Declare Sub Txdata()
Declare Sub Datawait()
Declare Sub Receivecommand()
Declare Sub Readtime()
Declare Sub Updatelcd()
Declare Sub Clrdata()
Declare Sub Clrbuffer()
Declare Sub Msgshow()
Declare Sub Modestatic()
' configuracion LCD
Config Lcdpin = Pin , Db4 = Portc.4 , Db5 = Portc.5 , Db6 = Portc.6 , Db7 = Portc.7 , E = Portc.3
, Rs = Portc.1 , Wr = Portc.2
Config Lcd = 16 * 2
' declaracion de variables
Dim Timemsg(7) As Byte                        ' tiempo para mostrar mensaje
Dim Datamsg(100) As Byte                      ' texto del mensaje
Dim Serialbuffer(127) As Byte
```

```
Dim Posmsg As Byte
Dim Lenmsg As Byte
Dim Cursorlcd As Byte
Dim Subkey As Byte
Dim Modeletter As Byte
Dim Rxlength As Byte
Dim Rxtemp As Byte
Dim Frxdata As Bit
Dim Rxcounter As Byte
Dim Fdata As Byte
Dim Msgcounter As Byte
Dim Year As Byte
Dim Month As Byte
Dim Datem As Byte
Dim Hour As Byte
Dim Minutes As Byte
Dim Seconds As Byte
Dim Weekday As Byte
Dim Fclave As Bit
Dim I As Byte
Dim Password(4) As Byte
Dim Temp1 As Word
Dim Temp2 As Byte
Dim Temp3 As Byte
Dim Temp4 As Byte
Dim Temp5 As Byte
Dim Temp6 As Byte
Dim Temp7 As Byte
Dim Key As Byte
Dim J As Byte
Dim Ypos As Byte
Dim Xpos As Byte
Dim Pos As Byte
Dim Temptime(15) As Byte
Dim Timepos As Byte
Dim Ftime As Bit
Dim Numbermsg As Byte
Dim Delaywait As Word
Dim Ferror As Bit
Dim Temptimer1 As Word
Dim Delayexit As Word
Dim Timecounter As Word
```

```
Dim Fmode As Bit
' alias
' entradas teclado
F1 Alias Pina.7
F2 Alias Pina.6
F3 Alias Pina.5
F4 Alias Pina.4
' salidas teclado
C1 Alias Porta.3
C2 Alias Porta.2
C3 Alias Porta.1
C4 Alias Porta.0
'ENTRADAS
' teclas de navegacion
Swup Alias Pinb.4
Swdown Alias Pinb.5
Swright Alias Pinb.6
Swleft Alias Pinb.7
Swadd Alias Pinb.3
Swdel Alias Pinb.2
Swsave Alias Pinb.1
Swstatic Alias Pinb.0
' constantes
Const Asterisco = 10
Const Numeral = 11
Const Keya = 12
Const Keyb = 13
Const Keyc = 14
Const Keyd = 15
Const Lengthbuffermax = 127
Const Lengthmsgmax = 50
Const Delaywaitmax = 30000
Const Consttimecounter = 10

' configuracion puertos
Ddra = &B00001111
Porta = &B11110000
Ddrb = &B00000000
Portb = &B11111111
Ddrc = &B11111111
Portc = &B00000000
Ddrd = &B00010110
```

```
Portd = &B11100001
' inicialiacion de variables
Fmode = 0
Fclave = 0
Delaywait = 0
Ferror = 0
Ftime = 0
Timecounter = 0
Hour = 24
Year = 10
Datem = 25
Month = 12
Minutes = 59
Seconds = 38
Subkey = 1
Cursor Off Noblink
Disable Timer0
Enable Interrupts
Waitms 300
Cls
Enable Timer0
Start Timer0
Msgcounter = 1
Locate 2 , 16
Lcd Msgcounter
' lazo principal
Do
  If Ftime = 1 Then
    Ftime = 0
    Incr Timecounter
    If Timecounter > Consttimecounter Then
      Timecounter = 0
      Call Readtime()
      Call Showtime()
      Locate 2 , 16
      Lcd Msgcounter
    End If
  End If
  If Fdata = 1 Then
    Fdata = 0
    Call Receivecommand()
  End If
```

```
  If Swup = 0 Then
    Incr Msgcounter
    If Msgcounter > 9 Then Msgcounter = 9
    Locate 2 , 16
    Lcd Msgcounter
    Call Msgshow()
    Call Txdata()
    Waitms 200
  End If
  If Swdown = 0 Then
    Decr Msgcounter
    If Msgcounter = 0 Then Msgcounter = 1
    Locate 2 , 16
    Lcd Msgcounter
    Call Msgshow()
    Call Txdata()
    Waitms 200
  End If
  If Swup = 0 Then
    Incr Msgcounter
    If Msgcounter > 9 Then Msgcounter = 9
    Call Msgshow()
  End If
  If Swstatic = 0 Then
    Waitms 200
    Fmode = Not Fmode
    Call Modestatic()
    Call Txdata()
  End If
  Call Keycheck()
If Key = Keya Then                    ' configura fecha y hora
    Call Verify()                     ' verifica clave
If Fclave = 1 Then
      Call Settime()
    End If
  End If
  If Key = Keyb Then                      ' lee mensajes mensajes
    Call Verify()                     ' verifica clave
    If Fclave = 1 Then
      Call Readmsg()
    End If
  End If
```

```
  If Key = Asterisco Then                    ' cambio clave
    Call Verify()
    If Fclave = 1 Then
      Call Setclave()
    End If
  End If
Loop
Sub Modestatic()
Serialbuffer(1) = &HFE
Serialbuffer(2) = 6
Serialbuffer(3) = "M"
  If Fmode = 1 Then Serialbuffer(4) = "S"
  If Fmode = 0 Then Serialbuffer(4) = "V"
Serialbuffer(5) = Msgcounter + &H30
Serialbuffer(6) = Crc(5)
End Sub
Sub Msgshow()
Serialbuffer(1) = &HFE
Serialbuffer(2) = 6
Serialbuffer(3) = "N"
Serialbuffer(4) = "M"
Serialbuffer(5) = Msgcounter + &H30
Serialbuffer(6) = Crc(5)
End Sub
Sub Clrdata()
  For Temp1 = 1 To Lengthmsgmax
Datamsg(temp1) = 0
  Next
End Sub
Sub Clrbuffer()
  For Temp1 = 1 To Lengthbuffermax
Serialbuffer(temp1) = 0
Next
End Sub
' subrutinas
' convierte al tecla hexadecimal para ingresar letras y numeros
Sub Keytable()
        If Key = 1 Then Key = "1"
        If Key = 2 Then
          If Modeletter = 0 Then
            If Subkey = 1 Then Key = "a"
            If Subkey = 2 Then Key = "b"
```

```
      If Subkey = 3 Then Key = "c"
      If Subkey = 4 Then Key = "2"
    Else
      If Subkey = 1 Then Key = "A"
      If Subkey = 2 Then Key = "B"
      If Subkey = 3 Then Key = "C"
      If Subkey = 4 Then Key = "2"
    End If
    Incr Subkey
    If Subkey > 4 Then Subkey = 1
  End If
  If Key = 3 Then
    If Modeletter = 0 Then
      If Subkey = 1 Then Key = "d"
      If Subkey = 2 Then Key = "e"
      If Subkey = 3 Then Key = "f"
      If Subkey = 4 Then Key = "3"
    Else
      If Subkey = 1 Then Key = "D"
      If Subkey = 2 Then Key = "E"
      If Subkey = 3 Then Key = "F"
      If Subkey = 4 Then Key = "3"
    End If
    Incr Subkey
    If Subkey > 4 Then Subkey = 1
  End If
  If Key = 4 Then
    If Modeletter = 0 Then
      If Subkey = 1 Then Key = "g"
      If Subkey = 2 Then Key = "h"
      If Subkey = 3 Then Key = "i"
      If Subkey = 4 Then Key = "4"
    Else
      If Subkey = 1 Then Key = "G"
      If Subkey = 2 Then Key = "H"
      If Subkey = 3 Then Key = "I"
      If Subkey = 4 Then Key = "4"
    End If
    Incr Subkey
    If Subkey > 4 Then Subkey = 1
  End If
  If Key = 5 Then
```

```
  If Modeletter = 0 Then
    If Subkey = 1 Then Key = "j"
    If Subkey = 2 Then Key = "k"
    If Subkey = 3 Then Key = "l"
    If Subkey = 4 Then Key = "5"
  Else
    If Subkey = 1 Then Key = "J"
    If Subkey = 2 Then Key = "K"
    If Subkey = 3 Then Key = "L"
    If Subkey = 4 Then Key = "5"
  End If
  Incr Subkey
  If Subkey > 4 Then Subkey = 1
End If
If Key = 6 Then
  If Modeletter = 0 Then
    If Subkey = 1 Then Key = "m"
    If Subkey = 2 Then Key = "n"
    If Subkey = 3 Then Key = "o"
    If Subkey = 4 Then Key = "6"
  Else
    If Subkey = 1 Then Key = "M"
    If Subkey = 2 Then Key = "N"
    If Subkey = 3 Then Key = "O"
    If Subkey = 4 Then Key = "6"
  End If
  Incr Subkey
  If Subkey > 4 Then Subkey = 1
End If
If Key = 7 Then
  If Modeletter = 0 Then
    If Subkey = 1 Then Key = "p"
    If Subkey = 2 Then Key = "q"
    If Subkey = 3 Then Key = "r"
    If Subkey = 4 Then Key = "s"
    If Subkey = 5 Then Key = "7"
  Else
    If Subkey = 1 Then Key = "P"
    If Subkey = 2 Then Key = "Q"
    If Subkey = 3 Then Key = "R"
    If Subkey = 4 Then Key = "S"
    If Subkey = 5 Then Key = "7"
```

```
    End If
    Incr Subkey
    If Subkey > 5 Then Subkey = 1
  End If
  If Key = 8 Then
    If Modeletter = 0 Then
      If Subkey = 1 Then Key = "t"
      If Subkey = 2 Then Key = "u"
      If Subkey = 3 Then Key = "v"
      If Subkey = 4 Then Key = "8"
    Else
      If Subkey = 1 Then Key = "T"
      If Subkey = 2 Then Key = "U"
      If Subkey = 3 Then Key = "V"
      If Subkey = 4 Then Key = "8"
    End If
    Incr Subkey
    If Subkey > 4 Then Subkey = 1
  End If
  If Key = 9 Then
    If Modeletter = 0 Then
      If Subkey = 1 Then Key = "w"
      If Subkey = 2 Then Key = "x"
      If Subkey = 3 Then Key = "y"
      If Subkey = 4 Then Key = "z"
      If Subkey = 5 Then Key = "9"
    Else
      If Subkey = 1 Then Key = "W"
      If Subkey = 2 Then Key = "X"
      If Subkey = 3 Then Key = "Y"
      If Subkey = 4 Then Key = "Z"
      If Subkey = 5 Then Key = "9"
    End If
    Incr Subkey
    If Subkey > 5 Then Subkey = 1
  End If
  If Key = 0 Then
    If Subkey = 1 Then Key = " "
    If Subkey = 2 Then Key = "0"
    Incr Subkey
    If Subkey > 2 Then Subkey = 1
  End If
```

```
End Sub
Function Crc(byval Crclength As Byte) As Byte
    Local Crccounter As Byte
    Local Crctemp As Byte
    Crctemp = 0
    For Crccounter = 1 To Crclength
      Crctemp = Serialbuffer(crccounter) Xor Crctemp
Next
    Crc = Crctemp
End Function
' escribe dato para enviarse mediante el puerto uart
Sub Writedata()
  Local Datacount As Byte
  Local Datatemp As Byte
Serialbuffer(1) = &HFE
Serialbuffer(3) = "M"
Serialbuffer(4) = "W"
Serialbuffer(5) = Numbermsg
  For Datatemp = 6 To 11
Serialbuffer(datatemp) = Timemsg(datatemp - 5)
  Next
  Datacount = 1
  While Datamsg(datacount) > 0
Serialbuffer(datacount + 11) = Datamsg(datacount)
    Incr Datacount
  Wend
Serialbuffer(2) = Datacount + 11
  Datatemp = Datacount + 10
Serialbuffer(datatemp + 1) = Crc(datatemp)
End Sub
' envia peticion del mensaje almacenado
Sub Readdata()
  Local Datacount As Byte
  Local Datatemp As Byte
Serialbuffer(1) = &HFE
Serialbuffer(2) = 6
Serialbuffer(3) = "M"
Serialbuffer(4) = "R"
Serialbuffer(5) = Numbermsg
Serialbuffer(6) = Crc(5)
End Sub
' envia dato mediante el puerto uart
```

```
Sub Txdata()
  Local Txtemp As Byte
  Local Txcount As Byte
  Local Txtemp1 As Byte
  Txtemp = Serialbuffer(2)
  For Txcount = 1 To Txtemp
    Txtemp1 = Serialbuffer(txcount)
    Print Chr(txtemp1);
Waitms 1
  Next
End Sub
' cambio datos mensaje
Sub Updatelcd()
Cls
        Locate 1 , 1
        For I = 1 To 15
          If Datamsg(i) = 0 Then Exit For
           Lcd Chr(datamsg(i));
        Next
        Locate 2 , 1 : Lcd Numbermsg
        Locate 2 , 5
        If Timemsg(1) <&H30 Or Timemsg(1) >&H39 Then
          For Temp3 = 1 To 6
Timemsg(temp3) = "-"
          Next
        End If
        Lcd Chr(timemsg(1)) : Lcd Chr(timemsg(2)) : Lcd ":"       'hour
        Lcd Chr(timemsg(3)) : Lcd Chr(timemsg(4)) : Lcd ":"       'minutos
        Lcd Chr(timemsg(5)) : Lcd Chr(timemsg(6)) :     'seconds
End Sub
' configuracion de mensaje numero a lamacenar y hora de salida
Sub Readmsg()
  Temp2 = 0
  Temp3 = 0
  Cls
  Cursor On Blink
  Shiftcursor Right
  Posmsg = 0
  Cursorlcd = 1
  Numbermsg = 1
  Fdata = 0
   Call Clrdata()
```

```
  Call Readdata()
  Call Txdata()
  Call Datawait()
  Call Receivecommand()
  If Ferror = 1 Then
    Cls
Lcd "TX error"
    Waitms 700
    Exit Sub
  End If
    Call Updatelcd()
  Do
    Call Keycheck()
    If Key = Keyc Then
      Cls
      Exit Do
    End If
    Select Case Pos
    Case 1                                  ' posicion mensaje
      If Key < 10 Then
        Call Keytable()
        Locate 1 , Cursorlcd
        Temp3 = Cursorlcd + Posmsg
Datamsg(temp3) = Key
        Lcd Chr(key)
        Locate 1 , Cursorlcd
      End If
      If Key = Asterisco Then
        Incr Modeletter
        If Modeletter > 1 Then Modeletter = 0
      End If
      If Swright = 0 Then
        Waitms 100
        Temp3 = Cursorlcd + Posmsg
        If Datamsg(temp3) = 0 Then Datamsg(temp3) = " "
        If Cursorlcd <= 14 Then
          Incr Cursorlcd
        Else
          If Temp3 < Lengthmsgmax Then Incr Posmsg
        End If
        Cls
        For I = 1 To 15
```

```
        Temp3 = I + Posmsg
        If Datamsg(temp3) = 0 Then
          Exit For
        End If
        Lcd Chr(datamsg(temp3));
      Next
      Locate 2 , 1 : Lcd Numbermsg
      Locate 2 , 5
      Lcd Chr(timemsg(1)) : Lcd Chr(timemsg(2)) : Lcd ":"       'hour
      Lcd Chr(timemsg(3)) : Lcd Chr(timemsg(4)) : Lcd ":"       'minutos
      Lcd Chr(timemsg(5)) : Lcd Chr(timemsg(6)) :     'segundos
      Locate 1 , Cursorlcd
    End If
    If Swleft = 0 Then
      Waitms 100
     If Posmsg = 0 Then
      If Cursorlcd > 1 Then Decr Cursorlcd
     Else
      If Posmsg > 0 Then Decr Posmsg
     End If
      Cls
      For I = 1 To 15
        Temp3 = I + Posmsg
        If Datamsg(temp3) = 0 Then Exit For
        Lcd Chr(datamsg(temp3));
      Next
      Locate 2 , 1 : Lcd Numbermsg
      Locate 2 , 5
      Lcd Chr(timemsg(1)) : Lcd Chr(timemsg(2)) : Lcd ":"       'hour
      Lcd Chr(timemsg(3)) : Lcd Chr(timemsg(4)) : Lcd ":"       'minutos
      Lcd Chr(timemsg(5)) : Lcd Chr(timemsg(6)) :     'segundos
      Locate 1 , Cursorlcd
    End If
  Case 2                                  ' numero mensaje
    If Swright = 0 Then
      Waitms 100
      Incr Numbermsg
      If Numbermsg > 10 Then Numbermsg = 10
      Posmsg = 0
      Cursorlcd = 1
      Fdata = 0
      Call Clrdata()
```

```
          Call Clrbuffer()
          Call Readdata()
          Call Txdata()
          Call Datawait()
          If Ferror = 1 Then
            Cls
Lcd "TX error"
            Waitms 700
            Exit Sub
          End If
          Call Receivecommand()
          Call Updatelcd()
          Locate 2 , 1 : Lcd Numbermsg
        End If
        If Swleft = 0 Then
          Waitms 100
          Decr Numbermsg
          If Numbermsg = 0 Then Numbermsg = 1
          Posmsg = 0
          Cursorlcd = 1
          Fdata = 0
          Call Clrbuffer()
          Call Clrdata()
          Call Readdata()
          Call Txdata()
          Call Datawait()
          If Ferror = 1 Then
            Cls
Lcd "TX error"
            Waitms 700
            Exit Sub
          End If
          Call Receivecommand()
          Call Updatelcd()
          Locate 2 , 1 : Lcd Numbermsg
        End If
      Case 3                                    ' hora mensaje
        If Key < 10 Then
          Key = Key + &H30
          If Timepos = 5 Then Timemsg(1) = Key
          If Timepos = 6 Then Timemsg(2) = Key
          If Timepos = 8 Then Timemsg(3) = Key
```

```
            If Timepos = 9 Then Timemsg(4) = Key
            If Timepos = 11 Then Timemsg(5) = Key
            If Timepos = 12 Then Timemsg(6) = Key
            Locate 2 , Timepos
            Lcd Chr(key)
            Locate 2 , Timepos
          End If
          If Swright = 0 Then
            Incr Timepos
            If Timepos = 7 Then Timepos = 8
            If Timepos = 10 Then Timepos = 11
            If Timepos = 13 Then Timepos = 12
            Locate 2 , Timepos
            Waitms 200
          End If
          If Swleft = 0 Then
            Decr Timepos
            If Timepos = 4 Then Timepos = 5
            If Timepos = 7 Then Timepos = 6
            If Timepos = 10 Then Timepos = 9
            Locate 2 , Timepos
            Waitms 200
          End If
          If Swdel = 0 Then
            For Temp3 = 1 To 6
Timemsg(temp3) = "-"
            Next
            Locate 2 , 5
            Lcd Chr(timemsg(1)) : Lcd Chr(timemsg(2)) : Lcd ":"       'hour
            Lcd Chr(timemsg(3)) : Lcd Chr(timemsg(4)) : Lcd ":"       'minutos
            Lcd Chr(timemsg(5)) : Lcd Chr(timemsg(6)) :     'segundos
            Locate 2 , Timepos
            Waitms 100
          End If
        End Select
        If Swsave = 0 Then
         Call Writedata()
         Call Txdata()
        End If
        If Swup = 0 Then
          Incr Pos
          If Pos > 3 Then Pos = 1
```

```
        Call Updateposmsg()
        Waitms 250
      End If
      If Swdown = 0 Then
        Decr Pos
        If Pos = 0 Then Pos = 3
        Call Updateposmsg()
        Waitms 250
      End If
    Loop
End Sub
' actualiza posicion dato en panatalla lcd
Sub Updateposmsg()
    Select Case Pos
    Case 1
Xpos = 1
      Ypos = 1
    Case 2
      Xpos = 1
      Ypos = 2
Case 3
      Xpos = 5
      Ypos = 2
      Timepos = 5
    End Select
Locate Ypos , Xpos
End Sub
' peticion de la hora actual
Sub Readtime()
Serialbuffer(1) = &HFE
Serialbuffer(2) = 5
Serialbuffer(3) = "T"
Serialbuffer(4) = "R"
Serialbuffer(5) = Crc(4)
Call Txdata()
End Sub
' muestra la hora actual
Sub Showtime()
Temp7 = Makebcd(seconds)
    Temp2 = Makebcd(minutes)
    Temp3 = Makebcd(hour)
    Temp4 = Makebcd(datem)
```

```
  Temp5 = Makebcd(month)
  Temp6 = Makebcd(year)
  Locate 1 , 1
  Lcd Hex(temp6) : Lcd "/"
  Lcd Hex(temp5) : Lcd "/"
  Lcd Hex(temp4) : Lcd " "
  If Weekday = 2 Then Lcd "Lunes"
  If Weekday = 3 Then Lcd "Martes"
  If Weekday = 4 Then Lcd "Mierco"
  If Weekday = 5 Then Lcd "Jueves"
  If Weekday = 6 Then Lcd "Viernes"
  If Weekday = 7 Then Lcd "Sabado"
  If Weekday = 1 Then Lcd "Domingo"
  Locate 2 , 1
  Lcd Hex(temp3) : Lcd ":"
  Lcd Hex(temp2) : Lcd ":"
  Lcd Hex(temp7)
End Sub
'espera la recepcion del dato tras la preticion de un dato
Sub Datawait()
  Ferror = 0
  While Fdata = 0
    Incr Delaywait
    If Delaywait > Delaywaitmax Then
      Delaywait = 0
      Ferror = 1
    End If
  Wend
End Sub
' procdesa dato recivido por el puerto uart
Sub Receivecommand()
  Local Datacounter As Byte
  Local Datalength As Byte
  If Serialbuffer(3) = "M" And Serialbuffer(4) = "D" Then
    Numbermsg = Serialbuffer(5)
    Datalength = Serialbuffer(2) - 13
Timemsg(1) = Serialbuffer(6)                    ' hora
Timemsg(2) = Serialbuffer(7)                    '
Timemsg(3) = Serialbuffer(8)                    ' minutos
Timemsg(4) = Serialbuffer(9)                    '
Timemsg(5) = Serialbuffer(10)                    ' segundos
Timemsg(6) = Serialbuffer(11)                    '
```

```
    For Datacounter = 1 To Datalength
      If Serialbuffer(datacounter + 11) = 0 Or Serialbuffer(datacounter + 11) = 255 Then Exit For
Datamsg(datacounter) = Serialbuffer(datacounter + 11)
    Next
  End If
  If Serialbuffer(3) = "T" And Serialbuffer(4) = "D" Then
    Year = Serialbuffer(5)
    Month = Serialbuffer(6)
    Datem = Serialbuffer(7)
    Hour = Serialbuffer(8)
    Minutes = Serialbuffer(9)
    Seconds = Serialbuffer(10)
    Weekday = Serialbuffer(11)
  End If
End Sub
' envia nueva fecha y hora
Sub Writetime()
Serialbuffer(1) = &HFE
Serialbuffer(2) = 12
Serialbuffer(3) = "T"
Serialbuffer(4) = "W"
Serialbuffer(5) = Year
Serialbuffer(6) = Month
Serialbuffer(7) = Datem
Serialbuffer(8) = Hour
Serialbuffer(9) = Minutes
Serialbuffer(10) = Seconds
Serialbuffer(11) = Weekday
Serialbuffer(12) = Crc(11)
  Call Txdata()
End Sub
'configura nueva hora y fecha
Sub Settime()
I = 1
Ypos = 1
  Cls
  Cursor On Noblink
  Updatetime:
  Update1lcd:
  Call Showtime()
  If Ypos = 1 Then Locate 1 , 1
  If Ypos = 2 Then Locate 2 , 1
```

```
  Update1:
  Call Keycheck()
  If Key = Keya Then                    ' incrementa dia
    Incr Weekday
    If Weekday > 7 Then Weekday = 1
    Cls
    Goto Updatetime
  End If
  If Key = Keyb Then
    Incr Ypos
    If Ypos > 2 Then Ypos = 1
    Goto Update1lcd
  End If
  If Key = Keyd Then                    ' actualiza hora
    Call Writetime()
    Cls
    Goto Finsettime
  End If
  If Key = Keyc Then
    Goto Finsettime
  End If
  If Key < 10 Then
    Temp1 = Key
    If Ypos = 2 Then
      Minutes = Makebcd(minutes)
      Seconds = Makebcd(seconds)
      Hour = Makebcd(hour)
      If I = 1 Then
Shift , Temp1 , Left , 4 : Hour = Hour And &H0F : Hour = Temp1 Or Hour
      End If
      If I = 2 Then
        Hour = &HF0 And Hour : Hour = Temp1 Or Hour
      End If
      If I = 3 Then
Shift , Temp1 , Left , 4 : Minutes = Minutes And &H0F : Minutes = Temp1 Or Minutes
      End If
      If I = 4 Then
        Minutes = &HF0 And Minutes : Minutes = Temp1 Or Minutes
      End If
      If I = 5 Then
Shift , Temp1 , Left , 4 : Seconds = Seconds And &H0F : Seconds = Temp1 Or Seconds
      End If
```

```
        If I = 6 Then
          Seconds = &HF0 And Seconds : Seconds = Temp1 Or Seconds
        End If
        Minutes = Makedec(minutes)
        Seconds = Makedec(seconds)
        Hour = Makedec(hour)
      End If
      If Ypos = 1 Then
        Year = Makebcd(year)
        Datem = Makebcd(datem)
        Month = Makebcd(month)

        If I = 1 Then
Shift , Temp1 , Left , 4 : Year = Year And &H0F : Year = Temp1 Or Year
        End If
        If I = 2 Then
          Year = &HF0 And Year : Year = Temp1 Or Year
        End If
        If I = 3 Then
Shift , Temp1 , Left , 4 : Month = Month And &H0F : Month = Temp1 Or Month
        End If
        If I = 4 Then
          Month = &HF0 And Month : Month = Temp1 Or Month
        End If
        If I = 5 Then
Shift , Temp1 , Left , 4 : Datem = Datem And &H0F : Datem = Temp1 Or Datem
        End If
        If I = 6 Then
          Datem = &HF0 And Datem : Datem = Temp1 Or Datem
        End If
        Year = Makedec(year)
        Datem = Makedec(datem)
        Month = Makedec(month)
      End If
      Incr I
      If I > 6 Then I = 1
      Goto Update1lcd
    End If
    Goto Update1
    Finsettime:
    Cursor Off Noblink
End Sub
```

```
' graba nueva clave
Sub Setclave()
  Cls
  Locate 1 , 1 : Lcd "Nueva Clave"
Locate 2 , 1
J = 1
  Menu2:
  Call Keycheck()
  If Key < 10 Then
    Lcd Key
Password(j) = Key
    J = J + 1
    If J > 4 Then
      For J = 1 To 4
        Writeeeprom Password(j) , J
      Next
      Goto Fsetclave
    End If
  End If
  If Key = Keyc Then
    Goto Fsetclave
  End If
  Goto Menu2
Fsetclave:
End Sub
'' lee teclado 4*4
Sub Keycheck()
  Key = 16
  Waitms 100
  C1 = 1 : C2 = 1 : C3 = 1 : C4 = 0
  Waitms 1
  If F1 = 0 Then Key = Keya
  If F2 = 0 Then Key = Keyb
  If F3 = 0 Then Key = Keyc
  If F4 = 0 Then Key = Keyd
  If Key < 16 Then Goto Finkey
  C1 = 0 : C2 = 1 : C3 = 1 : C4 = 1
  Waitms 1
  If F1 = 0 Then Key = 1
  If F2 = 0 Then Key = 4
  If F3 = 0 Then Key = 7
  If F4 = 0 Then Key = Asterisco
```

```
  If Key < 16 Then Goto Finkey
  C1 = 1 : C2 = 0 : C3 = 1 : C4 = 1
  Waitms 1
  If F1 = 0 Then Key = 2
  If F2 = 0 Then Key = 5
  If F3 = 0 Then Key = 8
  If F4 = 0 Then Key = 0
  If Key < 16 Then Goto Finkey
  Waitms 1
  C1 = 1 : C2 = 1 : C3 = 0 : C4 = 1
  Waitms 1
  If F1 = 0 Then Key = 3
  If F2 = 0 Then Key = 6
  If F3 = 0 Then Key = 9
  If F4 = 0 Then Key = Numeral
  C1 = 1 : C2 = 1 : C3 = 0 : C4 = 1
Finkey:
  If F1 = 0 Then Goto Finkey
  If F2 = 0 Then Goto Finkey
  If F3 = 0 Then Goto Finkey
  If F4 = 0 Then Goto Finkey
  C1 = 1 : C2 = 1 : C3 = 1 : C4 = 1
  If Key < 16 Then Delayexit = 0
End Sub
' verifica clave ingresa con la clave almacenada
Sub Verify()
    For J = 1 To 4
Password(j) = 255
    Next
    J = 1
    Cls
Locate 1 , 1 : Lcd "Ingrese Clave"
    Locate 2 , 1
Delayexit = 0
    Menu1:
      Incr Delayexit
      If Delayexit > 30 Then
        Fclave = 0
        Goto Fverify
      End If
      Call Keycheck()
      If Key < 10 Then
```

```
Password(j) = Key
' Lcd "*"
          Lcd Key
          J = J + 1
          If J > 4 Then
            Call Clave()
            If Fclave = 1 Then
              Goto Fverify
            Else
              Goto Fverify
            End If
          End If
        End If
        If Key = Numeral Then
          Goto Fverify
        End If
      Goto Menu1
Fverify:
End Sub
'lee clave almacenada
Sub Clave()
  Fclave = 1
  For I = 1 To 4
    Readeeprom Temp3 , I
    If Temp3 <>Password(i) Then
      Fclave = 0
      Exit For
    End If
  Next
End Sub
'vector iinterrupcion timer 1
Tim1_isr:
Ftime = 1
Return
' vector interrupcion de puerto rx de uart
Rec_isr:
  Rxtemp = Udr
  Delaywait = 0
  If Frxdata = 0 Then
    If Rxtemp = &HFE Then
Serialbuffer(1) = Rxtemp
        Rxcounter = 2
```

```
        Frxdata = 1
        Fdata = 0
      End If
    Else
      If Rxcounter = 2 Then Rxlength = Rxtemp
Serialbuffer(rxcounter) = Rxtemp
      If Rxcounter > 2 And Rxlength = Rxcounter Then
        Fdata = 1
        Frxdata = 0
      End If
      Incr Rxcounter
End If
Return
'valoresw por defecto de la memoria eeprom del uc
$eeprom
  Ini_eeprom:
' clave address  01 - 04
  Data &H00 ,&H00 , 0 , 0
  Data 0 , 2 , 0 , 0
  Data 25 , 0 , 0 , 0
  Data 20 , 0 , 40 , 0
  Data 75 , 0 , 100 , 0
  Data 0 , 0 , 0 , 0
  Data 0 , 0 , 0 , 0
  Data 0 , 0 , 0 , 0
Data 0 , 0 , 0 , 0
$data
End.
```

ANEXO 3. DATASHEET DEL MICRO-CONTROLADOR ATMEGA16.

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
  - 16K 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
  - 512 Bytes EEPROM
    Endurance: 100,000 Write/Erase Cycles
  - 1K Byte 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
  - Four PWM Channels
  - 8-channel, 10-bit ADC
    8 Single-ended Channels
    7 Differential Channels in TQFP Package Only
    2 Differential Channels with Programmable Gain at 1x, 10x, or 200x
  - Byte-oriented Two-wire Serial Interface
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
- 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
  - 40-pin PDIP, 44-lead TQFP, and 44-pad MLF
- Operating Voltages
  - 2.7 - 5.5V for ATmega16L
  - 4.5 - 5.5V for ATmega16
- Speed Grades
  - 0 - 8 MHz for ATmega16L
  - 0 - 16 MHz for ATmega16
- Power Consumption @ 1 MHz, 3V, and 25°C for ATmega16L
  - Active: 1.1 mA
  - Idle Mode: 0.35 mA
  - Power-down Mode: < 1 µA

ATMEL®

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

ATmega16
ATmega16L

Summary

2466HS–AVR–12/03

ANEXO 4. DATASHEET DEL AT24C256 (MEMORIA EEPROM).

Features

- Low Voltage and Standard Voltage Operation
  - 5.0 (V_{CC} = 4.5V to 5.5V)
  - 2.7 (V_{CC} = 2.7V to 5.5V)
  - 1.8 (V_{CC} = 1.8V to 3.6V)
- Internally Organized 16,384 x 8 and 32,768 x 8
- 2-Wire Serial Interface
- Schmitt Trigger, Filtered Inputs for Noise Suppression
- Bidirectional Data Transfer Protocol
- 1 MHz (5V), 400 kHz (2.7V) and 100 kHz (1.8V) Compatibility
- Write Protect Pin for Hardware and Software Data Protection
- 64-Byte Page Write Mode (Partial Page Writes Allowed)
- Self-Timed Write Cycle (5 ms typical)
- High Reliability
  - Endurance: 100,000 Write Cycles
  - Data Retention: 40 Years
  - ESD Protection: > 4000V
- Automotive Grade and Extended Temperature Devices Available
- 8-Pin JEDEC PDIP, 8-Pin JEDEC and EIAJ SOIC, 14-Pin TSSOP, and 8-Pin Leadless Array Packages

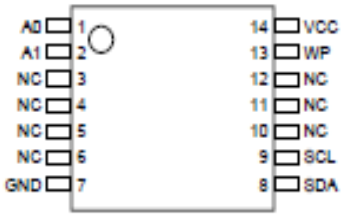
Description

The AT24C128/256 provides 131,072/262,144 bits of serial electrically erasable and programmable read only memory (EEPROM) organized as 16,384/32,768 words of 8 bits each. The device's cascadable feature allows up to 4 devices to share a common 2-wire bus. The device is optimized for use in many industrial and commercial applications where low power and low voltage operation are essential. The devices are available in space-saving 8-pin JEDEC PDIP, 8-pin EIAJ, 8-pin JEDEC SOIC, 14-pin TSSOP, and 8-pin LAP packages. In addition, the entire family is available in 5.0V (4.5V to 5.5V), 2.7V (2.7V to 5.5V) and 1.8V (1.8V to 3.6V) versions.

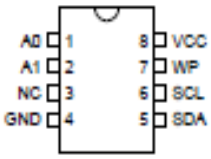
Pin Configurations

Pin Name	Function
A_0 to A_1	Address Inputs
SDA	Serial Data
SCL	Serial Clock Input
WP	Write Protect
NC	No Connect

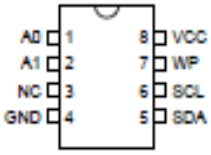
14-Pin TSSOP



8-Pin PDIP



8-Pin SOIC



8-Pin Leadless Array

Bottom View

ATMEL

2-Wire Serial EEPROMs

128K (16,384 x 8)

256K (32,768 x 8)

AT24C128
AT24C256

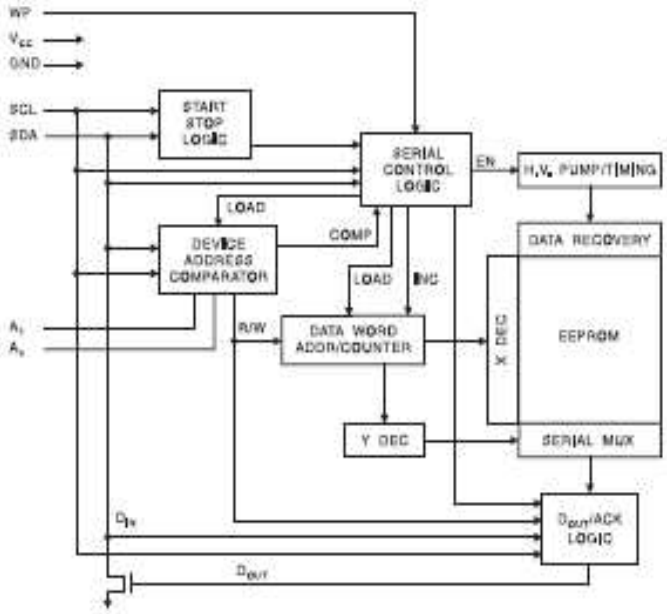
Rev. 0670C–08/98

Absolute Maximum Ratings*

Operating Temperature	-55°C to +125°C
Storage Temperature	-65°C to +150°C
Voltage on Any Pin with Respect to Ground	-1.0V to +7.0V
Maximum Operating Voltage	6.25V
DC Output Current	5.0 mA

*NOTICE: Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

Block Diagram



Pin Description

SERIAL CLOCK (SCL): The SCL input is used to positive edge clock data into each EEPROM device and negative edge clock data out of each device.

SERIAL DATA (SDA): The SDA pin is bidirectional for serial data transfer. This pin is open-drain driven and may be wire-ORed with any number of other open-drain or open collector devices.

DEVICE/PAGE ADDRESSES (A1, A0): The A1 and A0 pins are device address inputs that are hardwired or left not connected for hardware compatibility with AT24C32/64. When the pins are hardwired, as many as four 128K/256K devices may be addressed on a single bus system (device addressing is discussed in detail under the Device Addressing section). When the pins are not hardwired, the default A_1 and A_0 are zero.

WRITE PROTECT (WP): The write protect input, when tied to GND, allows normal write operations. When WP is tied high to V_{CC}, all write operations to the memory are inhibited. If left unconnected, WP is internally pulled down to GND. Switching WP to V_{CC} prior to a write operation creates a software write protect function.

Memory Organization

AT24C128/256, 128K/256K SERIAL EEPROM: The 128K/256K is internally organized as 256/512 pages of 64-bytes each. Random word addressing requires a 14/15-bit data word address.

Pin Capacitance[1]

Applicable over recommended operating range from T_A = 25°C, f = 1.0 MHz, V_{CC} = +1.8V.

Symbol	Test Condition	Max	Units	Conditions
$C_{I/O}$	Input/Output Capacitance (SDA)	8	pF	$V_{I/O}$ = 0V
C_{IN}	Input Capacitance (A_0, A_1, SCL)	6	pF	V_{IN} = 0V

Note: This parameter is characterized and is not 100% tested.

DC Characteristics

Applicable over recommended operating range from: T_{AI} = -40°C to +85°C, V_{CC} = +1.8V to +5.5V, T_{AC} = 0°C to +70°C, V_{CC} = +1.8V to +5.5V (unless otherwise noted).

Symbol	Parameter		Test Condition	Min	Typ	Max	Units
V_{CC1}	Supply Voltage			1.8		3.6	V
V_{CC2}	Supply Voltage			2.7		5.5	V
V_{CC3}	Supply Voltage			4.5		5.5	V
I_{CC1}	Supply Current	V_{CC} = 5.0V	READ at 400 kHz		1.0	2.0	mA
I_{CC2}	Supply Current	V_{CC} = 5.0V	WRITE at 400 kHz		2.0	3.0	mA
I_{SB1}	Standby Current (1.8V option)	V_{CC} = 1.8V	V_{IN} = V_{CC} or V_{SS}			0.2	µA
		V_{CC} = 3.6V				2.0	
I_{SB2}	Standby Current (2.7V option)	V_{CC} = 2.7V	V_{IN} = V_{CC} or V_{SS}			0.5	µA
		V_{CC} = 5.5V				6.0	
I_{SB3}	Standby Current (5.0V option)	V_{CC} = 4.5 - 5.5V	V_{IN} = V_{CC} or V_{SS}			6.0	µA
I_{LI}	Input Leakage Current		V_{IN} = V_{CC} or V_{SS}		0.10	3.0	µA
I_{LO}	Output Leakage Current		V_{OUT} = V_{CC} or V_{SS}		0.05	3.0	µA
V_{IL}	Input Low Level[Note:]			-0.6		V_{CC} x 0.3	V
V_{IH}	Input High Level[Note:]			V_{CC} x 0.7		V_{CC} + 0.5	V
V_{OL2}	Output Low Level	V_{CC} = 3.0V	I_{OL} = 2.1 mA			0.4	V
V_{OL1}	Output Low Level	V_{CC} = 1.8V	I_{OL} = 0.15 mA			0.2	V

Note: V_{IL} min and V_{IH} max are reference only and are not tested

AC Characteristics

Applicable over recommended operating range from T_A = -40°C to +85°C, V_{CC} = +1.8V to +5.5V, CL = 100 pF (unless otherwise noted). Test conditions are listed in Note 2.

Symbol	Parameter	1.8-volt		2.7-volt		5.0-volt		Units
		Min	Max	Min	Max	Min	Max	
f_{SCL}	Clock Frequency, SCL		100		400		1000	kHz
t_{LOW}	Clock Pulse Width Low	4.7		1.3		0.6		µs
t_{HIGH}	Clock Pulse Width High	4.0		1.0		0.4		µs
t_{AA}	Clock Low to Data Out Valid	0.1	4.5	0.05	0.9	0.05	0.55	µs
t_{BUF}	Time the bus must be free before a new transmission can start[1]	4.7		1.3		0.5		µs
$t_{HD.STA}$	Start Hold Time	4.0		0.6		0.25		µs
$t_{SU.STA}$	Start Set-up Time	4.7		0.6		0.25		µs
$t_{HD.DAT}$	Data In Hold Time	0		0		0		µs
$t_{SU.DAT}$	Data In Set-up Time	200		100		100		ns
t_R	Inputs Rise Time[1]		1.0		0.3		0.3	µs
t_F	Inputs Fall Time[1]		300		300		100	ns
$t_{SU.STO}$	Stop Set-up Time	4.7		0.6		0.25		µs
t_{DH}	Data Out Hold Time	100		50		50		ns
t_{WR}	Write Cycle Time		20		10		10	ms
Endurance[1]	5.0V, 25°C, Page Mode	100K		100K		100K		Write Cycles

Notes: 1. This parameter is characterized and is not 100% tested.

2. AC measurement conditions:
   R_L (connects to V_{CC}): 1.3KΩ (2.7V, 5V), 10KΩ (1.8V)
   Input pulse voltages: $0.3V_{CC}$ to $0.7V_{CC}$
   Input rise and fall times: ≤50ns
   Input and output timing reference voltages: $0.5V_{CC}$

Device Operation

CLOCK and DATA TRANSITIONS: The SDA pin is normally pulled high with an external device. Data on the SDA pin may change only during SCL low time periods (refer to Data Validity timing diagram). Data changes during SCL high periods will indicate a start or stop condition as defined below.

START CONDITION: A high-to-low transition of SDA with SCL high is a start condition which must precede any other command (refer to Start and Stop Definition timing diagram).

STOP CONDITION: A low-to-high transition of SDA with SCL high is a stop condition. After a read sequence, the stop command will place the EEPROM in a standby power mode (refer to Start and Stop Definition timing diagram).

ACKNOWLEDGE: All addresses and data words are serially transmitted to and from the EEPROM in 8-bit words. The EEPROM sends a zero during the ninth clock cycle to acknowledge that it has received each word.

STANDBY MODE: The AT24C128/256 features a low power standby mode which is enabled: a) upon power-up and b) after the receipt of the STOP bit and the completion of any internal operations.

MEMORY RESET: After an interruption in protocol, power loss or system reset, any 2-wire part can be reset by following these steps: (a) Clock up to 9 cycles, (b) look for SDA high in each cycle while SCL is high and then (c) create a start condition as SDA is high.

ANEXO 5. DATASHEET DEL DS1307 (RELOJ EN TIEMPO REAL).

DALLAS SEMICONDUCTOR

DS1307
64 x 8 Serial Real-Time Clock

www.maxim-ic.com

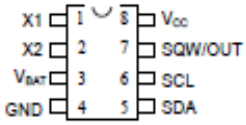
FEATURES

- 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

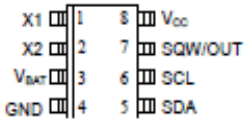
ORDERING INFORMATION

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)

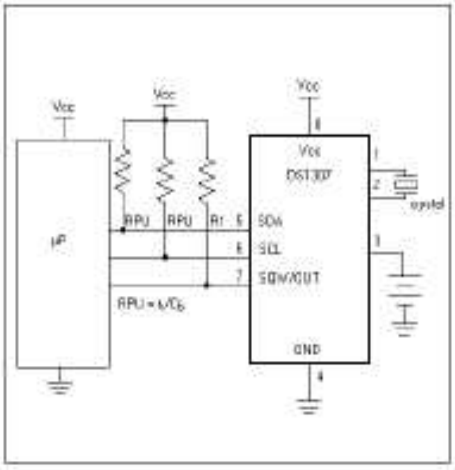
PIN DESCRIPTION

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.

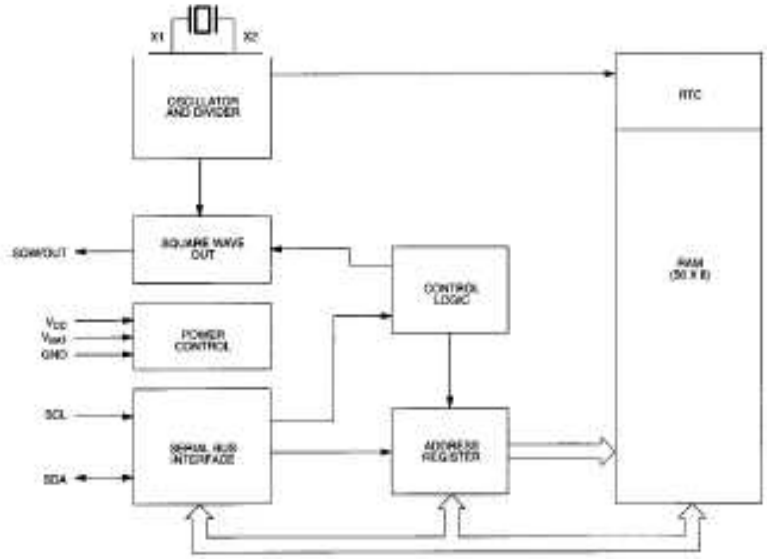
TYPICAL OPERATING CIRCUIT



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



ANEXO 6. DATASHEET DEL DS75176B

National Semiconductor

July 2004

DS75176B/DS75176BT
Multipoint RS-485/RS-422 Transceivers

General Description

The DS75176B is a high speed differential TRI-STATE® bus/line transceiver designed to meet the requirements of EIA standard RS485 with extended common mode range (+12V to −7V), for multipoint data transmission. In addition, it is compatible with RS-422.

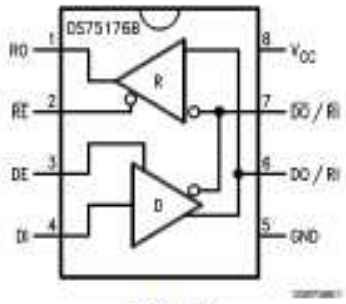
The driver and receiver outputs feature TRI-STATE capability, for the driver outputs over the entire common mode range of +12V to −7V. Bus contention or fault situations that cause excessive power dissipation within the device are handled by a thermal shutdown circuit, which forces the driver outputs into the high impedance state.

DC specifications are guaranteed over the 0 to 70°C temperature and 4.75V to 5.25V supply voltage range.

Features

- Meets EIA standard RS485 for multipoint bus transmission and is compatible with RS-422.
- Small Outline (SO) Package option available for minimum board space.
- 22 ns driver propagation delays.
- Single +5V supply.
- −7V to +12V bus common mode range permits ±7V ground difference between devices on the bus.
- Thermal shutdown protection.
- High impedance to bus with driver in TRI-STATE or with power off, over the entire common mode range allows the unused devices on the bus to be powered down.
- Pin out compatible with DS3695/A and SN75176A/B.
- Combined impedance of a driver output and receiver input is less than one RS485 unit load, allowing up to 32 transceivers on the bus.
- 70 mV typical receiver hysteresis.

Connection and Logic Diagram



Top View

Order Number DS75176BN, DS75176BTN, DS75176BM or DS75176BTM

See NS Package Number N08E or M08A

Absolute Maximum Ratings (Note 1)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/ Distributors for availability and specifications.

Supply Voltage, V_{CC}	7V
Control Input Voltages	7V
Driver Input Voltage	7V
Driver Output Voltages	+15V/ −10V
Receiver Input Voltages (DS75176B)	+15V/ −10V
Receiver Output Voltage	5.5V
Continuous Power Dissipation @ 25°C	
for M Package	675 mW (Note 5)
for N Package	900 mW (Note 4)
Storage Temperature Range	−65°C to +150°C
Lead Temperature (Soldering, 4 seconds)	260°C
ESD Rating (HBM)	500V

Recommended Operating Conditions

	Min	Max	Units
Supply Voltage, V_{CC}	4.75	5.25	V
Voltage at Any Bus Terminal (Separate or Common Mode)	−7	+12	V
Operating Free Air Temperature T_A			
DS75176B	0	+70	°C
DS75176BT	−40	+85	°C
Differential Input Voltage, VID (Note 6)	−12	+12	V

Electrical Characteristics (Notes 2, 3)

0°C ≤ T_A ≤ 70°C, 4.75V < V_{CC} < 5.25V unless otherwise specified

Symbol	Parameter		Conditions		Min	Typ	Max	Units
V_{OD1}	Differential Driver Output Voltage (Unloaded)		I_O = 0				5	V
V_{OD2}	Differential Driver Output Voltage (with Load)		(*Figure 1*)	R = 50Ω; (RS-422) (Note 7)	2			V
				R = 27Ω; (RS-485)	1.5			V
ΔV_{OD}	Change in Magnitude of Driver Differential Output Voltage For Complementary Output States		(*Figure 1*)	R = 27Ω			0.2	V
V_{OC}	Driver Common Mode Output Voltage						3.0	V
$\lvert\Delta V_{OC}\rvert$	Change in Magnitude of Driver Common Mode Output Voltage For Complementary Output States						0.2	V
V_{IH}	Input High Voltage	DI, DE, $\overline{RE}$, E			2			V
V_{IL}	Input Low Voltage						0.8	
V_{CL}	Input Clamp Voltage			I_{IN} = −18 mA			−1.5	
I_{IL}	Input Low Current			V_{IL} = 0.4V			−200	µA
I_{IH}	Input High Current			V_{IH} = 2.4V			20	µA
I_{IN}	Input Current	DO/RI, $\overline{DO/RI}$	V_{CC} = 0V or 5.25V, DE = 0V	V_{IN} = 12V			+1.0	mA
				V_{IN} = −7V			−0.8	mA
V_{TH}	Differential Input Threshold Voltage for Receiver		−7V ≤ V_{CM} ≤ +12V		−0.2		+0.2	V
ΔV_{TH}	Receiver Input Hysteresis		V_{CM} = 0V			70		mV
V_{OH}	Receiver Output High Voltage		I_{OH} = −400 µA		2.7			V
V_{OL}	Output Low Voltage	RO	I_{OL} = 16 mA (Note 7)				0.5	V
I_{OZR}	OFF-State (High Impedance) Output Current at Receiver		V_{CC} = Max, 0.4V ≤ V_O ≤ 2.4V				±20	µA
R_{IN}	Receiver Input Resistance		−7V ≤ V_{CM} ≤ +12V		12			kΩ
I_{CC}	Supply Current		No Load (Note 7)	Driver Outputs Enabled			55	mA
				Driver Outputs Disabled			35	mA

Electrical Characteristics (Notes 2, 3) (Continued)

$0°C \le T_A \le 70°C$, $4.75V < V_{CC} < 5.25V$ unless otherwise specified

Symbol	Parameter	Conditions	Min	Typ	Max	Units
I_{OSD}	Driver Short-Circuit Output Current	$V_O = -7V$ (Note 7)			−250	mA
		$V_O = +12V$ (Note 7)			+250	mA
I_{OSR}	Receiver Short-Circuit Output Current	$V_O = 0V$	−15		−85	mA

Note 1: "Absolute Maximum Ratings" are those beyond which the safety of the device cannot be guaranteed. They are not meant to imply that the device should be operated at these limits. The tables of "Electrical Characteristics" provide conditions for actual device operation.

Note 2: All currents into device pins are positive; all currents out of device pins are negative. All voltages are referenced to device ground unless otherwise specified.

Note 3: All typicals are given for $V_{CC} = 5V$ and $T_A = 25°C$.

Note 4: Derate linearly at 5.56 mW/°C to 650 mW at 70°C.

Note 5: Derate linearly @ 6.11 mW/°C to 400 mW at 70°C.

Note 6: Differential - Input/Output bus voltage is measured at the noninverting terminal A with respect to the inverting terminal B.

Note 7: All worst case paramaters for which note 7 is applied, must be increased by 10% for DS75176BT. The other paramaters remain valid for $-40°C < T_A < +85°C$.

Switching Characteristics

$V_{CC} = 5.0V$, $T_A = 25°C$

Symbol	Parameter	Conditions	Min	Typ	Max	Units
t_{PLH}	Driver Input to Output	$R_{DIFF} = 60\Omega$		12	22	ns
t_{PHL}	Driver Input to Output	$C_{L1} = C_{L2} = 100$ pF		17	22	ns
t_r	Driver Rise Time	$R_{DIFF} = 60\Omega$			18	ns
t_f	Driver Fall Time	$C_{L1} = C_{L2} = 100$ pF (*Figure 3* and *Figure 5*)			18	ns
t_{ZH}	Driver Enable to Output High	$C_L = 100$ pF (*Figure 4* and *Figure 6*) S1 Open		29	100	ns
t_{ZL}	Driver Enable to Output Low	$C_L = 100$ pF (*Figure 4* and *Figure 6*) S2 Open		31	60	ns
t_{LZ}	Driver Disable Time from Low	$C_L = 15$ pF (*Figure 4* and *Figure 6*) S2 Open		13	30	ns
t_{HZ}	Driver Disable Time from High	$C_L = 15$ pF (*Figure 4* and *Figure 6*) S1 Open		19	200	ns
t_{PLH}	Receiver Input to Output	$C_L = 15$ pF (*Figure 2* and *Figure 7*) S1 and S2 Closed		30	37	ns
t_{PHL}	Receiver Input to Output			32	37	ns
t_{ZL}	Receiver Enable to Output Low	$C_L = 15$ pF (*Figure 2* and *Figure 8*) S2 Open		15	20	ns
t_{ZH}	Receiver Enable to Output High	$C_L = 15$ pF (*Figure 2* and *Figure 8*) S1 Open		11	20	ns
t_{LZ}	Receiver Disable from Low	$C_L = 15$ pF (*Figure 2* and *Figure 8*) S2 Open		28	32	ns
t_{HZ}	Receiver Disable from High	$C_L = 15$ pF (*Figure 2* and *Figure 8*) S1 Open		13	35	ns