ESCUELA POLITECNICA NACIONAL

ESCUELA DE INGENIERIA

DISEÑO DE UN PROTOTIPO DE BAJO COSTO QUE PERMITA MOSTRAR INFORMACIÓN DE LA RUTA A LOS USUARIOS DE UNA UNIDAD DE TRANSPORTE

PROYECTO PREVIO A LA OBTENCION DEL TITULO DE INGENIERO EN ELECTRONICA Y TELECOMUNICAIONES

RICARDO MAURICIO ZHINGRE AYALA

DIRECTOR: ING. FERNANDO VÁSQUEZ

Quito, Marzo 2007

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Ricardo Mauricio Zhingre Ayala, bajo mi supervisión.

Ing. Fernando Vásquez
DIRECTOR DE PROYECTO

AGRADECIMIENTOS

Agradezco en primer lugar al Señor porque a pesar de mi, me ha acompañado y ha sido la ayuda en todo este proceso.

A mis padres, quienes con su paciencia y apoyo en todos los sentidos son los que han soportado todo mí caminar hasta ver el fin del esfuerzo en esta tesis.

A mi hermano por su ayuda en todo este trabajo.

A mi esposa y a mi hija por su compañía.

A mi tutor por su paciencia y exigencia.

Y a mis amigos, que en los momentos oportunos has sabido mostrar su ayuda.

DEDICATORIA

Dedico este trabajo a mis padres, hermano, esposa e hija, por su apoyo incondicional en todo este proceso.

TABLA DE CONTENIDO

RESUMEN

Es necesario comprender que el sistema de transporte en el Ecuador necesita de soluciones que puedan ser asequibles a la gran mayoría de este sector para de esta manera combatir el descuido que esta área de servicio público sufre.

Este proyecto se enfoca en brindar una solución a la necesidad de información que tienen los usuarios de las unidades de transporte en especial a las personas que utilizan el servicio interprovincial, ya que en su mayoría son turistas, que serían los beneficiarios de implementarse esta clase de servicio, utilizando para esto la tecnología disponible al integrar a un microcontrolador como es el AVR la información que provee el sistema GPS para brindar esta solución.

Lo que trata de hacer este prototipo es presentar una forma de resolver este problema para que pudiese llegar a ser implementada en las unidades y ofrecer una alternativa en información para las rutas de nuestro país de tal forma que la calidad del servicio de transporte se eleve.

En cuanto al aspecto técnico este trabajo integra la información del sistema GPS a un microcontrolador para tener como resultado información que se muestre en la pantalla de una televisión sin interrumpir el uso que los ocupantes de la unidad de trasporte hacen de ella. Esta información está predefinida antes, y mostrará el lugar dónde se encuentra la unidad, la velocidad a la que viaja y el tiempo que falta para llegar al destino final cuando se localice un punto dentro de la ruta por la cual se desplaza.

PRESENTACION

El principal interés de este proyecto se concentra en brindar una solución a la necesidad de información que tienen los usuarios de las unidades de transporte en especial a las personas que utilizan el servicio interprovincial, ya que en su mayoría son turistas, que aprovecharían este servicio.

En el presente trabajo se desarrolla el diseño y la construcción de un prototipo que permite mostrar información a los usuarios de una unidad de transporte utilizando para esto una pantalla de televisión.

El capítulo 1 da un marco teórico a lo que será el prototipo dando una visión general de lo que es el AVR, que es el microcontrolador usado en este proyecto. Junto con esto se muestra el funcionamiento del sistema GPS y en especial se describe el estándar NMEA, adicionalmente se revisa los principios de funcionamiento de la televisión para entender como se generan las imágenes en la pantalla y después se explica el principio de la generación digital de caracteres.

El capítulo 2 muestra el diseño, tanto del hardware como del software para este prototipo.

El capítulo 3 documenta el proceso de implementación del hardware como del software para el proyecto.

El capítulo 4 muestra por medio de un ejemplo muestra la utilización de este prototipo en una ruta establecida para el efecto de esta prueba. También se presenta el costo de este prototipo.

En el capítulo 5 se presentan conclusiones y recomendaciones acerca de este proyecto.

En el anexo 1 se encuentra un resumen del las especificaciones del AVR Atemega16.

El anexo 2 muestra las especificaciones técnicas del receptor GPS utilizado.

En el anexo 3 se encuentra la información sobre el estándar NMEA, con el formato de la misma y los tipos de tramas que ocupa con su respectiva descripción.

El anexo 4 contiene la especificación técnica del circuito integrado LM1881.

El anexo 5 contiene la tabla de los caracteres creados para este proyecto.

Las referencias bibliográficas que se indican en el texto se han colocado junto a los títulos o subtítulos entre corchetes.

CAPÍTULO 1. DESCRIPCION DE LOS COMPONENTES PRINCIPALES DEL PROTOTIPO

1.1 INTRODUCCION

En este capítulo se describe de manera concreta el funcionamiento de los elementos que se utilizan en este prototipo, teniendo en cuenta sus características básicas y su utilidad de una forma sencilla pero clara cada elemento utilizado.

Se describe en términos generales el funcionamiento de los microcontroladores AVR, que para fines del prototipo a construirse constituye la base del mismo. Dependiendo de las características requeridas se escoge uno en especial, teniendo en cuenta principalmente a los AVR de la serie Mega por su capacidad de almacenamiento.

A continuación se hace una exposición del funcionamiento del sistema GPS y la clase de información que entrega, describiendo el protocolo con el cual la información es tratada.

Continuando con la exposición se menciona cómo se generan las imágenes en una pantalla de televisión tendiendo en cuenta las características de la señal que se ha de generar por el prototipo, de tal forma que se logre exhibir caracteres en una pantalla de televisión.

Por último se da una idea de cómo se logra la generación de los caracteres de manera digital y a su vez cómo éstos se muestran en la pantalla de televisión, teniendo en cuenta los aspectos necesarios para que la señal digital sea reconocida por un dispositivo analógico como es un televisor.

1.2 CARACTERISTICAS GENERALES DE LOS AVR [1][2][3][4][5][1]

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

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

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

El set de instrucciones de los AVR es más regular que el de la mayoría de los microcontroladores de 8-bits, por ejemplo los PIC's, teniendo en cuenta para los AVR's lo siguiente:

[1] Se indica la bibliografía que es utilizada al final del trabajo.

- Los registros punteros X, Y y Z tienen capacidades de direccionamiento diferentes entre sí.
- Los registros 0 al 15 tienen diferentes capacidades de direccionamiento que los registros 16 al 31.
- Las registros de I/O 0 al 31 tienen distintas características que las posiciones 32 al 63.
- La instrucción CLR afecta los 'flag', mientras que la instrucción SER[2] no lo hace, a pesar de que parecen ser instrucciones complementarias (dejar todos los bits en 0, y dejar todos los bits en 1 respectivamente).

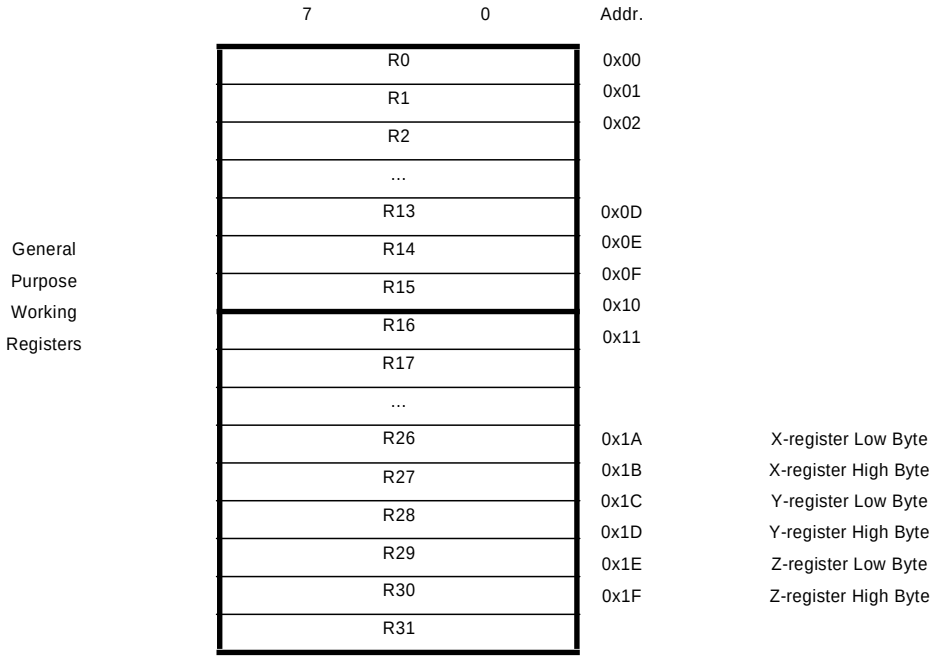


Figure 1.1. Registros de Propósito General del AVR

En la Figura 1.1 se muestra la distribución de los registros de propósito general del AVR, junto con las direcciones de memoria y donde se ubican los registros X, Y y Z dentro de un microcontrolador.

[2] Instrucción en lenguaje Assembler.

1.2.1 Arquitectura RISC

El hecho de que tenga una arquitectura computacional de tipo RISC, del inglés *Reduced Instruction Set Computer* (Computadora con Conjunto de Instrucciones Reducido), le da las siguientes características fundamentales:

1. Instrucciones de tamaño fijo y presentadas en un reducido número de formatos.
2. Sólo las instrucciones de carga y almacenamiento acceden a la memoria por datos.

Además de esto el tener muchos registros de propósito general que hacen posible la segmentación y el paralelismo en la ejecución de las instrucciones, dan como resultado el reducir los accesos a memoria.

Los microcontroladores AVR tienen un canal ('pipeline' en inglés) con dos etapas (cargar y ejecutar), que les permite ejecutar la mayoría de las instrucciones en un ciclo de reloj, lo que los hace relativamente rápidos entre los microcontroladores de 8-bits. La idea de incluir un canal por el cual se pudieran dividir las instrucciones en pasos y trabajar en cada paso muchas instrucciones diferentes al mismo tiempo. Un procesador normal podría leer una instrucción, decodificarla, enviar a la memoria la instrucción de origen, realizar la operación y luego enviar los resultados.

La clave de la canalización es que el procesador pueda comenzar a leer la siguiente instrucción tan pronto como termine de leer la última instrucción, esto significa que ahora dos instrucciones se están trabajando (una está siendo leída, la otra está comenzando a ser decodificada), y en el siguiente ciclo habrán dos instrucciones que están siendo procesadas simultáneamente, tal como se puede apreciar en la Figura 1.2.

Mientras que una sola instrucción no se completaría más rápido, la *siguiente* instrucción sería completada enseguida. La ilusión era la de un sistema mucho más rápido.

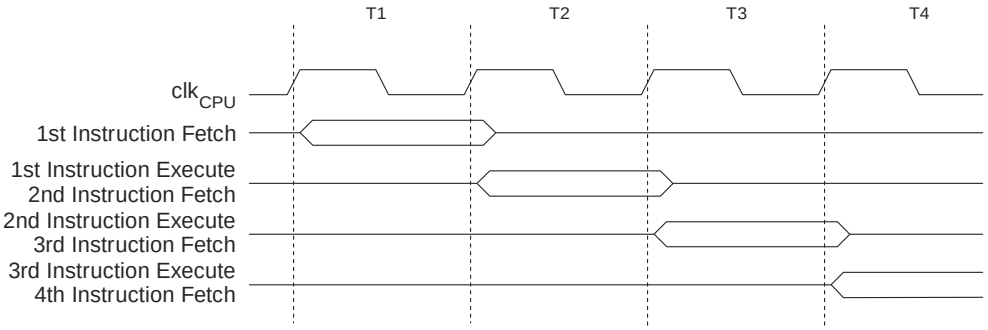


Figure 1.2. Instrucciones en paralelo carga y ejecuciór

Esto permite a los diseñadores una flexibilidad considerable, que permite, por ejemplo:

- Incrementar el tamaño del conjunto de registros
- Implementar medidas para aumentar el paralelismo interno
- Añadir grandes espacios de memoria
- Añadir otras funcionalidades, como E/S y relojes para minicontroladores

Las características que generalmente son encontradas en los diseños RISC son:

- Codificación uniforme de instrucciones (ejemplo: el código de operación se encuentra siempre en la misma posición de bit en cada instrucción, la cual es siempre una palabra), lo que permite una decodificación más rápida.
- Un conjunto de registros homogéneo, permitiendo que cualquier registro sea utilizado en cualquier contexto y así simplificar el diseño del compilador (aunque existen muchas formas de separar los ficheros de registro de entero y coma flotante).
- Modos de direccionamiento simple con modos más complejos reemplazados por secuencias de instrucciones aritméticas simples.

1.2.2 Los espacios de memoria

La arquitectura AVR tiene dos espacios principales de memoria, la Memoria de Datos y el espacio de Memoria de Programa. Además se destaca una Memoria EEPROM en algunos AVR para el almacenaje de datos que sean permanentes. Los tres espacios de memoria son lineales y regulares.

1.2.2.1 La memoria de Datos

En la Figura 1.3 se muestra cómo la memoria de datos se divide en tres regiones. La inferior, que ocupa las primeras 32 direcciones, está ocupada por el banco de registros de propósitos generales (direcciones $00-$1F). En este grupo de registros se encuentran tres registros con características diferentes a los demás, cada uno de ellos es de 16 bits, que empiezan desde el R26 hasta R31 que son conocidos como los registros X, Y, y Z.

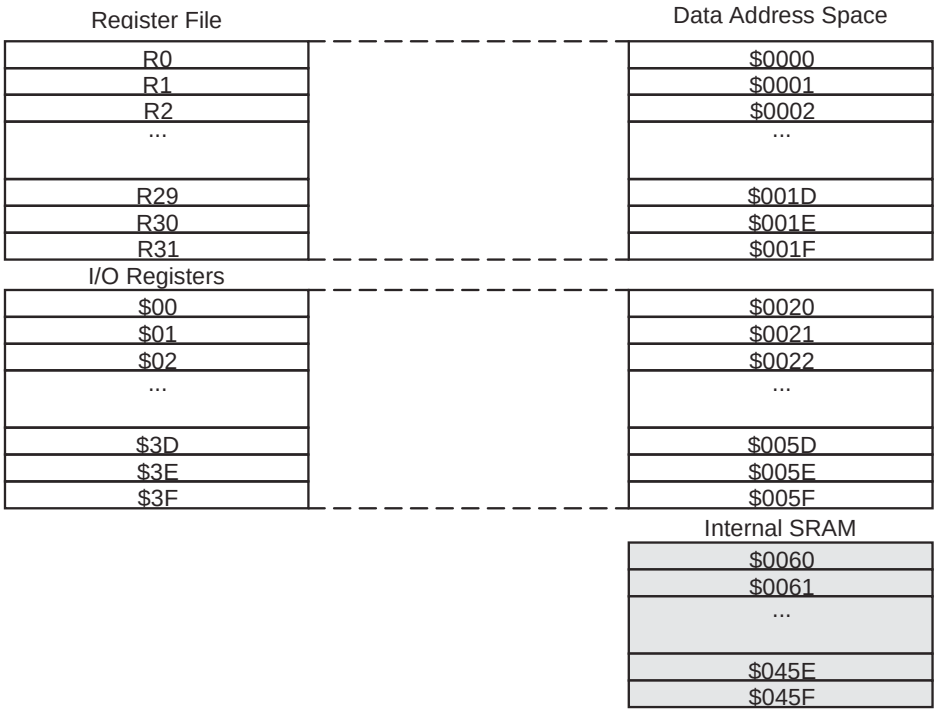


Figura 3. Mapa de la memoria de Datos

La región central, cubre **n** direcciones en donde se ubican los registros asociados a los diferentes puertos de entrada/salida del microcontrolador, UART, comparador analógico, PUERTO B, PUERTO D, etc. Además se encuentran los registros que permiten realizar las tareas de control de los distintos puertos de entrada/salida del microcontrolador, como también los registros de control de interrupciones internas y externas, controles para la utilización de osciladores internos o externos, etc.

La región más alta está asociada a la memoria SRAM, este espacio es utilizado en los modos de bajo consumo en los cuales el microcontrolador puede ser colocado. Aquí se guardará la información necesaria para lograr que el proceso de bajo consumo cambie a un modo normal de trabajo.

1.2.2.2 La memoria de Programa

Ya que las instrucciones para los AVR son de 16 o de 32 bits, en el espacio de la Memoria de Programa, que es una memoria de tipo Flash, se organiza como n kilobytes x 16 bits. Para la seguridad del software, en algunos AVR's, el espacio de Memoria de Programa es dividido en dos secciones, la sección de Programa de Boot y la sección de Programa de uso, como se muestra en la Figura 1.4.

La sección de Boot, está presente en algunos AVR's y es utilizada para proteger y dejar establecidos parámetros para el funcionamiento del AVR, tales como la frecuencia del oscilador, si este es interno o externo, permitir el acceso de instrucciones a la sección de Boot, seleccionar si un determinado pin va a ser utilizado como reset, cuál es su posición de inicio, etc.

La sección de programa de uso contiene las instrucciones que han de ser ejecutadas por el programa, todas estas instrucciones serán las que permitan controlar el funcionamiento del AVR junto con la utilización de los registros de

propósito general y los registros de entrada salida. En la figura 1.4 se muestra la arquitectura tipo Harvard, que poseen estos microcontroladores.

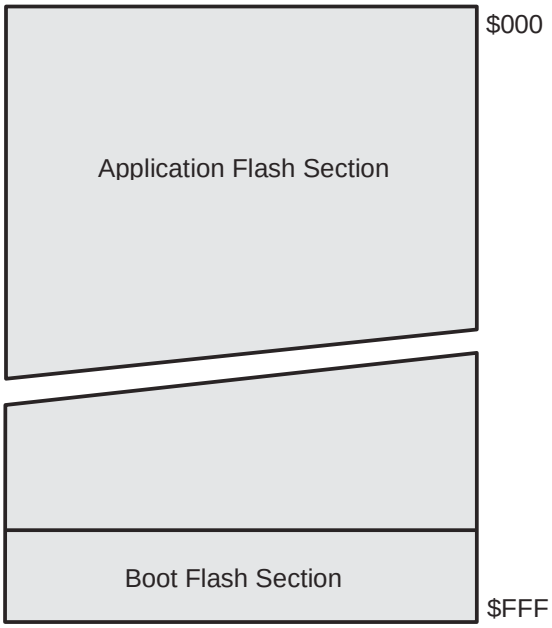


Figura 1.4 Mapa de la Memoria de Programa

La memoria de tipo Flash tiene, dependiendo del AVR, capacidad de entre 1000 a 10,000 ciclos de escritura y borrado, lo cual unido a una programación por sistema utilizando el bus SPI, hace que la carga y modificación de programas sea fácil, permitiendo el rápido desarrollo de aplicaciones.

1.2.2.3 La memoria EEPROM

En este espacio se ubica una memoria permanente del tipo ROM, que permite almacenar datos permanentemente en el microcontrolador aún cuando el suministro de energía no exista.

Para acceder a este espacio se tienen rutinas específicas que permiten trabajar con datos para los fines que se le dé por parte del programador, junto a una cantidad de registros que son utilizados por las instrucciones, hace posible un fácil

manejo de este espacio de memoria. Al igual que el resto de espacios de memoria son grabados o borrados por medio de la utilización del SPI, por lo tanto también tienen una cantidad de ciclos de escritura y borrado, como los demás espacios, que dependerán del microcontrolador y estos pueden ir desde 10,000 a 100,000 ciclos.

Los compiladores generan los archivos con una extensión específica, para notar que se trata de datos que serán almacenados en este espacio de memoria en particular, con lo cual aseguramos la integridad de estos datos en el microcontrolador.

1.2.3 Sistemas de reloj

Todo microcontrolador necesita una fuente de reloj para realizar su trabajo ya que el sistema de reloj es el corazón del mismo. Los AVR's tienen algunas opciones bastante interesantes que le dan ventajas que son necesarias de considerar.

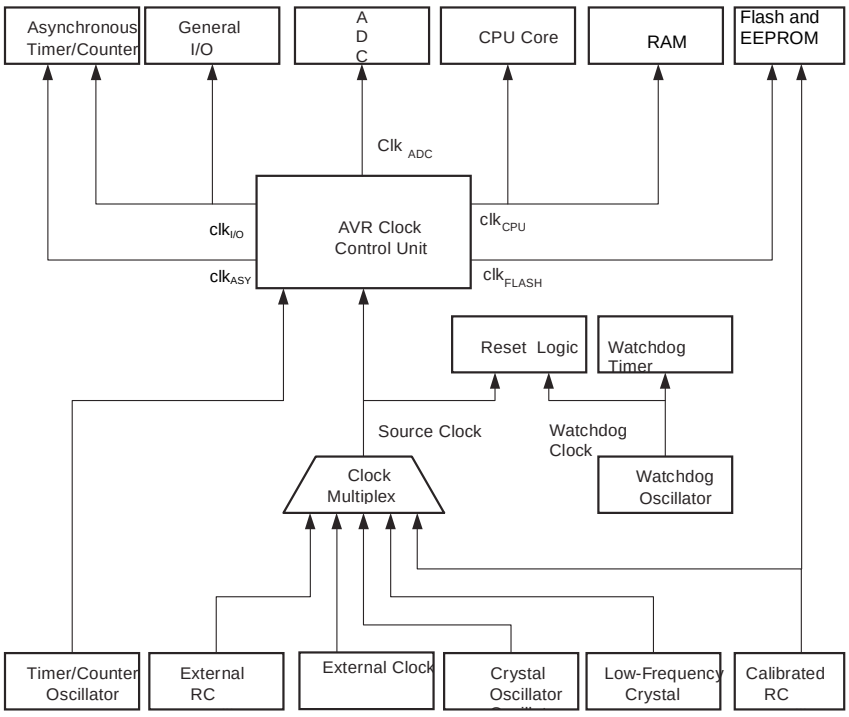


Figura 1.5 Opciones de Reloj

En la Figura 1.5 tenemos un esquema de cómo son administradas las fuentes que tiene el AVR para generar sus señales de reloj y cuáles son los usos dentro del microcontrolador. Por ejemplo, tenemos que el AVR puede generar una señal de reloj interna que dependiendo de los AVR variará desde 1Mhz, 4Mhz, 8Mhz y otros. La ventaja de este reloj es la posibilidad de contar con una fuente constante y segura de sincronización sin hardware adicional, útil en la mayoría de las aplicaciones.

Para cuando se tienen aplicaciones especiales con necesidades especiales de reloj, se tiene la posibilidad de incluir un oscilador externo con el valor que se desee, pero dentro de un rango especificado para cada AVR que siempre es mayor del que puede generar internamente. Aquí sí se necesita de un hardware adicional, que no es complicado, como se puede apreciar en la Figura 1.6.

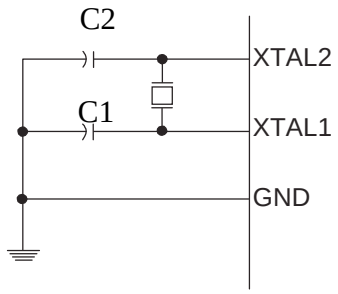


Figura 1.6 Conexiones del oscilador de cristal

La otra opción, mostrada en la Figura 1.7, es el configurar un oscilador externo por medio de una red RC mediante la cual se pueden obtener valores de oscilación parecidos a los que entrega el reloj interno y que para algún caso pueden ser útiles.

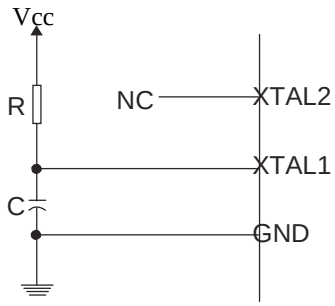


Figure 1.7 Configuración de una red RC externa

Otra alternativa es el tener una fuente externa, la cual estará en el orden de los valores que se tienen para un oscilador de cristal o una red RC, de la manera indicada en la Figura 1.8. Para todos estos casos se considera un error en la frecuencia de los relojes con una tolerancia entre el 1% y no mayor al 2%.

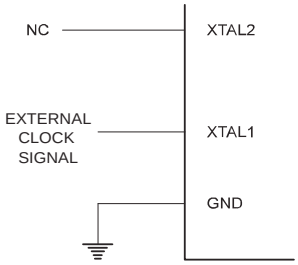


Figura 1.8. Configuración para un reloj externo

1.2.4 Interrupciones

Las interrupciones en los AVR permiten actuar de una manera clara al programador en el funcionamiento del programa, de tal forma que se pueden realizar tareas específicas. Todo AVR cuenta con algunas interrupciones básicas como: interrupciones externas, interrupciones debido a contadores internos, interrupciones generadas por una comunicación serial y como la más significativa de estas interrupciones tenemos el Reset, que no hace otra cosa que inicializar todo el sistema. En la Tabla 1.1 se muestran algunos ejemplos.

Program Address	Source	Interrupt Definition
0x000	RESET	External Pin, Power-on Reset, Reset
0x001	INT0	External Interrupt Request 0
0x002	INT1	External Interrupt Request 1
0x009	TIMER0 OVF	Timer/Counter0 Overflow
0x00B	USART, RXC	USART, Rx Complete
0x00C	USART, UDRE	USART Data Register Empty
0x00D	USART, TXC	USART, Tx Complete

Tabla 1.1. Interrupciones y Reset

Cada interrupción está asociada a un registro dentro de la memoria de Datos y dependiendo de la clase de interrupción se asociará a un registro o pin en el microcontrolador.

1.2.5 Puertos de entrada salida

Todos los puertos del AVR tienen la facilidad de ser puertos de lectura o escritura según sea necesario, además dentro de cada puerto los pines del mismo pueden ser configurados según sea el caso y esto es lo que hace que dentro de un mismo puerto puedan existir entradas como salidas. Para esto los AVR's tienen 3 registros por cada puerto y estos registros son:

- DDRx - Sirve para configurar los pines ya sea como entrada o salida, 0 es entrada, 1 es salida.
- PINx - Registro que sirve para entradas
- PORTx - Registro que sirve para salidas

Los pines de un puerto cualquiera, pueden tener más de una función dentro de la configuración de cada AVR, lo que dependerá de la aplicación que se le quiera dar.

Por ejemplo, en el Atmega8 se tiene como se muestra en la Tabla 1.2:

Pines del puerto	**Funciones alternativas**
PB7	XTAL2 (Chip Clock Oscillator pin 2) TOSC2 (Timer Oscillator pin 1)
PB6	XTAL1 (Chip Clock Oscillator pin 1 or External clock input) TOSC1 (Timer Oscillator pin 1)
PB5	SCK (SPI Bus Master clock Input)
PB4	MISO (SPI Bus Master Input/Slave Output)

PB3	MOSI (SPI Bus Master Output/Slave Input) OC2 (Timer/Counter2 Output Compare Match Output)
PB2	SS (SPI Bus Master Slave select) OCIB (Timer/Counter1 Output Compare Match B Output)
PB1	OC1A (Timer/Counter1 Output Compare Match A Output)
PB0	ICP (Timer/Counter1 Input Capture Input)

Tabla 1.2. Puerto B Funciones Alternativas

Aparte de las funciones dadas por el fabricante, que son descritas en las especificaciones técnicas de cada AVR en particular, cada usuario de los AVR's ocupará los pines como guste y esto hace que sea posible una gran versatilidad en las aplicaciones, debido a que se puede acceder a los distintos registros ya especificados, sea en la lectura o la escritura.

1.2.6 Temporizadores/Contadores

Los temporizadores y contadores son capaces de contar en forma ascendente solamente. Los temporizadores están equipados con un prescaler[3] programable, que permite seleccionar para el temporizador de entre las frecuencias disponibles en el reloj del procesador. Si se selecciona un pin del puerto como fuente externa, son capaces de contar los eventos externos. Pueden ser inicializados o detenidos en cualquier momento colocando la fuente de reloj a cero. El contenido de los temporizadores/contadores puede ser leído y modificado en cualquier momento.

Los temporizadores/contadores son usados al desbordarse el registro de cuenta, lo cual activa una bandera que puede ser tomada en cuenta para alguna acción específica o al unirla con la interrupción del temporizador para que realice una rutina específica de instrucciones. En la Figura 1.9 se puede apreciar el conjunto de

[3] Divisor de frecuencia, que permite disminuir la frecuencia del reloj de CPU[4].

registros dentro del AVR que hacen posible el funcionamiento del temporizador/contador de 8 bits.

Los temporizadores/contadores de 16 bits además proporcionan la oportunidad de comparar su contenido con uno o dos valores programados y fijar automáticamente el valor de una bandera. Algunos temporizadores/contadores pueden ser programados para producir una modulación PWM utilizando esta característica.

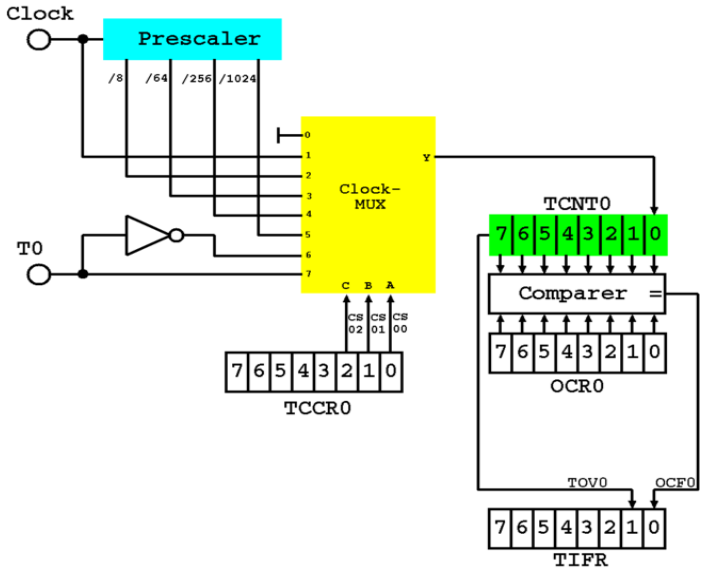


Figura 1.9. Estructura interna de un temporizador/contador

1.2.7 Comunicación serial

Existen algunos modos en los cuales se puede realizar la comunicación serial. Uno de los más conocidos es el USART (Universal Synchronous Asynchronous Reciever Transmitter), que generalmente se debe conectar a un convertidor de nivel, para acoplarlos a los estándares RS-232, RS-485, y CANBUS, que para el AVR son transparentes pues ellos sirven para que se acoplen a las características de red de otros dispositivos a los que se conecte.

El más conocido es el MAX-232 que nos sirve para hacer una comunicación con el puerto serial de la computadora, como se puede apreciar en la Figura 1.10. Esta interface hace posible la comunicación de la mayoría de microcontroladores con la computadora y con dispositivos que utilizan este protocolo de comunicaciones para periféricos.

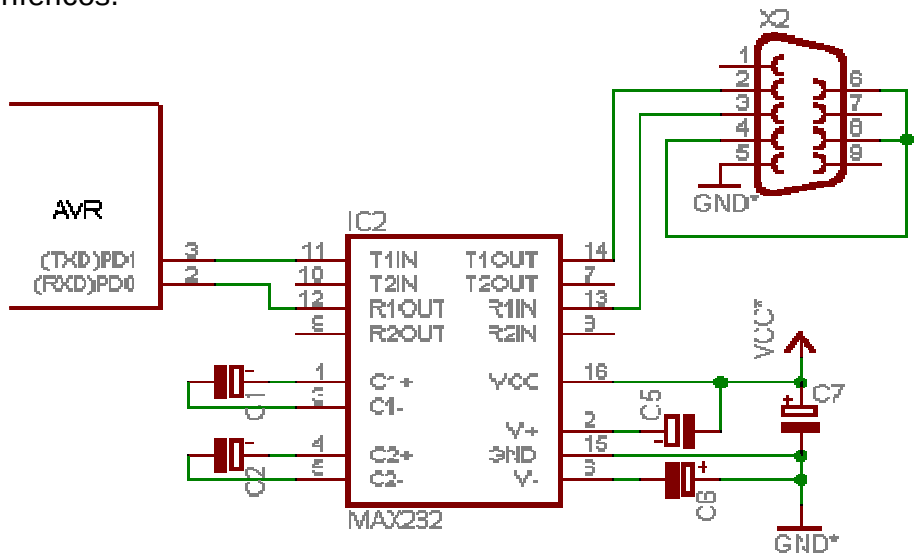


Figura 1.10. Conexión del AVR al puerto del computador por medio de un MAX232

RS-232 envía voltajes entre -12v a 12v, que están fuera de la capacidad eléctrica del AVR, y los formatos que puede aceptar van desde recibir tramas con 5,6,7,8 o 9 bits de datos con uno o dos bits de parada y el característico bit de inicio. La comunicación puede ser Full Duplex ya que cuenta con registros independientes para la transmisión como para la recepción, a velocidades estándar, 1200, 2400, 9600, etc. Este modo de comunicación hace posible la interacción del microcontrolador con otros dispositivos que le permiten ampliar su campo de acción en cuanto a las aplicaciones que pueden tener.

1.3 EL RECEPTOR GPS [6][7][8][9]

El Global Positioning System (GPS) o Sistema de Posicionamiento Global (más conocido por sus siglas GPS; NAVSTAR GPS es su nombre correcto) es un

Sistema Global de Navegación por Satélite (GNSS) que permite determinar en todo el mundo la posición de un objeto, una persona, un vehículo o una nave, con una precisión que va desde los centímetros hasta unos pocos metros. El sistema fue desarrollado e instalado, y actualmente es operado, por el Departamento de Defensa de los Estados Unidos.

El GPS funciona mediante una red de 24 satélites (21 operativos y 3 de respaldo) en órbita alrededor del globo a 20,200 km con trayectorias sincronizadas para cubrir toda la superficie de la tierra. Cuando se desea determinar la posición, el dispositivo que desea saber su posición localiza automáticamente como mínimo cuatro satélites de la red, de los que recibe unas señales indicando la posición y el reloj de cada uno de ellos. En base a estas señales, el aparato sincroniza el reloj del GPS y calcula el retraso de las señales, es decir, la distancia al satélite por "triangulación".

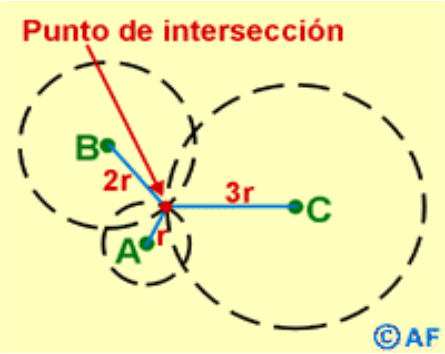


Figura 1.11. Triangulación en 3D

La triangulación en el caso del GPS, se puede apreciar en la Figura 1.11, a diferencia del caso 2-D que consiste en averiguar el ángulo respecto de puntos conocidos, se basa en determinar la distancia de cada satélite respecto al punto de medición. Conocidas las distancias, se determina fácilmente la propia posición relativa respecto a los tres satélites. Conociendo además las coordenadas o posición de cada uno de ellos por la señal que emiten, se obtienen la posición absoluta o las coordenadas reales del punto de medición. También se consigue una exactitud

extrema en el reloj del GPS, similar a la de los relojes atómicos que desde tierra sincronizan a los satélites.

La posición se calcula a partir de la medición de la distancia de tres satélites. Pero como la posición de los satélites no es conocida, lo que se hace es medir el tiempo que tarda en llegar la señal emitida por el satélite hasta nuestro receptor de GPS.

Matemáticamente se trata de medir una distancia a partir de una señal de radio que viaja a la velocidad de la luz 300,000 Km por segundo. Para esto necesitamos saber la velocidad, que es la de la luz y el otro parámetro es el tiempo de viaje de la señal que es extremadamente corto. Esto hace que necesitemos relojes sumamente precisos tanto en el receptor GPS como en el satélite y que además estén sincronizados. Los dos equipos, tanto satélite como receptor emiten sus señales al mismo tiempo y de esta forma cuando una señal llega a un receptor, se sabe cuál es el tiempo de retraso entre las señales, lo que daría el parámetro del tiempo que se necesita para calcular la distancia al satélite.

Ahora consideremos que los relojes de los receptores GPS no son tan precisos como los que tienen los satélites. Lo que se hace para compensar esta imprecisión, es realizar una cuarta medición. El propósito de esta cuarta medición será obtener una medida que permitirá tomar una decisión eliminando cualquier discrepancia que se dé con respecto a la hora universal, ésta afectará a las cuatro mediciones y el receptor buscará un factor de corrección único, que siendo aplicado a sus mediciones de tiempo hará que los rangos coincidan en un solo punto.

Dicha corrección permitirá al reloj del receptor ajustarse nuevamente a la hora universal y de esa manera tenemos un reloj atómico que hará que las demás mediciones del receptor GPS den un posicionamiento preciso. Ya con esta medida de tiempo asegurada, tenemos todo lo necesario para medir nuestra distancia a un satélite en el espacio.

Pero, para que la triangulación funcione necesitamos conocer no sólo la distancia sino que debemos conocer dónde están los satélites con toda exactitud ya que éstos se convertirán en nuestros puntos de referencia. La altura de 20,000 Km. brinda un gran beneficio ya que el satélite orbitará de manera regular y predecible mediante ecuaciones matemáticas sencillas. En tierra, todos los receptores GPS tienen un almanaque programado que les informa dónde está cada satélite en el espacio, en cada momento, lo que permite conseguir el punto deseado.

Una vez que se ha ubicado la posición en la superficie terrestre necesitamos alguna forma de representar los datos para que puedan ser utilizados por las diversas aplicaciones que utilizan el GPS.

1.3.1 EL NMEA [10]

La Asociación Electrónica Marítima Nacional (NMEA) ha desarrollado un estándar que permite a los equipos electrónicos marítimos enviar la información a computadores y a otros equipos marítimos. El receptor GPS está definido dentro de este estándar. La mayor parte de programas de computación que proporcionan información en tiempo real esperan que los datos estén en el formato del estándar NMEA. Estos datos incluyen el PVT (posición, velocidad, tiempo) que es una solución calculada por el receptor GPS.

La idea del NMEA es enviar una línea de datos llamada sentencia, que es totalmente independiente de otras sentencias. Hay sentencias estándar por cada tipo de dispositivo así como hay también la capacidad de definir sentencias propietarias para el empleo específico. Todas las sentencias estándar tienen dos letras como prefijo, que definen el dispositivo que usa aquella sentencia. (Para receptores GPS el prefijo es GP) que son seguidas por tres letras que definen el contenido de la sentencia. Además NMEA permite a los fabricantes de hardware definir sus propias sentencias propietarias para cualquier objetivo que ellos deseen. Todas las sentencias propietarias comienzan con la letra P y son seguidos por 3 letras que

identifican al fabricante que es propietaria de esa sentencia. Por ejemplo una sentencia de Garmin comenzaría con PGRM y Magellan comenzaría con PMGN.

Cada sentencia comienza con un ‘$' y termina con un retorno de carro y el inicio de una nueva línea, que son caracteres no imprimibles. La cantidad de datos contenida no puede ser mayor a 80 caracteres de texto visible, más el retorno de carro. Los datos contenidos dentro de esta línea están separados por comas. Los datos recibidos son texto ASCII y pueden ser parte de varias sentencias de longitud variable. Los datos pueden variar en su cantidad de acuerdo a la precisión contenida en el mensaje. Por ejemplo el tiempo podría ser indicado con dos decimales en cuanto a los segundos o la posición puede ser mostrada con 3 o 4 dígitos después del punto decimal. Los programas que leen los datos usan las comas para determinar los límites de los campos de la sentencia. Al final de cada sentencia existe un campo de checksum que es precedido por '*' que contiene dos dígitos hexadecimales que representan un OR exclusivo entre todos los caracteres sin incluir el ‘$' y ''*''. El checksum es requerido en algunas sentencias.

1.3.1.1 Sentencias NMEA

El estándar NMEA consiste en sentencias, la primera palabra, la que está entre ‘$' y la primera coma, define la interpretación del resto de la sentencia. Cada tipo de datos tendrá su propia interpretación y está definida en el estándar. El estándar NMEA no define órdenes para indicar al GPS algún cambio en su funcionamiento, en cambio cada receptor envía todos los datos y espera que la mayor parte de ellos sean utilizados. Algunos receptores tienen la posibilidad de seleccionar un subconjunto de todas las sentencias o, en algunos casos, hasta sentencias individuales para enviar. No hay ningún modo de pedir retransmisión de las sentencias, cuando no son leídas correctamente, o pedir una de las sentencias ya leídas. Con el campo de checksum se realiza la verificación, si es incorrecta se espera a las siguientes sentencias para tener datos válidos.

De las muchas sentencias definidas en el estándar podemos considerar algunas sentencias interesantes, por ejemplo:

- GGA - Fix information.
- GSA - Overall Satellite data.
- GSV - Detailed Satellite data.
- RMB - Recommended navigation data for GPS.
- RMC - Recommended minimum data for GPS.
- VTG - Vector track an Speed over the Ground.
- ZDA - Date and Time.

De las sentencias indicadas analizaremos la sentencia RMC con un ejemplo de una trama recibida.

RMC - NMEA tiene su propia versión de GPS esencial con los datos PVT (posición, velocidad, tiempo). RMC, el Mínimo Recomendado, sería similar a esta sentencia:

$GPRMC, 123519, A, 4807.038, N, 01131.000, E, 022.4,084.4,230394,003.1, W*6A

Donde:

RMC	Sentencia Mínima Recomendada RMC
123519,	Fijan la hora, tomada como 12:35:19 UTC
A,	Estado A = activo o V = Void.
4807.038, N	Latitud 48 deg 07.038 ' N
01131.000, E	Longitud 11 deg 31.000 ' E
022.4,	Velocidad sobre la tierra en nudos
084.4,	Pista del ángulo en grados
230394,	Fecha 23 de marzo de 1994
003.1,	Variación Magnética
*6A	Los datos de checksum, siempre comienza con *

De esta manera cada una de las sentencias NMEA contienen información que será utilizada según sea el caso y toda esta información se encuentra presente en el estándar.

1.4 GENERACIÓN DE IMÁGENES EN LA PANTALLA DE TELEVISIÓN [11][12][13]

La televisión (TV) es un sistema de telecomunicación para la transmisión y recepción de imágenes (en movimiento) y sonido a distancia. Esta transmisión puede ser efectuada mediante ondas de radio o por redes especializadas de televisión por cable.

La palabra "televisión" es un híbrido de la voz griega "Tele" (distancia) y la latina "visio" (visión). El término televisión se refiere a todos los aspectos de transmisión y programación de audio y video. A veces se abrevia como TV. Televisor es el terminal receptor de la señal, que usualmente consta de una pantalla y controles; esta idea se muestra en la Figura 1.12.

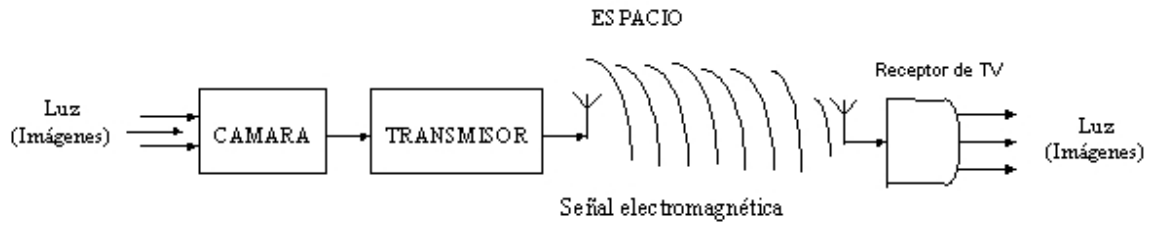


Figura 1.12. Esquema general del sistema de televisión

La televisión es uno de los aparatos de uso diario en la vida del hombre, que es el medio de comunicación más difundido del mundo, influenciando en los hábitos culturales y de consumo. Su funcionamiento se basa en el fenómeno de la fotoelectricidad, que es el responsable de la transformación de la luz en corriente eléctrica. Las imágenes que capta una cámara se emiten por ondas de alta

frecuencia hasta las antenas de recepción y se reproducen en nuestros hogares, a través del tubo de imagen del televisor.

El ojo humano puede ser engañado fácilmente, ya que ante una sucesión rápida de imágenes tenemos la percepción de un movimiento continuo. Por ejemplo una cámara de cine no es otra cosa que una cámara de fotos que entrega fotos muy rápido, de tal forma que en un cine las 24 imágenes, o fotogramas, que se usan por segundo, dan la impresión de movimiento continúo. Esto es conocido como un formato **progresivo**, eso quiere decir que se pasa de una imagen a otra rápidamente de tal forma que vemos una imagen completa y, casi de inmediato, vemos la siguiente. Si tenemos en cuenta que vemos 24 imágenes por segundo, cada imagen se reproduce durante 0.04167 segundos. Por tanto las diferencias entre una imagen y otra son mínimas ya que de otra manera se notaría que son imágenes separadas y no un video continuo.

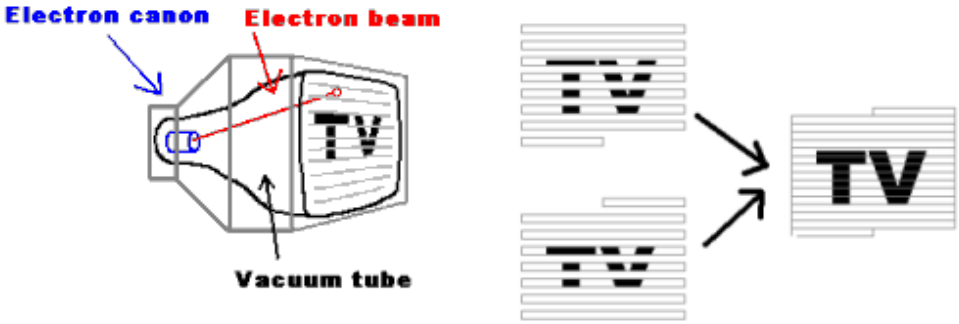
Por otra parte una pantalla de un televisor no funciona como un proyector de cine, se construye de un tubo de vacío, que contiene una pantalla de fósforo en la cual un cañón de electrones que la apunta hace chocar electrones en la pantalla, la luz emitida por el fósforo mientras chocan los electrones genera una posluminiscencia corta. Inicia su barrido por la primera línea hasta que regresa a ella. Un televisor está dividido en líneas horizontales, y dependiendo del número de ellas se tienen los formatos de televisión. Estas líneas no se muestran todas a la vez un mismo fotograma, sino que la imagen comienza a aparecer desde las líneas superiores y sucesivamente se va llenando el resto de la pantalla hasta llegar a las líneas inferiores. Un único fotograma no es mostrado de una sola vez, sino de modo secuencial. Al igual que pasaba con el cine, este proceso de actualización de líneas es tan rápido que, en principio, nuestro ojo lo percibe todo como un continuo.

Sin embargo, este proceso presentaba un problema debido a que los tubos de imagen de los primeros televisores hacían que cuando la imagen actual llegaba a las

últimas líneas la imagen de las líneas superiores comenzaba a desvanecerse. Fue entonces cuando surgió la idea de los **campos** y del **video entrelazado**.

En la actualidad no se transmite la imagen completa de una sola vez, sino primero una mitad y luego la otra mitad. Ambas mitades se complementan reproduciendo la escena. A cada grupo de líneas, par o impar, se le llama campo. En este caso cada escena o imagen completa requiere dos campos, el primer campo formado por las líneas pares y el otro campo formado por las líneas impares.

Así tendríamos el campo A o superior (Upper o Top en inglés) formado por las líneas pares (Even en inglés) y el campo B, inferior o secundario (Lower o Bottom en inglés) formado por las líneas impares (Odd en inglés). Esta forma de crear el video es conocida como video entrelazado, en la Figura 1.13 se muestra una grafica que permite observar la idea del video entrelazado y la formación de la misma por medio del haz de electrones en el tubo de rayos catódicos.



El haz electrónico que dibuja la pantalla

Las imágenes bipartitas se convierten en una imagen entera.

Figura 1.13. Haz electrónico y campos de imagen

El video entrelazado se obtiene efectuando primero un barrido de líneas pares (de izquierda a derecha), regresa de nuevo a la parte superior de la pantalla e inicia entonces todo el barrido de las líneas impares, como se muestra en la Figura 1.14. El motivo fundamental de usar esta técnica es debido a la aparición de un efecto secundario consistente en un centelleo de la imagen, el cual resulta desagradable.

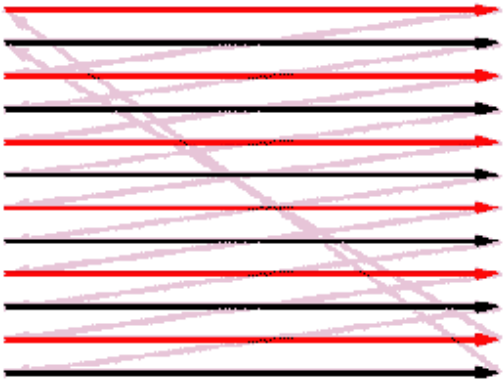


Figura 1.14. Video entrelazado

El presentar imágenes en forma continua tiene asociado un corte de luz entre cada una de las imágenes, lo que origina que el ojo perciba el desagradable efecto denominado **parpadeo**; esta sensación de destello o centelleo de las imágenes presentadas que se observan como una constante variación de la luminosidad depende de la velocidad con que se realizan los cortes de luz. Este efecto se elimina con un aumento en la velocidad en que se presentan las imágenes o realizando la composición de las imágenes a través del video entrelazado.

En televisión se denomina **cuadro** a la reproducción completa en la pantalla de una escena, es decir la imagen que resulta de la composición de todos sus campos en los que está dividida, cada vez que cambia el cuadro, debe existir una señal eléctrica que controle el cambio, es decir que haga pasar el punto desde la parte inferior de la pantalla a la parte superior para que se inicie un nuevo cuadro.

Dependiendo de la cantidad de imágenes que se presentan por segundo y el número de líneas por imagen se tienen los distintos formatos de televisión. Hay dos formatos diferentes: uno es el formato PAL, usado en Europa, otro el NTSC usado en América y Japón como zonas más destacadas, como último tenemos el SECAM. En los formatos PAL y SECAM la velocidad de imágenes por segundo es de 25 y de 29.97 en el formato NTSC. A esta velocidad de imágenes por segundo se le llama Cuadros Por Segundo en español (CPS), o Frames Per Second en inglés (FPS)

1.4.1 PAL [14] [15]

PAL es la sigla de Phase Alternating Line (línea alternada en fase). Este es el nombre con que se designa al sistema de codificación empleado en la transmisión de señales de televisión en color en la mayor parte del mundo. De origen alemán, se utiliza en la mayoría de los países africanos, asiáticos y europeos, además de Australia.

El sistema PAL se deriva directamente del NTSC con algunas correcciones técnicas y surgió en el año 1963, de manos del Dr. Walter Bruch en los laboratorios de Telefunken en su intento por mejorar la calidad y reducir los defectos en los tonos de color que presentaba el sistema NTSC. No obstante, los conceptos fundamentales de la transmisión de señales han sido adoptados del sistema NTSC.

El sistema PAL es más robusto que el sistema NTSC. Este último puede ser técnicamente superior en aquellos casos en los que la señal es transmitida sin variaciones de fase (por tanto, sin los defectos de tono de color). Pero para eso deberían darse unas condiciones de transmisión ideales (sin obstáculos como montes, estructuras metálicas...) entre el emisor y el receptor. En cualquier caso en el que haya *rebotes* de señal, el sistema PAL se ha demostrado netamente superior al NTSC. Esa fue una razón por la cual la mayoría de los países europeos eligieron el sistema PAL, ya que la orografía europea es mucho más compleja que la norteamericana (todo el medio oeste es prácticamente llano). En el único aspecto en el que el NTSC es superior al PAL es en evitar la sensación de parpadeo que se puede apreciar en la zona de visión periférica cuando se mira la TV en una pantalla grande (más de 21 pulgadas), porque la velocidad de refresco es superior (30 Hz en NTSC frente a 25 Hz en PAL).

Las líneas en las que la fase está invertida con respecto a cómo se transmitirían en NTSC se llaman a menudo líneas PAL, y las líneas que coincidirían en fase a cómo se transmitirían en NTSC se denominan precisamente líneas NTSC.

1.4.2 SECAM [15][16]

Son las siglas de *Séquentiel Couleur avec Mémoire* en francés o "Color secuencial con memoria". Es un sistema para la codificación de televisión en color analógica utilizado por primera vez en Francia. Este sistema fue inventado por un equipo liderado por Henri de France trabajando para la firma Thomson. Es históricamente la primera norma de televisión en color europea.

Igual que los demás sistemas utilizados para la transmisión de televisión en color en el mundo el SECAM es una norma compatible, lo que significa que los televisores monocromos (B/N) preexistentes a su introducción son aptos para visualizar correctamente los programas codificados en SECAM, aunque naturalmente en blanco y negro.

SECAM utiliza el mismo ancho de banda que PAL, pero transmite la información de color secuencialmente. SECAM funciona a 625 líneas por imagen.

1.4.3 NTSC [17][18]

NTSC es el sistema de codificación que se usa en los países de América Latina, excepto Argentina, de tal forma que se hará un análisis más detallado de este sistema que de los anteriormente mencionados debido a que es parte de este proyecto de titulación, haciendo énfasis en la generación de las señales de video.

NTSC es un sistema de codificación y transmisión de televisión analógica desarrollada en Estados Unidos alrededor de 1940, y que se emplea actualmente en la mayor parte de América y Japón, entre otros países. Las siglas del comité de expertos que desarrolló el sistema, la *National Television System(s) Comité,* da el nombre con que comercialmente se conoce.

Un derivado de NTSC es el sistema PAL como se ha dicho antes. El problema de insertar el color en la señal de televisión sin pérdida de compatibilidad con la televisión en blanco y negro, sin aumentar notablemente su ancho de banda, se solucionó utilizando el concepto de modulación de amplitud en cuadratura.

1.4.3.1 Video NTSC

El formato NTSC consiste en la transmisión de 29.97 cuadros de vídeo en modo entrelazado con un total de 525 líneas de resolución y una velocidad de actualización de 30 cuadros de video por segundo y 60 campos de alternación de líneas.

Para garantizar la compatibilidad con el sistema NTSC en blanco y negro, el sistema NTSC de color mantiene la señal monocromática en blanco y negro como componente de luminancia de la imagen en color, mientras que las dos componentes de crominancia se modulan con una modulación de amplitud en cuadratura sobre una subportadora de 3.579545 MHz. La demodulación de las componentes de crominancia es necesariamente síncrona, por lo tanto se envía al inicio de cada línea una señal sinusoidal de referencia de fase conocida como "salva de color", "burst" o "colorburst". Esta señal tiene una fase de 180º y es utilizada por el demodulador de la crominancia para realizar correctamente la demodulación. A veces, el nivel del "burst" es utilizado como referencia para corregir variaciones de amplitud de la crominancia.

1.4.3.2 Radiodifusión

Un canal de televisión transmitido en el sistema NTSC utiliza alrededor de 6 Mhz de ancho de banda, para contener la señal de video, más una banda de resguardo de 250 khz entre la señal de video y la de audio. Los 6 Mhz de ancho de banda se distribuyen de la siguiente forma: 1.25Mhz para la portadora de vídeo principal con dos bandas laterales de 4.2Mhz; las componentes de color a 3.579545

Mhz sobre la portadora de video principal, moduladas en cuadratura; la portadora de audio principal de 4.5 Mhz transmitida sobre la señal de video principal y los últimos 250 Khz de cada canal para la señal audio estereofónica en frecuencia modulada. La señal de crominancia en la norma NTSC se transmite en una frecuencia subportadora FM en los 3.58 Mhz

1.4.3.3 Inconvenientes

Los problemas de transmisión e interferencia tienden a degradar la calidad de la imagen en el sistema NTSC, alterando la fase de la señal del color, por lo que en algunas ocasiones el cuadro pierde su equilibrio en el color al momento de ser recibido, esto hace necesario incluir un control de tinte, que no es necesario en los sistemas PAL o SECAM. Por eso en broma se le denomina "NTSC: Never The Same Color" ("NTSC: Nunca el mismo color"). Otra de sus desventajas es su limitada resolución, de solo 525 líneas de resolución vertical, la más baja entre todos los sistemas de televisión, lo que da lugar a una imagen de calidad inferior a la que es posible enviar en el mismo ancho de banda con otros sistemas. Además, la conversión de los formatos cinematográficos a NTSC requiere un proceso adicional conocido como " pulldown de 3:2".

1.4.3.4 La información en la señal video [19]

La imagen considerada en la pantalla tiene diversas intensidades en cuanto al color que se puede apreciar, como los barridos del haz electrónico sobre la pantalla se hacen línea a línea, la intensidad que debe estar en determinada posición de la línea se envía como nivel de voltaje en la señal video. No hay información en alguna señal apreciable para la posición del haz en la pantalla. Para solucionar esto, se tiene un pulso de la sincronización que se envía al principio de cada línea para indicarle a la televisión que la línea actual se ha terminado y bajar el haz a la línea siguiente, como si fuera un retorno de carro. La televisión debe también saber

cuándo se terminó un campo y está empezando otro, esto se hace con un patrón especial en la sincronización.

Una imagen contiene 30 cuadros actualizados cada segundo, lo que nos da 60 campos por segundo, lo que hace necesario tener una señal video con una gama de voltaje de 0 a 1V, donde 0.3V representa negro, y 1.0V es blanco (las intensidades grises tienen voltajes entre estos valores). Los niveles cerca de cero representan pulsos de la sincronización.

1.4.3.5 Explorando una línea [20][21]

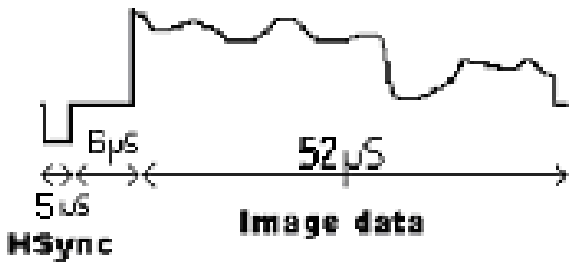


Figura 1.15. Línea que compone un campo

En la Figura 1.15 se tiene una idea de lo que es una línea dentro de la imagen pues toda ella se divide en líneas, las cuales son la parte más importante de la imagen puesto que contienen la información a ser mostrada. Cada línea tiene una duración de 63 µs y está compuesta de algunas partes que le permiten a la televisión saber qué acción realizar.

- Primero un pulso de sincronismo de 4.7 µs que es enviado al inicio de la línea para indicar que se inicia una nueva línea, fijando la señal a 0V.

- Posteriormente se tienen 5.9 µs después del pulso de sincronismo para conseguir que el haz llegue a la posición deseada para iniciar la escritura

de la imagen en esa línea. Durante este tiempo la señal tiene un valor de 0.3V que es el nivel negro.

- A continuación se tiene un período para la imagen de 51.5 µs, que inicia en la izquierda de la pantalla y que va hasta la derecha con las intensidades de voltaje obtenidas de la señal de video.

- Por último se tiene 1.4 µs en los cuales se coloca la señal en 0.3V para que luego de esto se produzca el pulso de sincronismo horizontal.

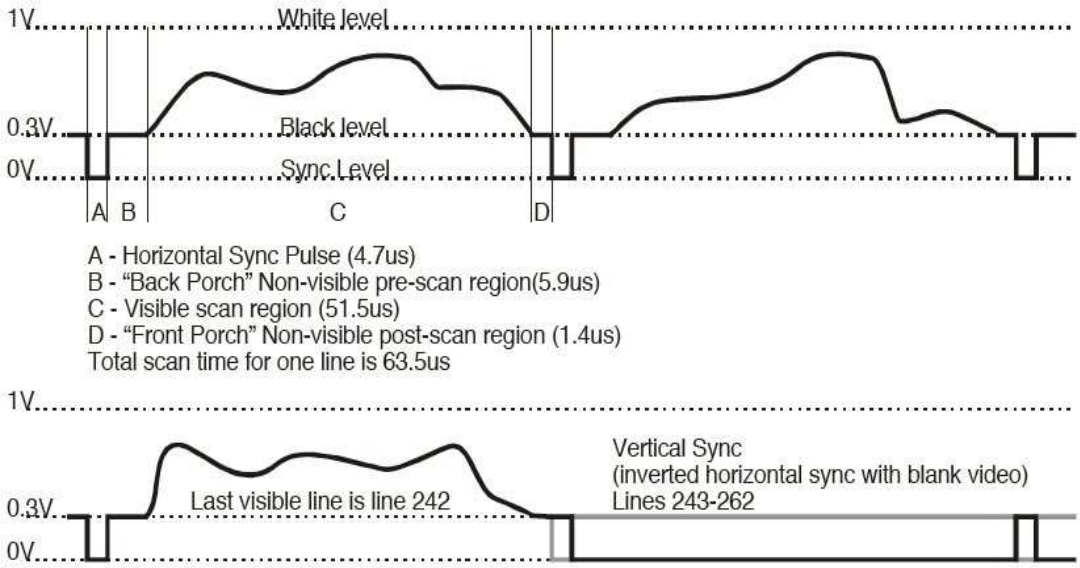


Figura 1.16. Líneas de sincronismo vertical y horizontal

En la Figura 1.16 se muestra cómo se dividen los tiempos de exploración de una línea y los pulsos de sincronismo horizontal que son necesarios para una imagen. La imagen se construye a partir de 525 líneas, pero una TV no muestra 525 líneas, algunas de las líneas se utilizan para los pulsos de sincronización. Los pulsos de sincronismo vertical en el caso del NTSC son 20 por campo mostrado, es decir que en el instante que ya están dibujadas 242 líneas, se tienen 20 líneas que son utilizadas por la televisión para colocarse en la posición necesaria para dibujar el nuevo campo, un patrón de pulsos especial es enviado.

En una línea de sincronismo horizontal se tiene que el pulso que indicaba que es un pulso de sincronismo vertical, cambia de un valor de 0V a un valor de 0.3v y en el espacio que debería contener la información de la imagen se tiene un valor de 0V.

De esta forma se tiene una señal que genera la imagen que mostramos en un televisor blanco y negro, de tal manera que el tiempo da la posición de un punto y el valor en el voltaje da la intensidad de la luminosidad que va desde el negro que está asociado a 0.3V hasta el valor de 1V que da el valor del color blanco, los valores 0.3 y 1V representan el color negro y blanco respectivamente, en la escala de grises.

1.5 GENERACIÓN DIGITAL DE CARACTERES PARA TELEVISIÓN

1.5.1 La generación de caracteres [22]

Como se consideró anteriormente al mirar cómo se producen las imágenes en una televisión, se tiene una idea de cómo generar caracteres, considerando que se puede controlar son la intensidad del rayo incidente en la pantalla de televisión, que provocará una luminosidad dando como resultado un color determinado. Con respecto a la ubicación de un punto determinado en la pantalla, solo se puede considerar a través del tiempo, ya que al ser la exploración de las líneas algo constante se puede predecir su posición.

Con estos criterios podemos entender que la generación de los caracteres se puede realizar por medio de un barrido por filas de una matriz de caracteres fabricada de modo que se muestren los caracteres deseados. Por lo general se utilizan matrices de 35 puntos con 5 puntos de ancho y 7 puntos de alto o una matriz de 63 puntos con 7 puntos de ancho y 9 de alto.

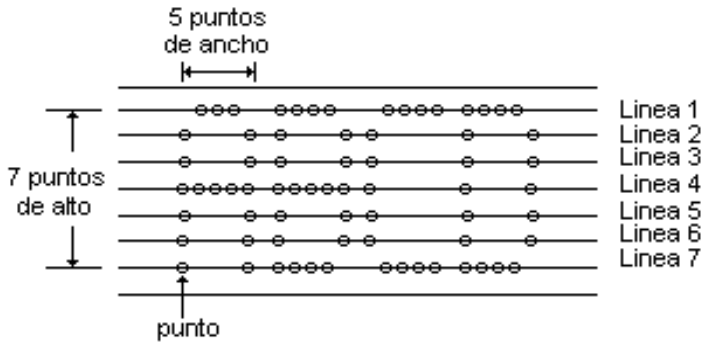


Figura 1.17. Generación de caracteres en la pantalla

Independientemente de la cantidad de puntos de la matriz para formar el carácter, el principio de la generación de los mismos es como se mira en la Figura 1.17. A medida que se realiza la exploración de una línea en la pantalla se va controlando la luminosidad de acuerdo a la matriz de puntos, todos los caracteres a ser mostrados deben ser colocados en la línea, teniendo así un barrido por filas. Cuando se haga la exploración de la segunda línea se repite el proceso y así se tiene una segunda línea que ha dibujado los caracteres de acuerdo a la matriz. Cuando se completa la exploración de las 7 líneas en un caso o 9 líneas en otros, se han dibujado caracteres sobre un campo. Inmediatamente hay que repetir este proceso durante los 60 campos que en un segundo forman una imagen que es presentada en la pantalla de televisión.

Si se reúnen las condiciones antes mencionadas se puede realizar la generación de caracteres en cualquier pantalla de televisión. El problema fundamental radica en conseguir circuitos que trabajen a la velocidad necesaria, pues se tiene 51.5 μs para realizar lo que se indica en la figura anterior para poder apreciar una imagen continua en el tiempo.

1.5.2 Aplicaciones comunes

La generación digital de caracteres para televisión es algo que se ha hecho desde hace algunos años, teniendo como aplicaciones prácticas: el teletexto, el sistema de Close Caption, la subtitulación en la programación.

1.5.2.1 El teletexto [23]

El teletexto es un servicio de información en forma de texto que se emite junto con la señal de televisión. Necesita televisores adecuados para acceder al servicio; la mayor parte de televisores que se venden desde el año 1990 incorporan este sistema. Básicamente, consiste en un conjunto (más o menos extenso) de páginas, las cuáles se eligen a través del control a distancia. Se escribe el número de la página (de tres cifras) y transcurrido un tiempo, se muestra en la pantalla del televisor. También pueden existir “enlaces", saltos o accesos rápidos entre páginas en la última fila de la pantalla, que se activan a través de los botones de colores que posee el control a distancia. Algunas páginas son de tipo rodante, lo que quiere decir que van apareciendo distintas “subpáginas” cada cierto tiempo, si bien el número de página es el mismo. Esto se indica (por ejemplo) así: 2/4, lo cuál quiere decir que estamos viendo la “subpágina” 2 de 4 que existen en total.

Se han aprovechado las líneas del sincronismo vertical para la transmisión del teletexto que emplea un sistema de codificación digital. De este modo, los receptores no preparados simplemente ignoran estas líneas, y los preparados pueden captar esta señal sin afectar a la calidad de la imagen.

1.5.2.2 Closed Caption [24]

Closed Caption (CC) es un sistema de texto que permite incorporar a la señal de video una serie de mensajes para las personas con dificultades de audición que

les permite leer subtítulos de lo que sucede en la pantalla de televisión. A diferencia del subtitulaje común, el sistema Closed Caption permite la descripción de todo el contenido de audio presente (incluyendo música de fondo y efectos de sonido) mediante palabras o símbolos, mientras que el subtitulaje común solo traduce los diálogos presentes.

Este sistema no sólo sirve de utilidad a las personas con dificultades de audición, también sirve a:

- Personas que están aprendiendo a leer.
- Personas que quieren ver un programa en un lugar público pero no pueden oírlo por el ruido externo de éste.
- Extranjeros que quieren aprender a leer el idioma del programa con CC.

1.5.2.3 Subtitulación en la Programación

Esta es la aplicación más común y que se puede encontrar en casi todos los videos en otro idioma, cuya finalidad es el de poder ayudar al televidente a comprender los diálogos que se producen en las imágenes que se están observando. Este texto es preparado y colocado en las imágenes para que sea desplegado en la parte central inferior de la pantalla con el fin de observarlas sin interrumpir la imagen total que está siendo vista.

De todo lo dicho anteriormente se puede concluir que la información que se presenta como un texto sobrepuesto en las imágenes es una forma de presentar información, sin alterar la concentración del televidente en la imagen presentada y más bien le ayuda para que pueda comprender de mejor manera lo que está viendo.

CAPÍTULO 2. DESCRIPCION DE LOS COMPONENTES PRINCIPALES DEL PROTOTIPO

2.1 INTRODUCCION

El diseño de todo sistema se desarrolla tomando en cuenta las necesidades de los usuarios, con el fin de ofrecer las mejores soluciones. El diseño de este prototipo suplirá la necesidad de mostrar información a las personas que viajan en una unidad de transporte. Esta información tomará principalmente en cuenta que los sitios por donde pase la unidad de transporte, tengan algún valor para el viajero, de modo que se pueda conocer dónde se encuentra y de esa forma pueda tomar alguna decisión según se desee, además siempre es interesante y útil conocer cuánto tiempo falta para llegar a un destino final y a qué velocidad se está viajando.

2.2 PROPÓSITO DEL SISTEMA

El propósito del presente trabajo es diseñar e implementar un sistema que permita mostrar a los usuarios de una unidad de transporte, información referente a la ruta por la cual se desplazan, en la pantalla de una televisión.

2.2.1 Objetivos

2.2.1.1 Objetivo General

Diseñar e implementar un prototipo que permita mostrar información, a los usuarios de la unidad de transporte, en la pantalla de televisión sin alterar el uso normal de la misma, esto implica sobreponer los caracteres generados al video original, mostrando los sectores por los cuales se está desplazando, la velocidad a la que viajan y el tiempo aproximado que necesitarían para llegar a su destino.

2.2.1.2 Objetivos específicos

- Implementar el prototipo utilizando un microcontrolador AVR de la serie MEGA.
- Determinar la forma de obtener la información para ser mostrada.
- Diseñar el hardware que se necesita para que el prototipo entre en funcionamiento.
- Determinar la frecuencia y el tipo de la información a ser mostrada, para los usuarios de una unidad de transporte.

2.2.2 Alcance del sistema

El uso de los sistemas de transporte, terrestre en nuestro país es muy difundido, pero a la vez es uno de los más descuidados en cuanto a brindar a los usuarios mayores facilidades, como los tienen otros medios de transporte, como el servicio aéreo para el transporte internacional. El área de servicio terrestre más descuidada es el servicio interprovincial, este sería el grupo al cual se llegaría con este proyecto, sabiendo que gran cantidad de usuarios son turistas nacionales y extranjeros, quienes serían los beneficiarios directos de este prototipo, pues conocerían mejor las rutas por las cuales se desplazan, haciendo que su viaje sea más instructivo.

Actualmente la única información que tienen los turistas son los mapas de carretera, pero este prototipo también les ayudaría, y a la vez les permitirá a las compañías de transporte tener una herramienta para mejorar su servicio, dando como resultado mayor calidad en el servicio de transporte.

2.3 DESCRIPCIÓN GLOBAL

Este prototipo consta de 2 etapas: la adquisición de datos y la puesta en funcionamiento.

2.3.1 Etapa de adquisición de datos

Cuando nos referimos a la adquisición de datos, es necesario comprender que la ruta en la cual entrará en funcionamiento este prototipo, necesita ser definida de antemano para que se tomen los datos de interés para los usuarios de la unidad de transporte.

En el momento de tomar los datos que van a ser ingresados en el prototipo, se tendrá en cuenta detalles como por ejemplo si el sitio escogido para mostrar la información es una ciudad o un pueblo o algún paisaje natural o el destino del viaje, en fin, la información ayudará al usuario a tomar alguna decisión.

Para ubicar un punto especifico dentro de una ruta, se necesitarán los datos de latitud y longitud de dicho punto, que será el dato que el prototipo procesará. Junto con cada posición se asociará un nombre que el usuario observará desplegado en la pantalla de televisión. En el capítulo 4 se podrá apreciar la manera cómo se hace este proceso con un ejemplo de demostración.

2.3.1.1 Características del usuario

El usuario de una unidad de transporte interprovincial en la mayoría de los casos es una persona que no conoce de manera detallada la ruta por la cual se desplaza, es decir son turistas, que pueden ayudarse en su viaje por medio de la información que se mostrará en la pantalla de televisión.

La información mostrada en la pantalla de televisión le resultará un fastidio al usuario, si esta interrumpe el uso de la misma, es decir si los datos presentados ocupan mucho espacio por su tamaño o posición ocultando la imagen que estaba observando. Además si se muestra por mucho tiempo el mismo mensaje hará que se pierda el interés por la información.

2.3.2 Puesta en marcha del prototipo

Una vez que se ha programado la ruta por la cual se ha de desplazar, se necesitaría únicamente el suministro de energía adecuada, y la conexión a la televisión que le permita mostrar la información debiendo tener un puerto de entrada de video del tipo RCA[4] para tal efecto.

2.4 REQUERIMIENTOS FUNCIONALES

Los requerimientos funcionales de este prototipo son los siguientes:

1) Almacenar la información de la ruta que se ha seleccionado de manera ordenada, una vez que se ha elegido para ser mostrada en cada punto de interés.

2) Tener la capacidad de mostrar caracteres legibles y que presenten la información sin que se perturbe el uso normal de la televisión.

3) Iniciar y suspender la presentación de la información que el usuario observará, durante un tiempo determinado, de acuerdo a la posición en la que se encuentre la unidad de transporte.

4) A partir de una posición definida en los datos almacenados establecer un área en la cual se mostrarán los mensajes asociados a dicha posición.

5) Funcionar alimentado por la fuente de energía que es común en los buses de transporte produciendo el menor consumo posible.

[4] Entrada no balanceada utilizada para la transmisión del video presente en la mayoría de los equipos de televisión identificado con el color amarillo.[25]

2.4.1 Adquisición de datos

Para realizar esta tarea se debe tener en cuenta la ruta en la que se hará uso de este prototipo. Una vez considerado esto, el prototipo almacenará la información a ser mostrada en cada punto de interés seleccionado. Para esto se utilizará un programa llamado Google Earth, el cual provee de información acerca de una región determinada.

2.4.1.1 Google Earth [26]

El Google Earth es una aplicación de acceso gratuito que permite obtener información variada acerca de una región determinada conocida como información georefenciada, que es la información de un punto especifico sobre un mapa. Este programa se instala como una aplicación gratuita, que es una versión limitada, dándole la facilidad de acceder a un servidor, en calidad de cliente, que le provee de información como, la posición dada en latitud y longitud, el mapa de la región desde una vista satelital, información sobre nombres de ciudades y lugares naturales. Se necesita contar con una conexión a Internet, con una buena velocidad de descarga, ya que los mapas de esta aplicación se descargan en tiempo real, a través de *streaming*[5].

En la Figura 2.1 se muestra un ejemplo de cómo se puede obtener la posición de un punto seleccionado, en este caso es el campus de la Escuela Politécnica Nacional asociado con sus coordenadas geográficas. Para la ruta en la que se utilizará el prototipo, esta es la forma en que se obtendrán los datos de cada uno de los puntos de interés para almacenarlos. Junto con estos datos el nombre asociado a dicho punto, que será lo que el usuario podrá observar.

[5] Es un término que se refiere a ver u oír un archivo directamente en una página *web* sin necesidad de bajárselo antes al ordenador o computador.[27]

Figura 2.1. Fotografía satelital de la EPN obtenida del Google Earth

Una vez realizado el proceso anteriormente descrito, se termina este proceso ingresando un mensaje de bienvenida a los usuarios de la unidad de transporte y un mensaje para cuando el vehículo llegue a su destino. Con todos estos datos se los ubica en la parte correspondiente en el programa como datos almacenados.

2.4.2 Información a ser presentada

Una vez que se ha hecho la adquisición de los datos de la ruta de transporte, se hace indispensable el poder saber qué información debe ser presentada al usuario en el momento que éste ocupa la unidad de transporte.

La información que se mostrará utilizará el dato de la posición para poder ubicar el o los mensajes asociados a dicha posición, considerando cierta área en la cual dicho mensaje sea válido, con el fin de hacer referencia al sitio por el cual se desplaza el vehículo, luego se mostrará el tiempo aproximado para llegar al destino final del viaje. También se indicará la velocidad a la que se desplaza el vehículo así como la hora actual.

Con esta información el usuario tendrá una idea concreta de lo que será el resto del viaje y de los sitios por los cuales está pasando.

2.4.3 Presentación de la información en la pantalla de televisión

La información a ser mostrada, debe ser como se ha dicho antes, legible y además no interrumpir el uso normal de la televisión. Tomando en cuenta esto se considera que la información se ubicará en la parte superior de la pantalla de televisión haciendo que se desplace de izquierda a derecha en una sola línea a una velocidad prudente, ya que la mayor parte de la información para el usuario que se presenta en los videos, como subtítulos o mensajes, se los muestra en la parte inferior de la pantalla.

Teniendo en cuenta lo explicado en los puntos anteriores se procederá a la descripción de los diseños a ser implementados para la construcción de este prototipo. Estos diseños se enfocan a dos áreas: la una es el hardware y la segunda es el software.

2.5 DISEÑO DEL HARDWARE DEL PROTOTIPO

El hardware que se necesita para la implementación de este prototipo debe satisfacer las necesidades del diagrama de la Figura 2.2 que indica lo que se necesita en este proyecto para su funcionamiento.

Ahora se considerará cada una de estas etapas y se detallará el hardware requerido para cada una de ellas en esta etapa de diseño.

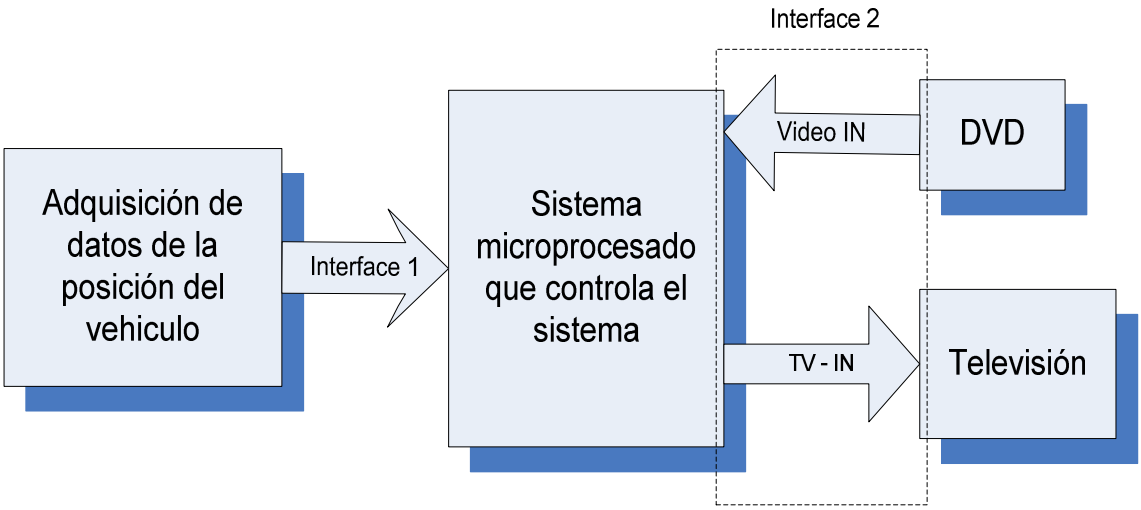


Figura 2.2. Diagrama de bloques del prototipo

2.5.1 Adquisición de datos de la posición del vehiculo

Como se ha mencionado antes, los datos de la posición y los mensajes asociados a la ruta estarán previamente almacenados en el sistema microprocesado, necesitando ahora de información del vehículo cuando éste se desplaza, para poder mostrar la información en los puntos de interés seleccionados.

Figura 2.3. Fotografía del receptor GPS.

Como se vio en el capítulo anterior, un receptor GPS sería el elemento utilizado para obtener los datos de la posición del vehículo mientras éste se

desplaza, y que junto con la interacción con el sistema microprocesado hará que este prototipo funcione adecuadamente. En la Figura 2.3 se muestra el receptor GPS que se utilizará en este proyecto.

Este receptor GPS, modelo SKU: 90-020, de la compañía Delou tiene características que son las adecuadas para este prototipo, las mismas que se describen a continuación. [28]

- El error que produce en las mediciones de longitud y la latitud está entre 5 y 25 metros.
- El error en la medición del tiempo, la hora que entrega, es de 1 ±1µ sec
- El error en la velocidad es de 0.1 m/s.
- Puede entregar información a una altitud máxima de 18,000 m, a una velocidad de 500 m/s.
- Entrega datos cada segundo.
- Utiliza el protocolo RS-232 a 4800bps
- Del estándar NMEA utiliza las sentencias GGA, GSV, GSA y RMC que es la que entrega la información básica de posición, velocidad y tiempo.
- Funciona a un voltaje de 5V.
- Tiene un rango de temperatura entre -40°y 85°C p ara su funcionamiento.
- Puede trabajar en ambientes con humedad desde el 5% hasta el 95%

De las especificaciones que se acaban de indicar, se puede obtener más información en el Anexo 2. En cuanto al estándar NMEA ya se habló en el capítulo 1 y si se desea más información, ésta se encuentra en el Anexo 3.

En la Figura 2.4 se muestra la configuración de los pines del conector del receptor GPS, en donde:

- El pin 1 hace la transmisión de los datos utilizando el protocolo RS-232.

- El pin 3 es por donde recibe la alimentación a 5V.
- El pin 4 utilizado para la conexión a GND.
- El pin 2 no tiene ninguna función.
- El pin 5 no es utilizado.
- El pin 6 no es utilizado.

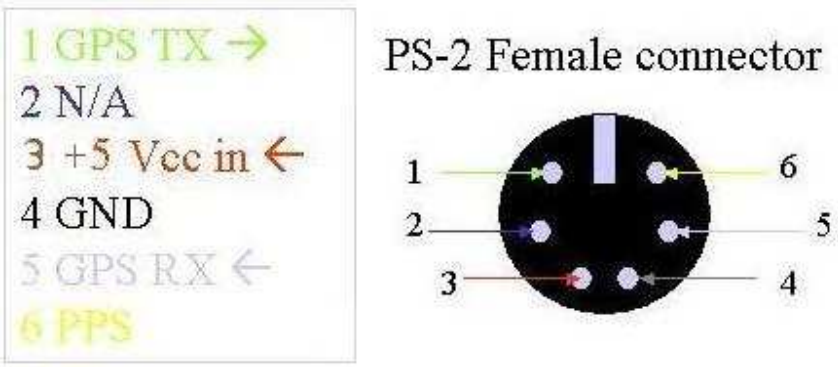


Figura 2.4. Pinout del receptor GPS

Para este prototipo se utilizarán los pines 1, 3 y 4 ya descritos.

2.5.2 Interface 1

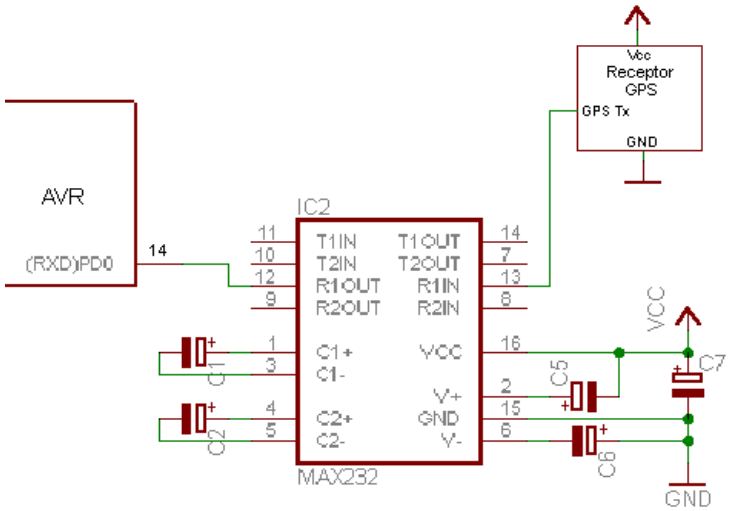


Figura 2.5. Conexión entre el receptor GPS y el microcontrolador

En lo detallado anteriormente se puede apreciar el uso del protocolo RS-232 para la transmisión de los datos, este protocolo también es utilizado por el sistema microprocesado para recepción de datos. Para lograr que esta comunicación se realice, se utilizará el circuito integrado MAX-232 que hará la conversión de voltajes RS-232 a niveles TTL, que son los valores que el microprocesador puede aceptar. En la Figura 2.5 se indica el diagrama de esta conexión.

2.5.3 Sistema microprocesado que controla el prototipo [29]

Esta parte del prototipo será implementada con el AVR ATMEGA16 que tiene las siguientes características:

- Microcontroladores RISC de 8 bits, arquitectura Harvard
- Frecuencia de reloj de hasta 16 Mhz con cristal externo (Xtal1,Xtal2)
- Hasta 16 MIPS[6]
- 16 Kb Flash, 1Kb SRAM, 512 b EEPROM
- Dos contadores de 8 bits con prescaler.
- Un contador de 16 bits con prescaler
- Comunicación serial utilizando USART
- Dos interrupciones externas.
- Voltajes de operación entre 2.7 y 5.5 V.

Estas características, son comunes para los AVR de la serie MEGA tales como el ATMEGA8, ATMEGA16, ATMEGA32 y el ATMEGA64, teniendo como diferencia la cantidad de memoria flash. El número asociado a cada nombre representa la cantidad de memoria flash en kbytes, por ejemplo el ATMEGA8 tiene 8 kbytes y así con cada AVR.

[6] MIPS, abreviatura de Mega Instrucciones Por Segundo.

Para la aplicación a realizarse se necesita gran cantidad de memoria ya que se almacenarán datos correspondientes a los mensajes a ser desplegados, así como los datos de los caracteres que se mostrarán en la pantalla de televisión por lo que se toma como base de este prototipo al ATMEGA16, que tiene 16 kbytes de memoria flash para almacenar y procesar la información. En la Figura 2.6 se muestra la configuración de los pines del ATMEGA16. Para más detalles de este microcontrolador ver el Anexo 1.

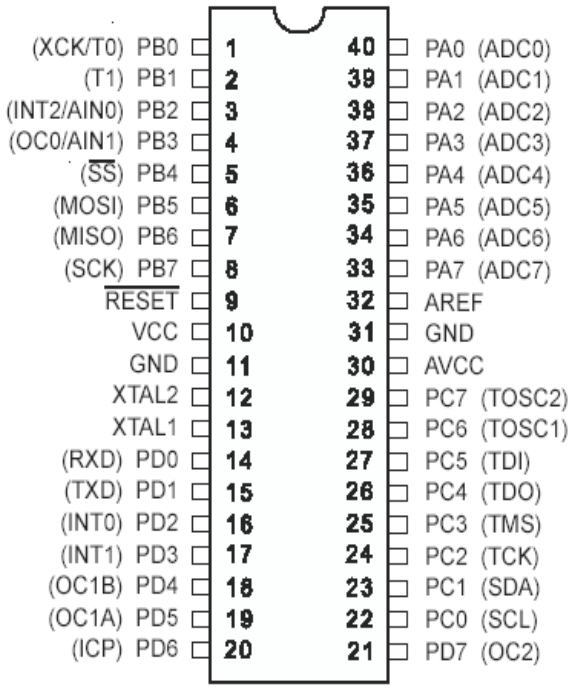


Figura 2.6. Configuración de los pines del ATMEGA16

Debido a que la generación de los caracteres en la pantalla de televisión necesita de alta velocidad de procesamiento, la aplicación que se va a implementar necesita igualmente alta velocidad de procesamiento de las instrucciones, por lo que se utilizará un cristal externo de 16 Mhz. En la Figura 2.7 se muestra la conexión de este cristal externo.

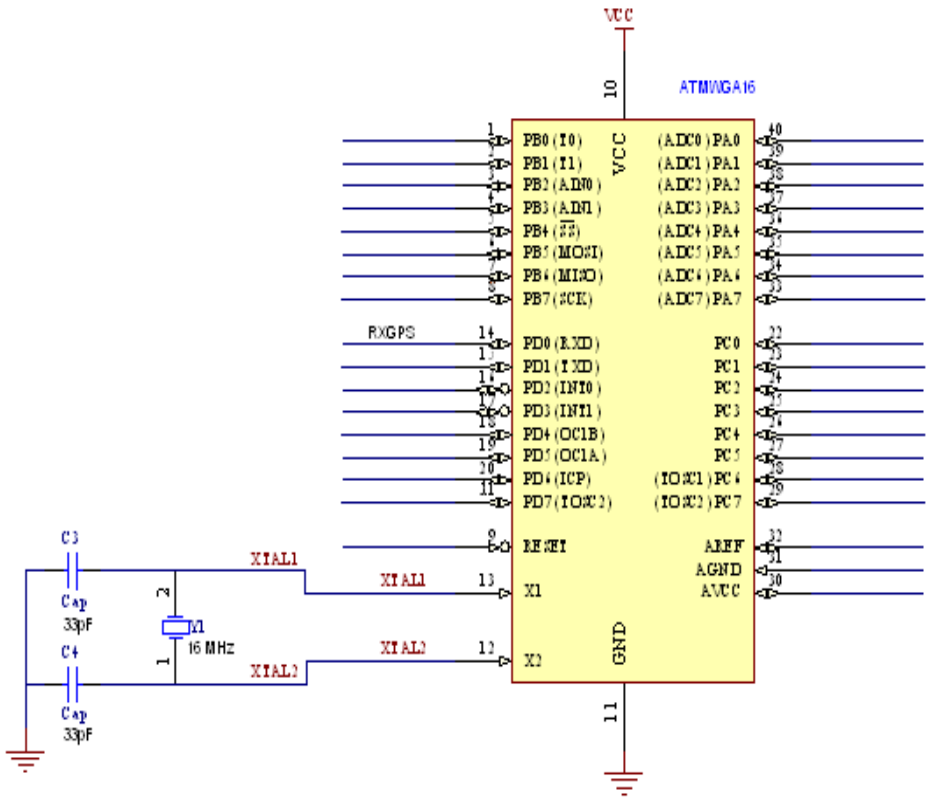


Figura 2.7. Conexión del cristal externo al ATMEGA16

2.5.4 Interface 2

En esta parte del diseño se describirá la conexión entre el ATMEGA16, el reproductor de DVD's y el receptor de televisión, teniendo en cuenta que los niveles de voltaje que entrega el AVR, que son de 5V en sus salidas, no son compatibles con los niveles de señal, utilizados por la televisión para procesar el video, que están alrededor de 1Vp-p. es necesario tener una interface para la conexión.

Partiendo de esta consideración este proyecto utilizará la entrada de video compuesto RCA, para poder ingresar la señal generada por el prototipo al receptor de televisión. Para este efecto se ha diseñado en dos partes esta interface, una es la llamada Video - IN y la otra es la TV - IN que se muestra en la Figura 2.8.

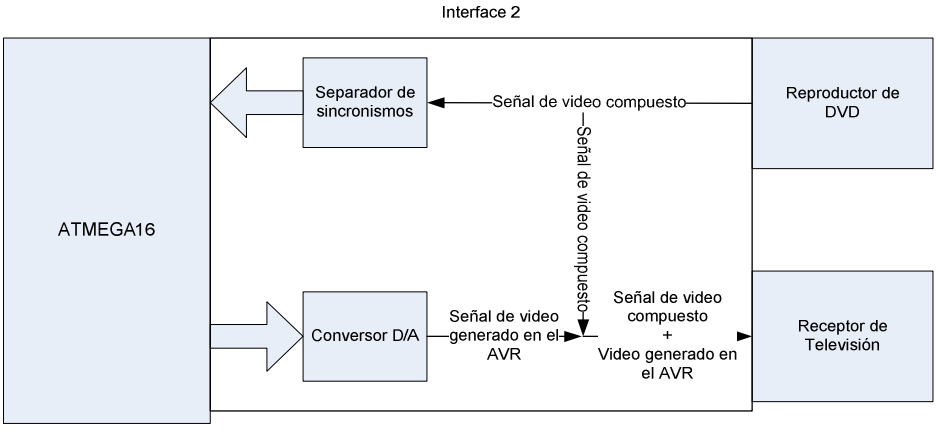


Figura 2.8. Diagrama de la interface 2

2.5.4.1 VIDEO – IN.

Esta parte de la interface tiene como objetivo captar la señal que proviene del reproductor de DVD's, usualmente común en la unidades de transporte interprovincial. Usada para proveer de entretenimiento a los usuarios debido a que el mostrar la programación de los canales no es una opción aceptable, ya que la recepción de los mismos no tiene una buena calidad. Se hace necesaria esta parte ya que se utilizará como referencia la señal proveniente del reproductor de DVD, para poder realizar la sincronización de esta señal y la generada por microcontrolador. Así se logrará la superposición de los caracteres generados en la imagen que proviene del reproductor de DVD.

2.5.4.1.1 Separador de sincronismos [30]

Para lograr esto se utilizará el circuito integrado LM1881 que es conocido como separador de sincronismos, el cual permite descomponer la señal ingresada por uno de sus terminales en sus componentes. En la Figura 2.9 se muestra el circuito integrado con su conexión básica recomendada por el fabricante y el grupo de señales que entrega.

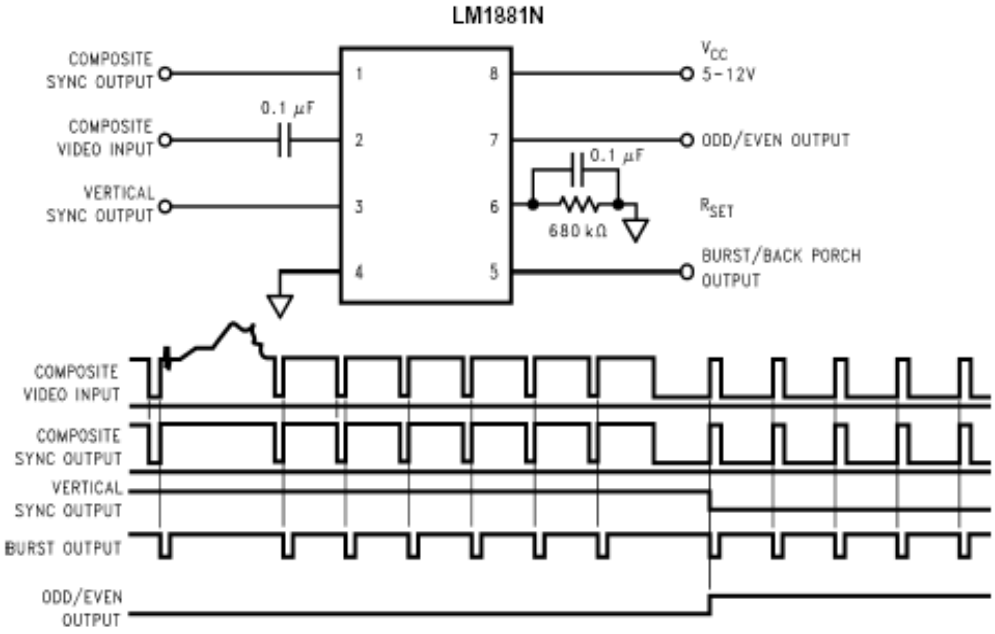


Figura 2.9. Diagrama de conexión del LM181 y de las señales que entrega

Para tener una mejor idea de cuál es el trabajo que desempeña el separador de sincronismos, se lo explicará brevemente:

- El pin 2 recibe la señal de video compuesto de cualquier equipo que produzca este tipo de señal entre valores de 0.5V y 2V p-p. Esta señal es la que será objeto de su proceso en este circuito integrado. Se debe conectar el capacitor de 0.1µF para realizar el desacoplamiento de las señales.
- El pin 1 entrega una señal de video compuesto parecida a la anterior pero con la diferencia que ahora su voltaje máximo es de 5V y no tiene ninguna de las componentes de imagen que tenía la señal original, como se aprecia en la Figura 2.9.
- El pin 3 entrega normalmente un valor de 5V, pero cuando empiezan los pulsos de sincronismo vertical de la señal entrante cambia a un valor de 0V hasta que se terminen todos los pulsos necesarios para el sincronismo vertical.
- El pin 4 es la conexión a GND.

- El pin 5 entrega pulsos de valor 0V cuando es necesario colocar las señales de color dentro de la señal de imagen, útiles para generar señales con colores distintos al blanco y negro.
- El pin 6 es un reset que necesita también un capacitor de desacoplamiento y la resistencia de 680 kΩ para inicializar los valores internos y además ajustar las señales que entrega este dispositivo a la frecuencia de trabajo de las señales de video de 15.734 kHz.
- El pin 7 es una señal que varía entre 0V para el campo par y 5V para el campo impar.
- El pin 8 recibe el voltaje de alimentación entre 5 y 12V.

Para información adicional acerca de este circuito integrado se puede consultar las especificaciones técnicas que se encuentran en el Anexo 4. Aquí se recomienda colocar un filtro en la entrada de video para eliminar todas las componentes que le permiten al televisor generar el color para que la señal sea tratada como una señal que genera video en blanco y negro. En la Figura 2.10 se muestra dicho filtro conectado al circuito integrado.

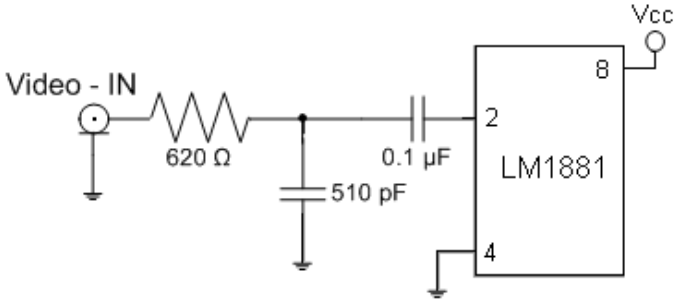


Figura 2.10. Conexión del filtro recomendado por el fabricante

Al calcular la frecuencia de corte para este filtro se tiene lo siguiente:

$$f_0 = \frac{1}{2 * \pi * R * C}$$

$$f_0 = \frac{1}{2 * \pi * 620\,\Omega * 510\ pF} = 503.336\,kHz$$

Ecuación 2.1. Cálculo de la frecuencia de corte del filtro recomendado

2.5.4.1.2 Conexión al microcontrolador

Con la información del circuito separador de sincronismos ahora se procederá a realizar el procesamiento por parte del AVR. Con la lectura de los pines 1 y 3 que se conectan a las interrupciones 1 y 0 respectivamente, se logrará que se desencadenen los procesos necesarios en cada una de las interrupciones. Así se puede lograr obtener el sincronismo necesario para generar la señal que contiene la información de los caracteres a ser desplegados en el receptor de televisión. En la Figura 2.11 se muestra esta conexión.

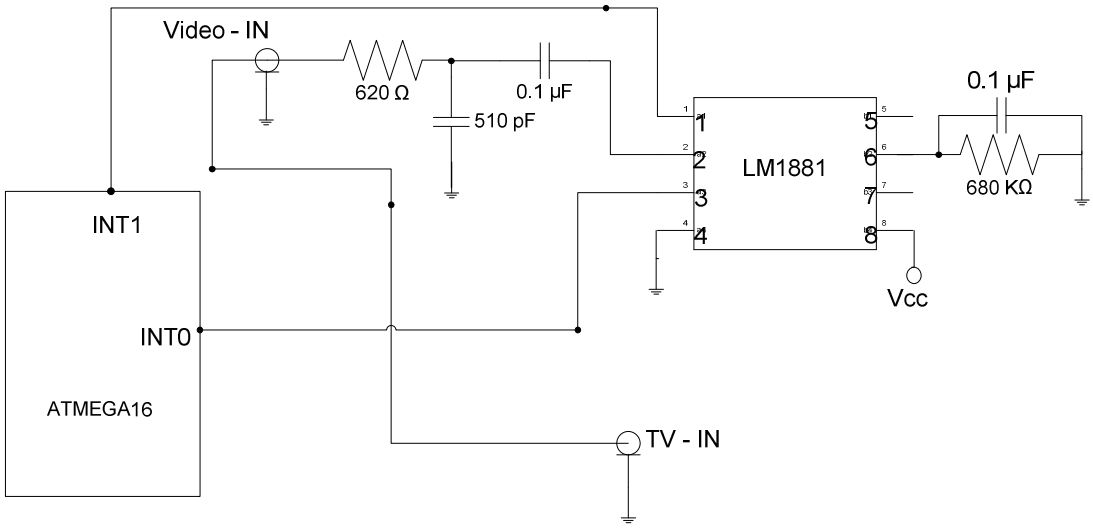


Figura 2.11. Conexión entre el AVR y el DVD

Con la conexión indicada se logra obtener información de la señal proveniente del DVD por la entrada Video-In y ésta se conectará al receptor de televisión por la salida Tv-In sin perjuicio de que la señal en la televisión sufra alguna alteración.

2.5.4.2 TV - IN

Esta parte de la interface está formada por un conversor D/A utilizado para lograr que la señal generada por el AVR sea aceptada por la televisión. Los valores que esta red R-2R entrega se muestran en la Tabla 2.1, considerando que el valor de

1 representa 5V y 0 representa 0V. Además se encuentra la representación de cada valor en la señal de video.

PB1	PC1	TV - IN
0	0	0 V, Usado para sincronismo vertical y horizontal
0	1	0.33 V, Usado para el nivel de color negro
1	0	0.67 V, Usado para el nivel de color gris
1	1	1 V, Usado para el nivel de color blanco

Tabla 2.1. Descripción de los valores entregados por la red R-2R

La conexión se realiza como se señala en la Figura 2.12, tomando como salidas los pines del AVR indicados. Se escogieron estos pines por las siguientes razones:

- El pin PC1 que corresponde al número 23 dentro del AVR, es el encargado de generar los niveles necesarios para generar los caracteres. Adicionalmente se ocupa todo el registro asociado al puerto C para colocar el valor que forma parte del carácter y hacerlo rotar hacia la derecha, por lo cual no se utilizará este puerto para otra conexión. La generación de caracteres se explicará más adelante.
- El pin PB1 que corresponde al número 2 dentro del AVR, es el encargado de generar las señales de sincronismo que son necesarias para la generación de los caracteres, variando esta señal entre el valor de 0V y 0.33V.

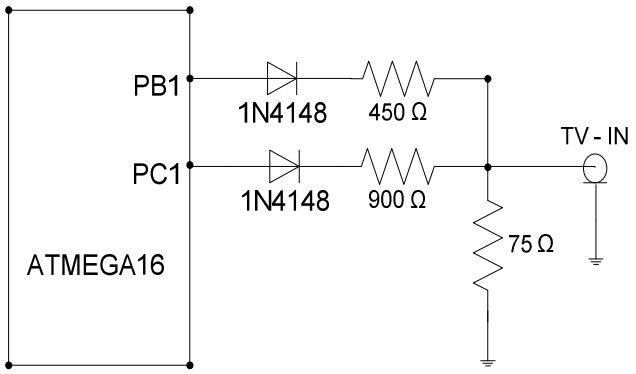


Figura 2.12. Conversor D/A

Entre la red R – 2R y el AVR se colocaron dos diodos que son utilizados para aislar el AVR y la red R – 2R impidiendo que cualquier valor proveniente de la señal de video afecte el funcionamiento del AVR. Adicionalmente limitan el nivel de corriente que circula por la red ya que al realizar cambios de valor de 0 a 5V y viceversa se produce una gran cantidad de corriente que es controlada por estos diodos.

Los valores de las resistencias que producen los niveles de voltaje indicados en la Tabla 2.1 fueron calculados como se muestra en las Ecuaciones 2.2 y 2.3.

Para los valores de 0 en PB1, 5V en PC1 y la resistencia de 75Ω tenemos :

$$\frac{75\Omega}{(75+R1)\Omega} * 5V = 0.33V$$

$$Despejando\ R1 = 900\Omega$$

Ecuación 2.2. Cálculo de las resistencias de la red R - 2R

Para los valores de 5V en PB1, 0 en PC1 y la resistencia de 75Ω tenemos :

$$\frac{75\Omega}{(75+R2)\Omega} * 5V = 0.67V$$

$$Despejando\ R2 = 450\Omega$$

Ecuación 2.3. Cálculo de las resistencias de la red R - 2R

La resistencia de 75Ω es utilizada para asegurar que la señal generada por el AVR sea aceptada por los receptores de televisión ya que la impedancia de entrada del puerto de video RCA es de 75Ω.

2.5.4.3 Sincronización de las señales

Para sincronizar las señales de manera eficiente logrando el efecto de la superposición con una imagen estable, tan solo se unirá la salida del conversor D/A al punto llamado Video-In que a su vez está conectado con la entrada de video Video-In, logrando de esta manera tener el hardware necesario para superponer las imágenes. Todo esto será controlado por el software desarrollado.

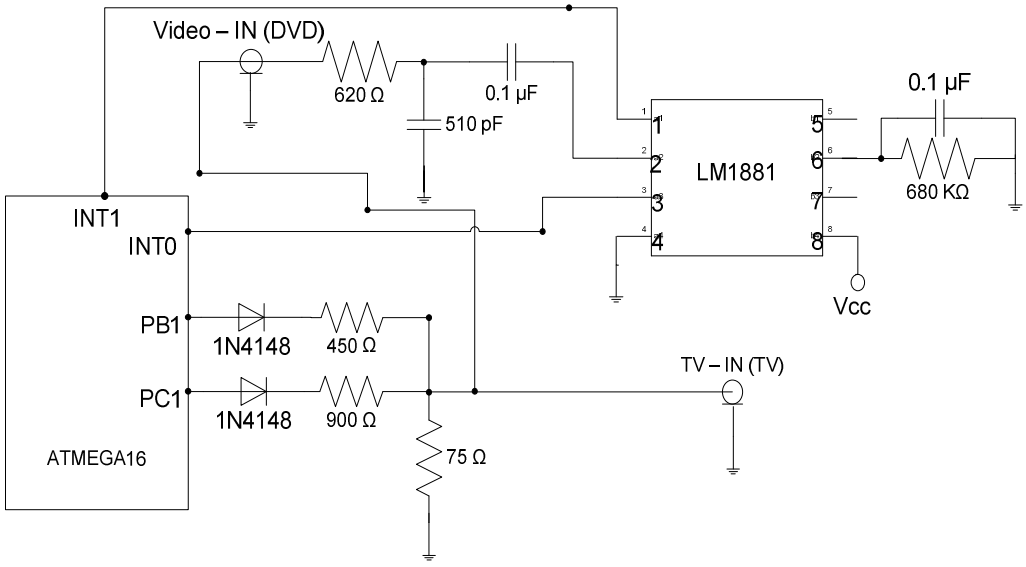


Figura 2.13. Circuito completo de la interface 2

Si el AVR no genera ninguna señal, la presentación de imágenes en la pantalla del receptor de televisión sólo pertenecerá a la señal entregada por el reproductor de DVD, si se empieza a generar señal por parte del AVR, ésta se sumará a la señal del DVD y el efecto será que en el receptor de televisión se muestran las imágenes del reproductor de DVD más los caracteres generados por el AVR. En la Figura 2.13 se muestra la conexión completa de esta interface.

2.5.5 El televisor

En cuanto a esta parte del sistema, sólo se hará la aclaración que la señal generada se ingresará en un televisor con entrada de video que utilice el sistema RCA, ya que si no es así se ignorará completamente. Los niveles de voltaje generados por el microcontrolador y el conversor D/A harán posible dibujar caracteres en color blanco que resaltará sobre la imagen, colocados en la posición antes mencionada para la visualización de los mismos.

2.6 DISEÑO DEL SOFTWARE PARA EL PROTOTIPO

Para desarrollar el software para este prototipo se tienen algunas opciones de entre las cuales se ha de escoger la que mejor se adapte a las necesidades del proyecto.

Para poder programar los AVR existen el lenguaje Assembler que es un lenguaje de bajo nivel. También se puede programar en lenguaje C que, no siendo un lenguaje de alto nivel, da grandes facilidades en relación al Assembler. Los compiladores que trabajan con lenguaje C aceptan también la escritura de programas en lenguaje Assembler, entre éstos tenemos: AVRStudio, Winavr, por mencionar los de mayor acogida.

La otra opción es utilizar un compilador que trabaje con Visual Basic, que es una ayuda grande para los programadores, pues viene con rutinas elaboradas que facilitan la programación de los AVR. Un compilador muy conocido es el Bascom, que también acepta código en Assembler.

Todos estos compiladores al final generan un archivo .HEX que es el que finalmente será grabado en el AVR. Con esta introducción se procederá al diseño en etapas del programa para este prototipo.

2.6.1 Descripción del programa principal

Como se puede apreciar en el diagrama de bloques de la Figura 2.2 se realiza un proceso en el cual la toma de datos de la posición del vehículo desencadena el resto de procesos. Considerando esto se presenta el diagrama de flujo del programa principal en la Figura 2.14.

Este programa comienza con la inicialización de todas las variables globales del sistema, las cuales son las condiciones iniciales del mismo, estas condiciones buscan que el sistema empiece a funcionar desde un estado conocido.

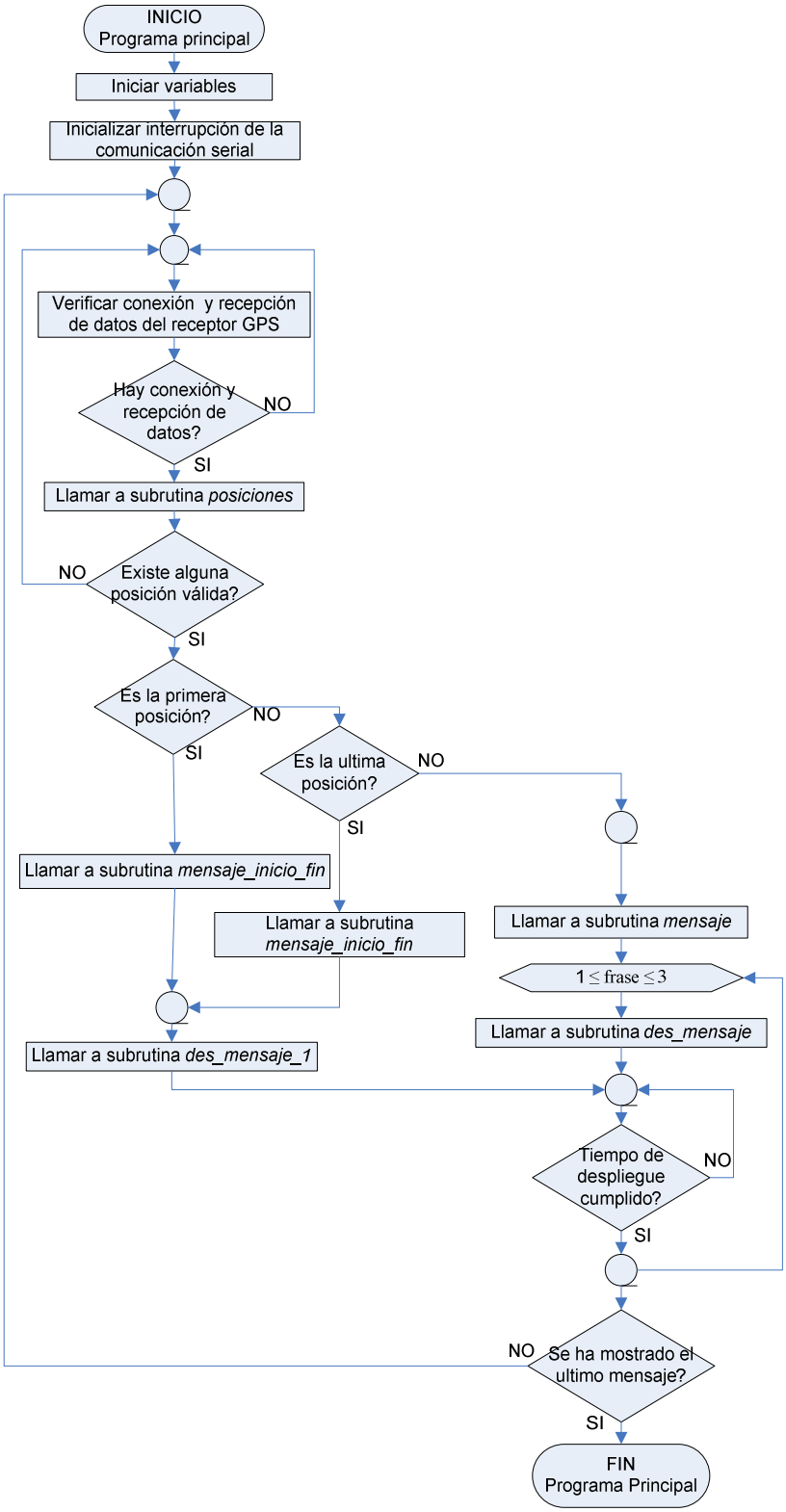


Figura 2.12. Diagrama de flujo del software del prototipo.

Luego se inicializa la interrupción generada por la recepción de datos a través de la comunicación serial, que se asocia a los datos del receptor GPS. Se verifica si están o no llegando datos, lo cual indicaría que si no hay datos la conexión con el GPS se ha interrumpido de alguna forma. Si por el contrario llegan datos, se tiene que revisar si éstos son útiles, ya que en ocasiones el receptor GPS no recibe adecuadamente los datos de los satélites y esto implica que sólo se reciba información de la hora y la fecha. Este error solo se supera esperando a que la recepción del GPS mejore y así poder obtener los datos útiles para la aplicación, esto es latitud y longitud al igual que la hora y la fecha.

En el instante en el que se tienen datos válidos se procede a llamar a la subrutina *posiciones* que es la encargada de revisar si el dato recibido está considerado como un punto de interés en el cual se debe mostrar algún mensaje. Una vez comprobado que es un punto de interés, se verifica si es la primera posición, es decir el origen de la ruta, para ahí desplegar el mensaje inicial, que lo hará la subrutina *mensaje_inicio_fin*. De no ser así, se pregunta si es la última posición y en caso de serlo se llamará a la subrutina *mensaje_inicio_fin*. Cuando se han llamado a está subrutina también se llamará a la subrutina *des_mensaje_1* para mostrar estos dos mensajes.

Si realizadas las comprobaciones indicadas anteriormente, el resultado es NO, se procede a mostrar el mensaje asociado al punto de interés, esto se realiza llamando a la subrutina *mensaje,* que es la encargada de preparar los datos almacenados en una matriz de datos, que será la forma en la que la subrutina *des_mensaje* despliega los datos en la pantalla del televisor.

Cuando se está ejecutando la subrutina *des_mensaje* se controla el tiempo en el que se despliega el mensaje, una vez transcurrido dicho tiempo se verifica si existen mensajes que mostrar y se repite el proceso. Una vez mostrados todos los mensajes se pregunta si el último mensaje ha sido desplegado, si la respuesta es

afirmativa, se da por terminado el programa, caso contrario se vuelve a verificar los datos del GPS y se repite todo el proceso.

2.6.2 Subrutina *posiciones*

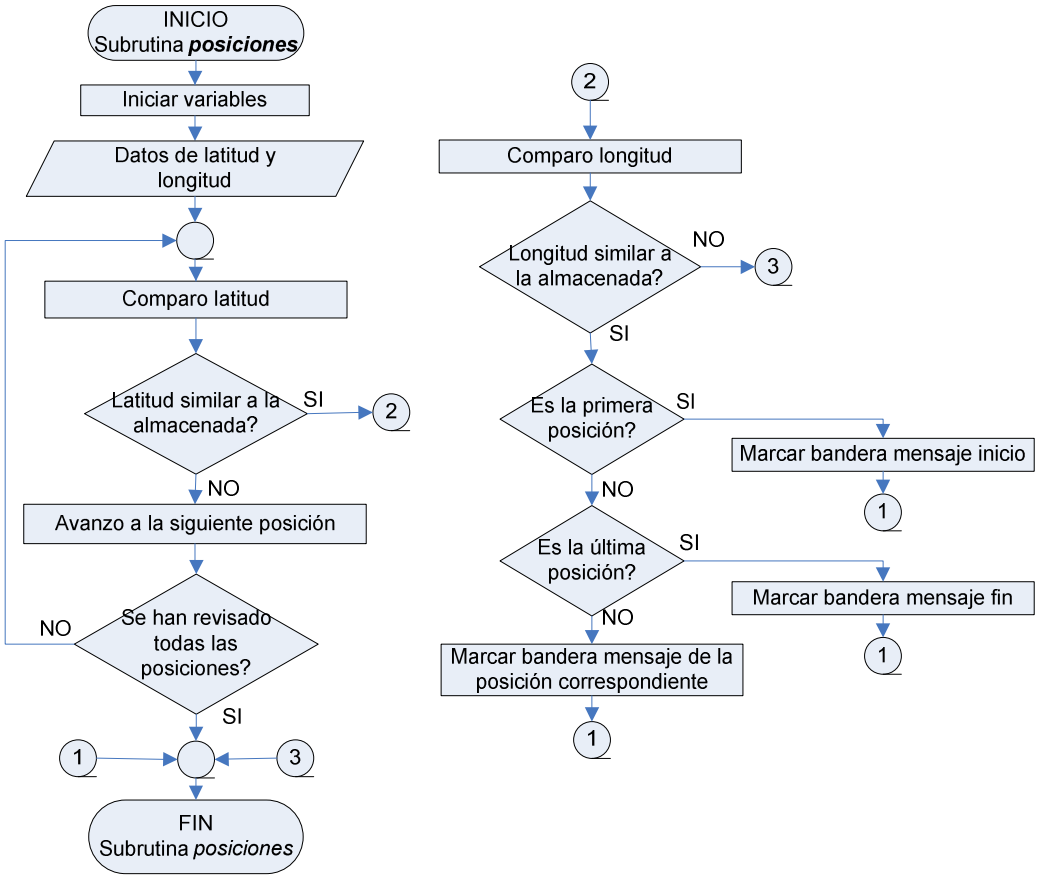


Figura 2.15. Diagrama de flujo de la subrutina *posiciones*.

Cuando se llama a esta subrutina lo que se espera es realizar la comprobación de los datos de posición recibidos con los puntos de interés almacenados, sea que coincidan completamente o estén dentro de un área determinada. Si esta comprobación es positiva se procede a marcar cuáles son los mensajes asociados a dicha posición ya sea que se deba mostrar el mensaje de inicio, el mensaje final o el mensaje de una determinada posición. Para esto se marcarán banderas que indicarán cuál mensaje debe ser mostrado. En la Figura 2.15 se muestra el diagrama de flujo de esta subrutina.

Para poder aceptar una lectura como válida dentro de una posición, se hace la diferencia en valor absoluto entre la latitud de la posición almacenada y la latitud recibida. Si esta diferencia es menor o igual a 1’ (un minuto) que equivale a 1840 metros, entonces se procede a verificar la longitud que debe cumplir con la misma condición que para la latitud. De esta forma se estará creando un área de 3,385,600 m^2 alrededor del punto de interés. En la Figura 2.16 se muestra esta situación.

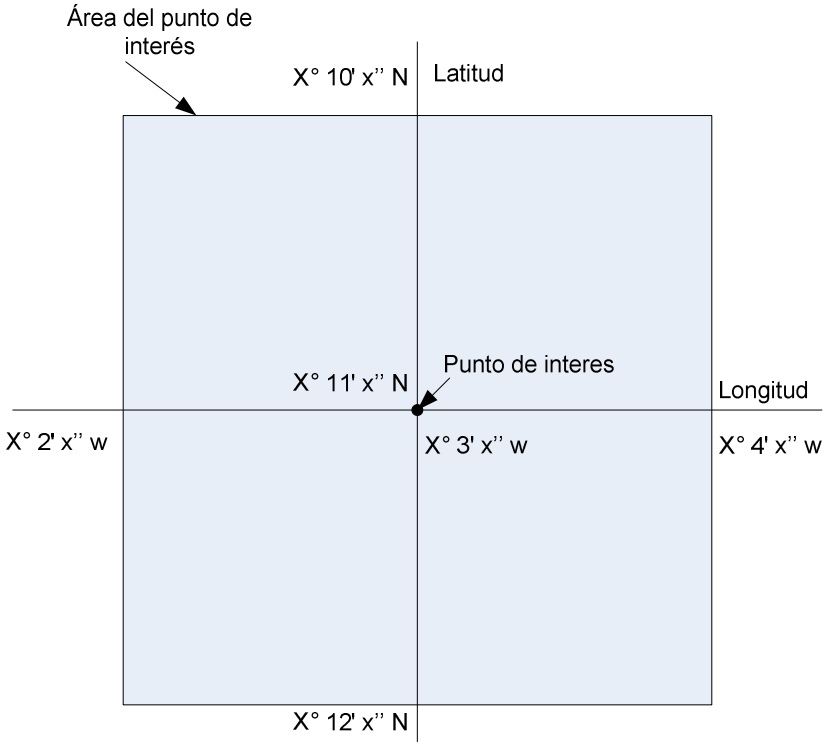


Figura 2.16. Descripción del área alrededor de los puntos de interés

2.6.3 Subrutina *mensaje*

Cuando se llama a esta subrutina, el objetivo es seleccionar los mensajes almacenados para cada punto de interés. Se recibe la marca del mensaje seleccionado y se procede a preparar la matriz de datos que se necesitará para desplegar los mensajes en la pantalla del televisor.

Como se puede apreciar los mensajes de inicio y de fin son marcados de manera diferente, ya que su presentación también es diferente y por eso es necesario la distinción, en este caso se presentarán líneas de manera fija sin que se desplace el mensaje, en cambio para el mensaje de las distintas posiciones se lo hará en una sola línea que se desplaza de izquierda a derecha.

Para este efecto se carga en frase[i] la información a ser mostrada para que se llame a la subrutina *filtro*, que convertirá la información almacenada en una secuencia de valores que serán tomados para asociar cada carácter del formato ASCII al tipo de caracteres que serán mostrados en la pantalla de televisión. En la Figura 2.17 se muestra el diagrama de flujo de esta subrutina.

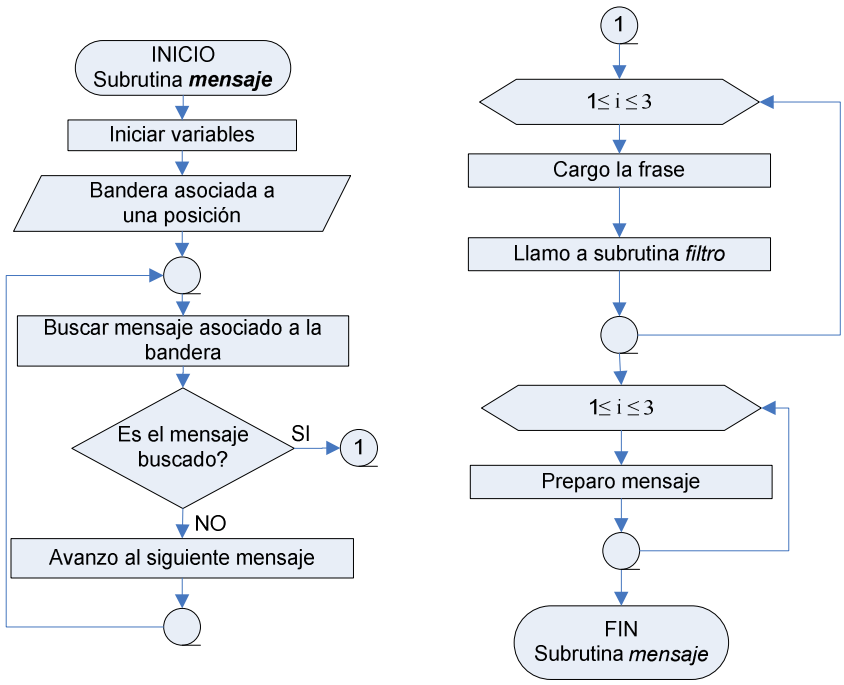


Figura 2.17. Diagrama de flujo de la subrutina *mensaje*.

Ya con frase[i] procesada se asocia cada valor de la misma con el carácter que será mostrado en la pantalla del televisor en una matriz de datos que ordenará los caracteres para que sean mostrados en la pantalla considerando que se desplazarán de izquierda a derecha. La matriz está formada por los valores equivalentes de cada uno de los caracteres ASCII a una tabla de caracteres creados

para ser mostrados en la pantalla de televisión. La tabla completa se muestra en el Anexo 5.

Como se muestra en la Figura 2.18 cada byte dentro de la matriz contiene los datos de una parte del carácter y ordenados como se muestran, se leerán línea a línea y el resultado de esto será una imagen, en la que se pueden ver los caracteres. El otro detalle a tomarse en cuenta es, que todos los caracteres se colocarán en la matriz como una imagen reflejada, ya que para poder presentarlos se hará desplazar el valor correspondiente a la derecha por 5 veces, este es el ancho de cada carácter y por 7 líneas que es la altura de cada carácter.

Para ilustrarlo con un ejemplo, el mensaje original es “HOLA” pero para poder generar la imagen a ser vista, se deben colocar los datos como se muestra en la figura, todos los caracteres creados cumplen con este requisito.

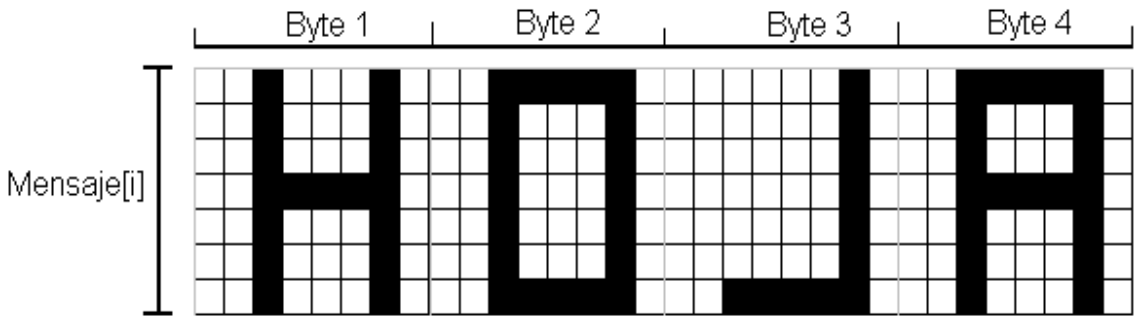


Figura 2.18. Matriz preparada para desplegar los caracteres en la pantalla de televisión

2.6.3.1 Subrutina *filtro*

Esta subrutina lo único que hace es revisar el carácter ASCII y por medio de comparaciones identificar el valor dentro de la tabla de caracteres diseñados que serán presentados en la pantalla de televisión. No importa si se ingresan, en el caso de las letras, mayúsculas o minúsculas ya que el resultado siempre será con caracteres del tipo mayúscula en la pantalla. También se pueden presentar los números, ciertos caracteres comunes como el punto, la coma, los dos puntos, el

espacio. Si algún carácter no es reconocido, se presentará la letra Ñ mayúscula como muestra de que no se tiene ese carácter. En la Figura 2.19 se muestra este proceso.

2.6.4 Subrutina *mensaje_inicio_fin*

Cuando se llama a esta subrutina se procede a preparar el mensaje guardado específicamente para esta posición determinada, creando las líneas que serán leídas por la subrutina que dibujará los caracteres en la pantalla del televisor. En la Figura 2.20 se muestra cómo se realizará el proceso.

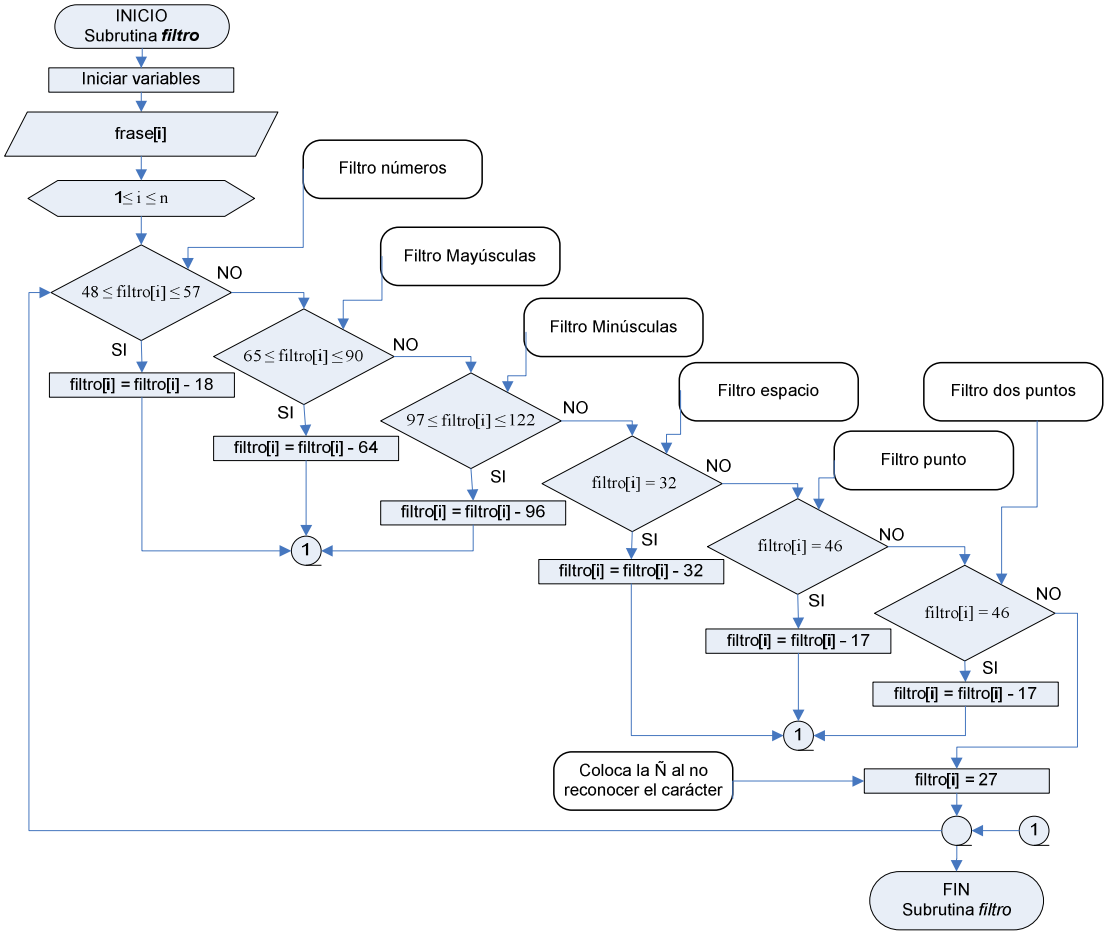


Figura 2.19. Diagrama de flujo de la subrutina *filtro*.

Lo que se destaca en esta subrutina es que se desplegarán algunas líneas en lugar de una sola y el procedimiento de preparación del mensaje es parecido al descrito anteriormente.

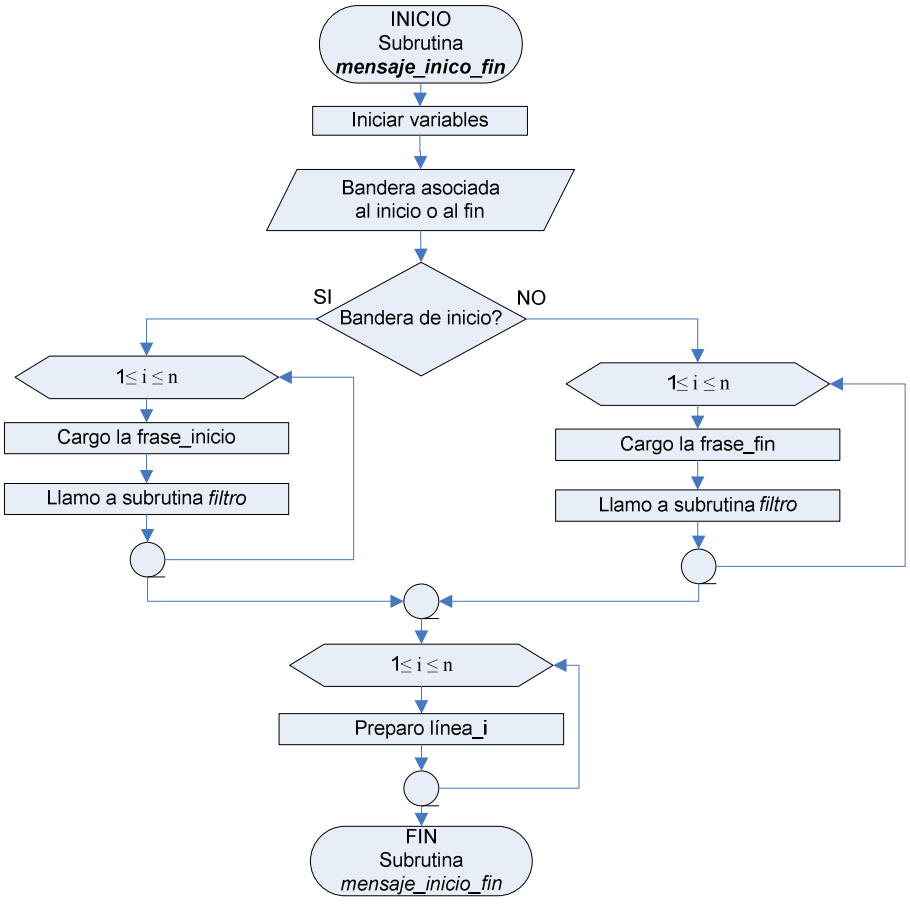


Figura 2.20. Diagrama de flujo de la subrutina *mensaje_inicio*

2.6.5 Subrutina *des_mensaje*

En la Figura 2.21 se muestra un diagrama de flujo del proceso que a continuación se describe. Esta subrutina será la encargada de generar la señal para que sea transformada por el conversor D/A y así se pueda observar en la pantalla del televisor la imagen de los caracteres antes preparados. Para esto se hace la lectura de las matrices de datos etiquetadas como mensaje.

Se hace la lectura de la matriz línea a línea por siete ocasiones iniciando en una línea seleccionada dentro del televisor. Al estar ubicados los caracteres en el orden de presentación se reduce al mínimo el tiempo de localización de cada carácter y así mejorar el rendimiento de mostrar los caracteres.

Antes de realizar la lectura primero se debe leer el dato que entregará el separador de sincronismos que se relaciona con el inicio de los pulsos de sincronismo vertical, antes descrito, y con este dato presente inicializar los valores necesarios para que el D/A entregue el pulso de sincronismo vertical, el contador de líneas igual a 1, no permitir dibujar los caracteres y activar la interrupción externa para poder sincronizar la señal de video entrante y la señal generada.

Con esta inicialización lista, cuando la interrupción se ejecute incrementará un contador que indicará las líneas del campo que se está dibujando. Cuando se llegue a cierto número de líneas, se cambiarán los pulsos de sincronismo vertical por los de sincronismo horizontal. Además una vez que se haya generado un campo se incrementará otro contador necesario para poder desplazar el mensaje sobre la pantalla de la televisión.

Para graficar los caracteres se utilizará un registro llamado VIDEO. VIDEO será rotado hacia la derecha y un bit de este registro está asociado a un pin. Este pin cambiará entre 0 y 1, lo que será suficiente para generar el carácter a ser mostrado. 0 representa el color negro y 1 el color blanco.

Cuando el contador de líneas llegue a un valor específico, se iniciará la presentación del mensaje y cuando tome otro valor ya determinado se detendrá la muestra del mensaje.

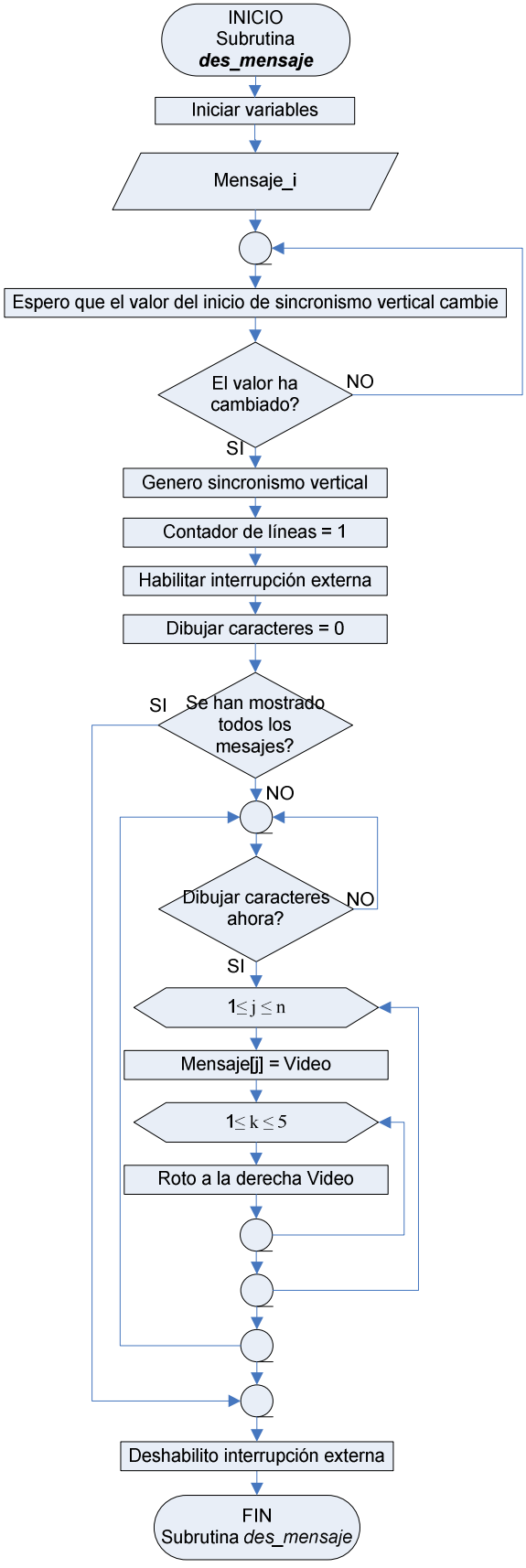


Figura 2.21. Diagrama de flujo de la subrutina *des_mensaje*

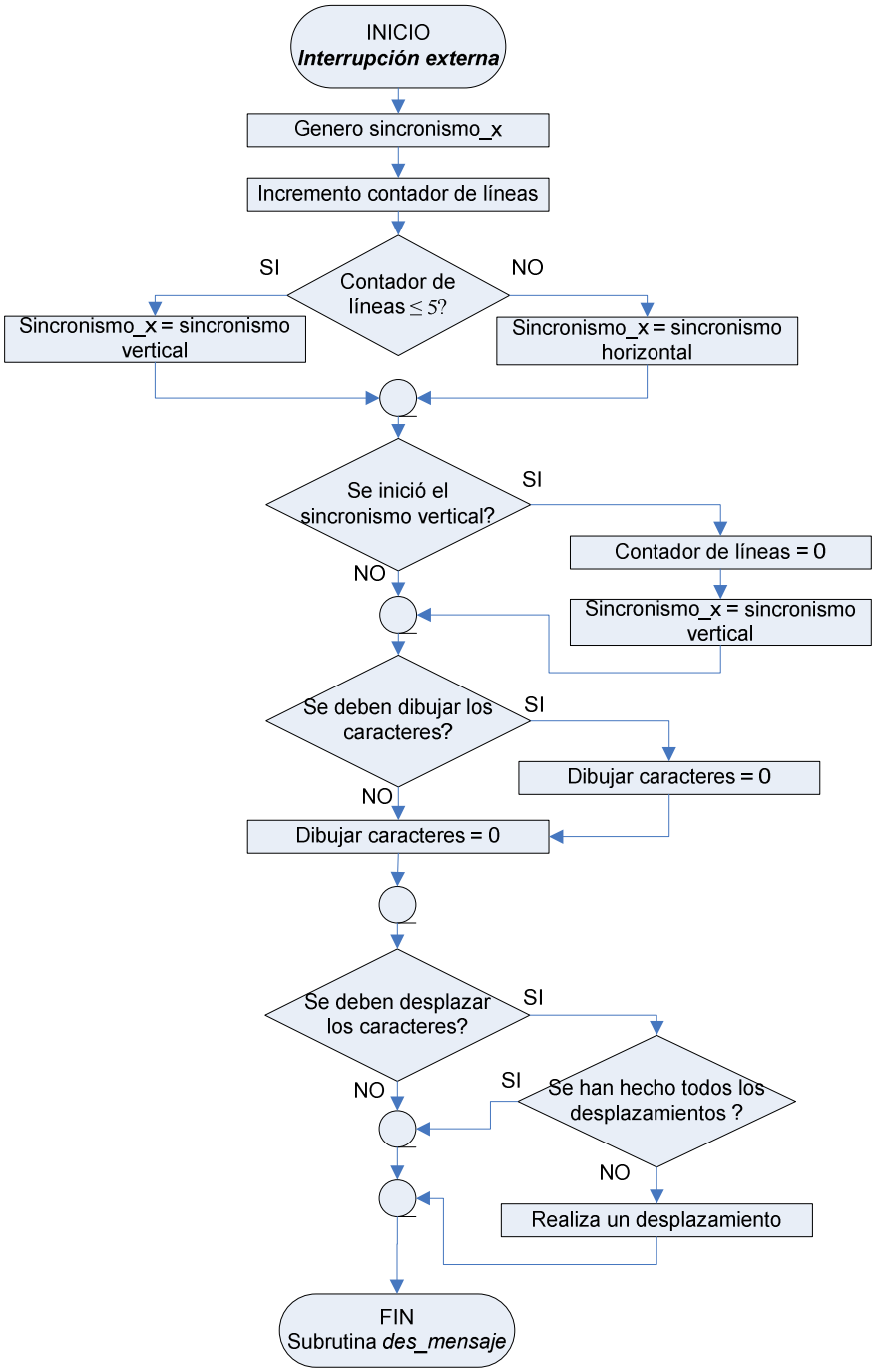


Figura 2.22. Diagrama de flujo de la interrupción externa

Una vez mostrado el mensaje durante el tiempo determinado, se cargará el segundo mensaje y así hasta completar los mensajes a ser mostrados. Completada la muestra de los mensajes se procederá a esperar para mostrar nuevamente

después, de otro instante de tiempo, hasta que se complete el número de veces que se desea mostrar el mensaje.

Una vez realizadas las tareas indicadas se procederá a desactivar la interrupción externa y de esta forma salir de la subrutina para esperar nuevos datos. En todo este proceso la interrupción que se asocia a la comunicación serial se deshabilitará para concentrarse en las operaciones de la generación de los caracteres. En la Figura 2.22 se describe el proceso mencionado.

2.6.6 Subrutina *des_mensaje_1*

En la Figura 2.23 se muestra el diagrama de flujo de esta subrutina, que se diseñó para mostrar los mensajes de inicio y fin de la ruta. A diferencia de la anterior subrutina, el texto presentado no se desplazará sino que se mostrará en algunas líneas.

Al igual que en la subrutina anterior se sincronizará la imagen generada y la señal de video entrante, lo cual activará la interrupción externa para poder hacer los cambios en el sincronismo. Aquí no se tomará en cuenta el desplazamiento del texto.

El tiempo necesario para mostrar estos mensajes estará dado por la cuenta de los campos presentados. Sabiendo que en un minuto se presentan 60 campos, si se desea presentar el mensaje por dos minutos, lo que se esperará es que la cuenta llegue a 120 y se dejará de mostrar el mensaje.

Con todo lo anteriormente descrito se muestra el diseño del hardware y el software que se implementarán en este prototipo.

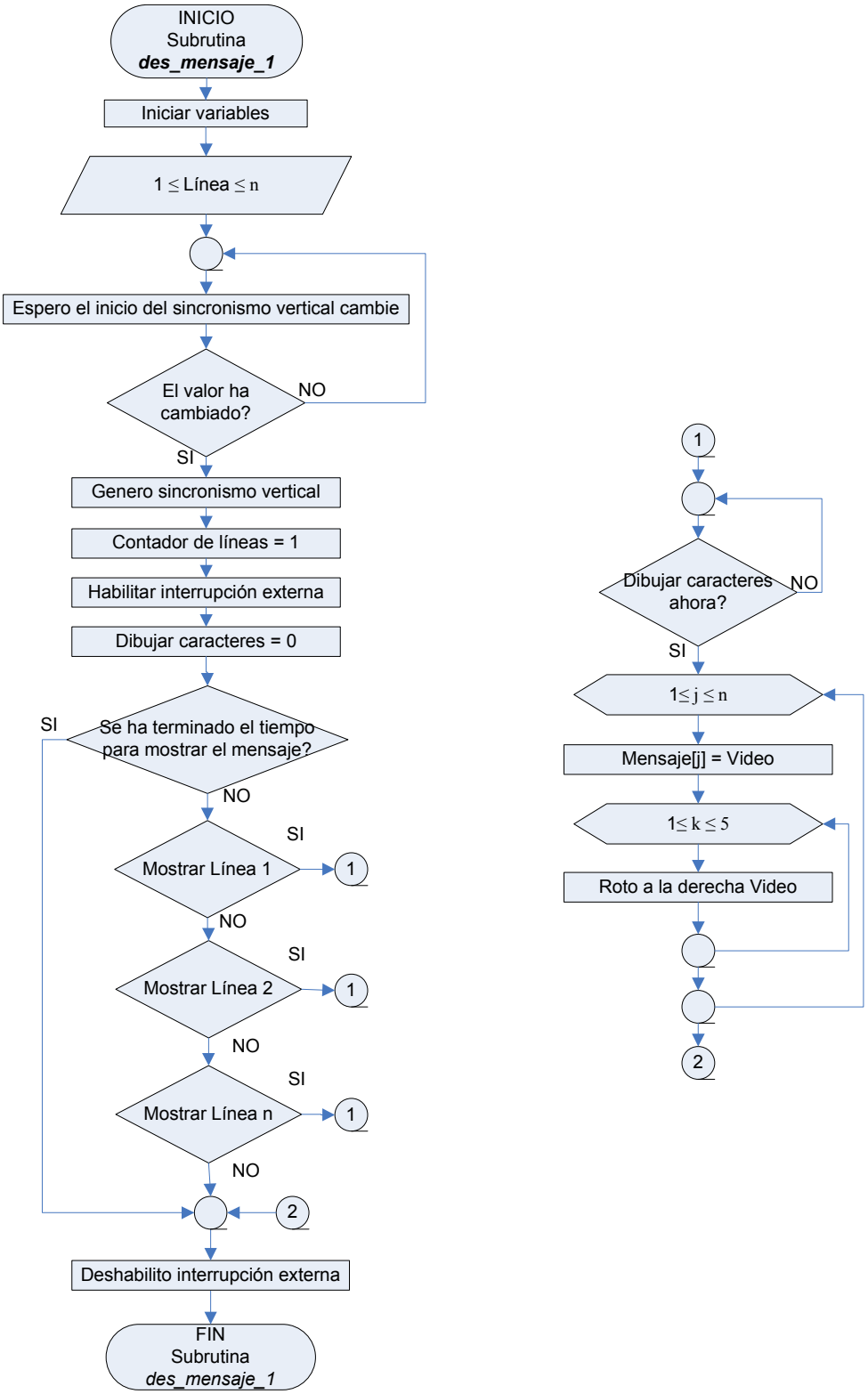


Figura 2.23. Diagrama de flujo de la subrutina *des_mensaje_1*

CAPÍTULO 3. IMPLEMENTACION DEL HARDWARE DEL PROTOTIPO

3.1 INTRODUCCION

Este prototipo está siendo elaborado sobre una tarjeta de desarrollo para los AVR's, diseñado por el grupo SIMON - VE (Sistema de Monitoreo Vehicular), el cual tiene las facilidades de conexión con el GPS, comunicación serial, puerto para la programación por sistema del AVR, entradas a 5V y 8V entre sus funciones más importantes.

La intención de este capítulo es mostrar, el diseño implementado sobre el equipo de desarrollo, para poder realizar el proyecto. También se mostrarán las pruebas del software diseñado.

3.2 IMPLEMENTACIÓN DEL HARDWARE

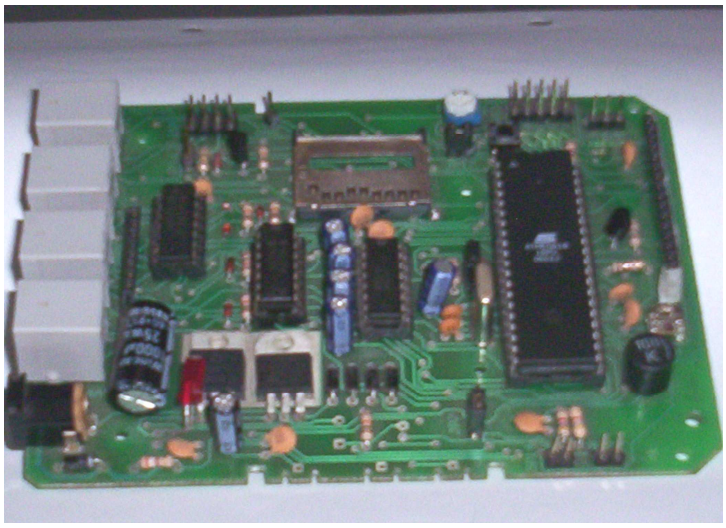


Figura 3.1. Fotografía del equipo de desarrollo

En la Figura 3.1 se muestra el equipo de desarrollo y el diagrama de conexiones de éste en la Figura 3.2.

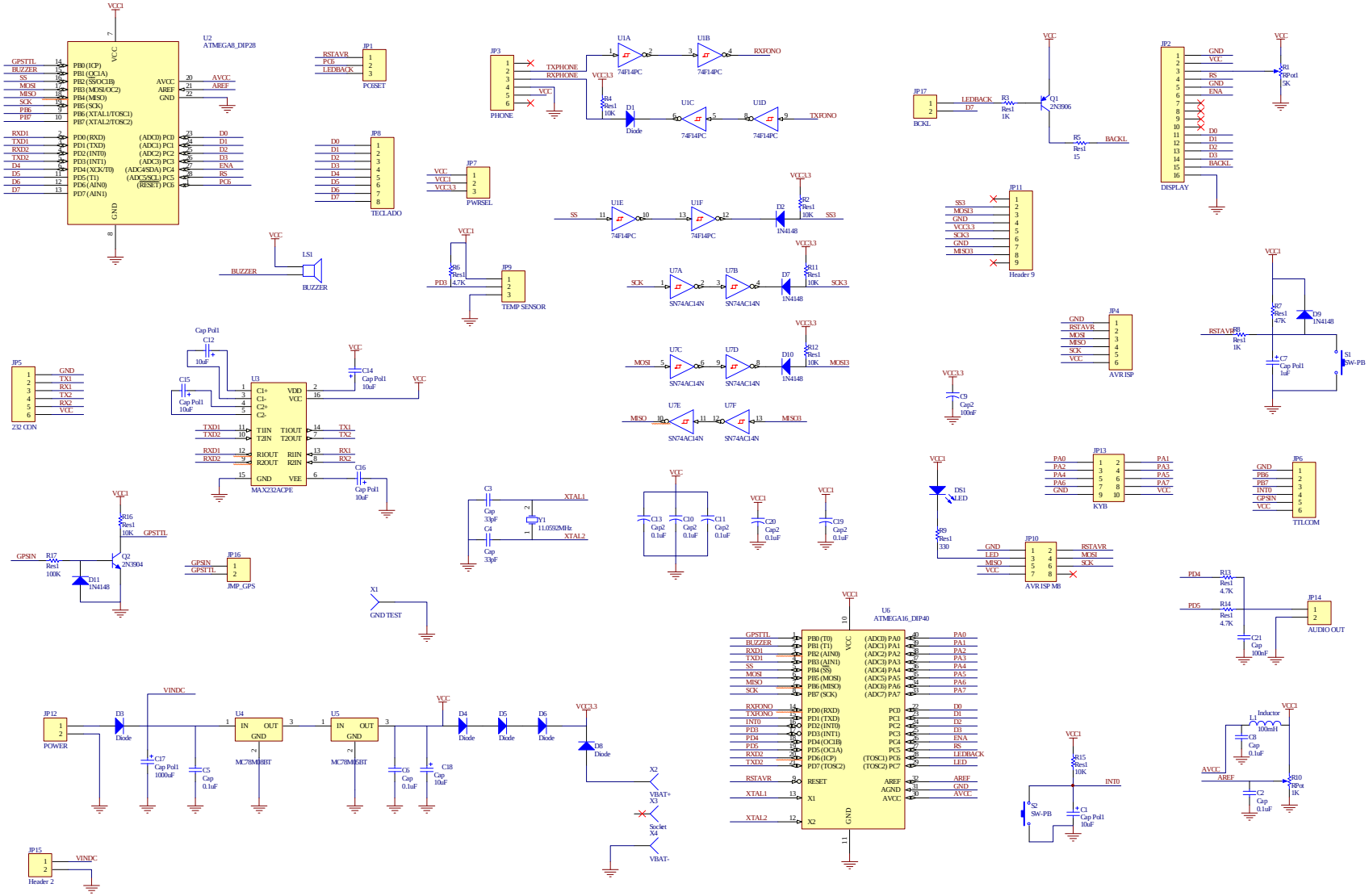


Figura 3.2. Diagrama de conexiones del equipo de desarrollo

A este equipo de desarrollo se procederá a unir el circuito del separador de sincronismos y adicionalmente el conversor D/A necesario para poder entregar la señal generada al televisor. En la Figura 3.3 se muestra el diagrama de conexión de estas dos partes del prototipo.

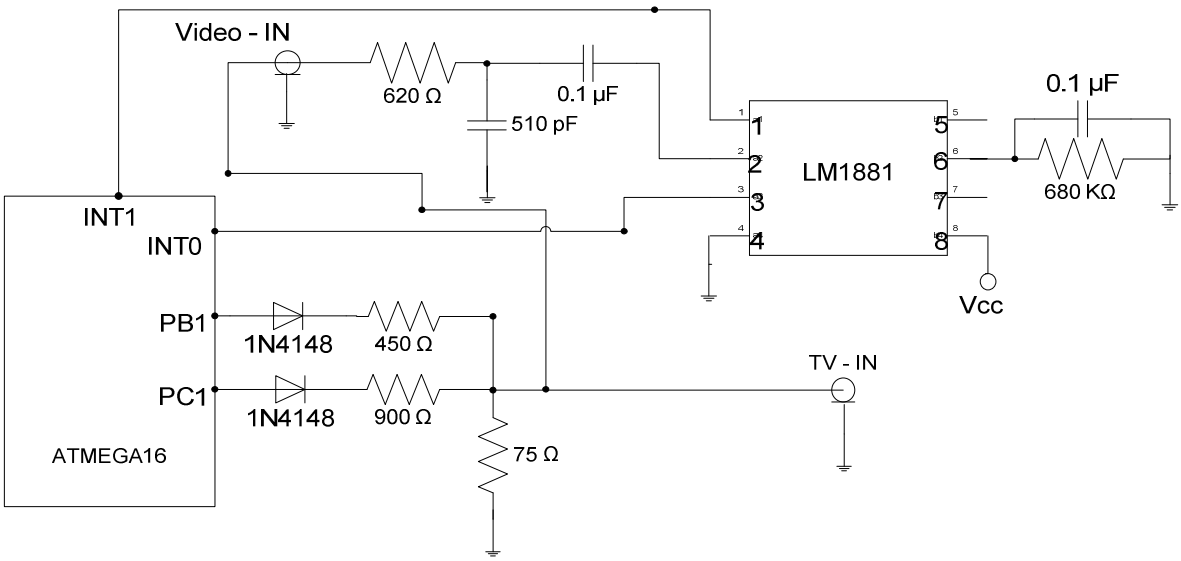


Figura 3.3. Circuito completo de la interface 2 a ser implementado

Esta parte del circuito se implementó sobre una baquelita perforada y se tiene una vista en la Figura 3.4.

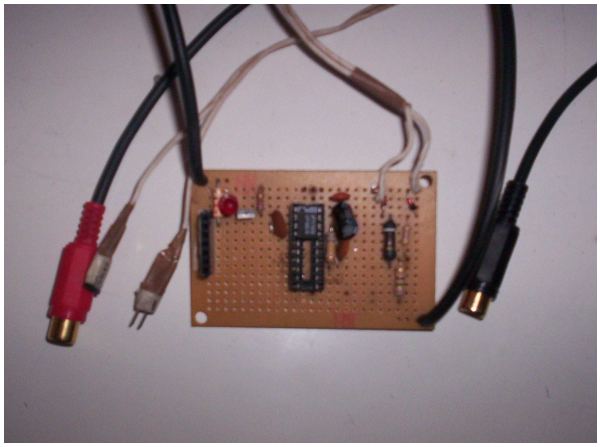


Figura 3.4. Fotografía del hardware implementado

Con este hardware se procederá a realizar la implementación del prototipo. Para esto es necesario implementar el software que permitirá realizar esta tarea.

3.3 SELECCIÓN DEL COMPILADOR

En este punto del proceso lo que se necesita es determinar con qué clase de compilador se va a trabajar en la implementación del software. De los compiladores mencionados en el capítulo 2 vamos a elegir dos: Bascom AVR y Winavr para las pruebas.

Ahora, ¿por qué no trabajar directamente con uno de los dos compiladores para realizar la aplicación?, la respuesta es que debido a que al tratar de generar una señal que se parezca a la señal de video de un televisor estamos hablando de generar una señal de aproximadamente 15.720 Khz con la cual no hay mucho inconveniente. El problema surge al tratar de generar sobre esta señal los caracteres para ser mostrados, ya que en un espacio de aproximadamente 51µs se debe realizar el proceso necesario par realizar esta tarea.

3.3.1 Pruebas con el compilador Bascom AVR

Para realizar las pruebas con este compilador se tratará de generar en la pantalla del televisor una imagen como la representada en la Figura 3.5. Para lograr obtener esta imagen, se tratará de hacer que cada parte a ser generada se la haga por medio de retardos, lo que significa que se ejecuta una instrucción y se esperará un determinado tiempo hasta cambiar a una nueva instrucción, de tal forma que se genere la señal de video mostrada en la figura.

Una vez elaborado el programa y luego de compilarlo y programarlo se obtuvieron los siguientes resultados mostrados en la Figura 3.6.

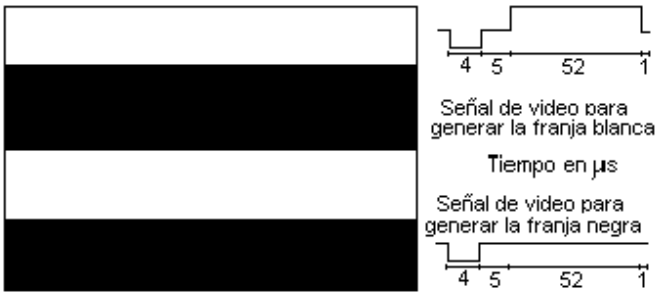


Figura 3.5. Imagen que se intenta lograr por medio del compilador Bascom AVR

Figura 3.6. Resultado del programa compilado en Bascom AVR

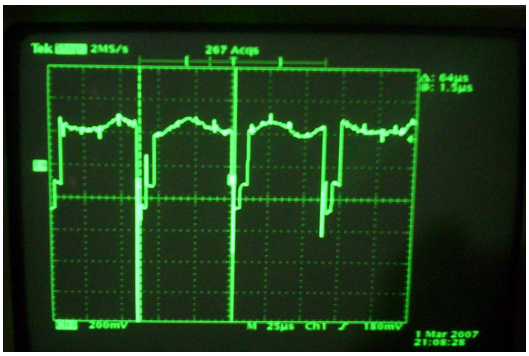


Figura 3.7. Imagen de la señal generada

Otro resultado es el obtenido en la imagen de un osciloscopio que se muestra en la Figura 3.7.

Otro de los aspectos a considerar es la cantidad de espacio que ocupa en la memoria, el programa implementado. Éste dependerá en buena manera de cómo el

compilador transforme sus instrucciones en las correspondientes instrucciones en el lenguaje de máquina, que es lo que realmente el AVR reconoce y ejecuta, para esto se muestra en la Figura 3.8 la pantalla de los datos grabados en el AVR.

Una vez realizadas estas pruebas se procedió a generar la misma señal de video ahora por medio de uno de los *Timers* del AVR. Lo que se pretendía era que cada vez que se desborde el contador se genere una interrupción que generaría el pulso de sincronismo horizontal.

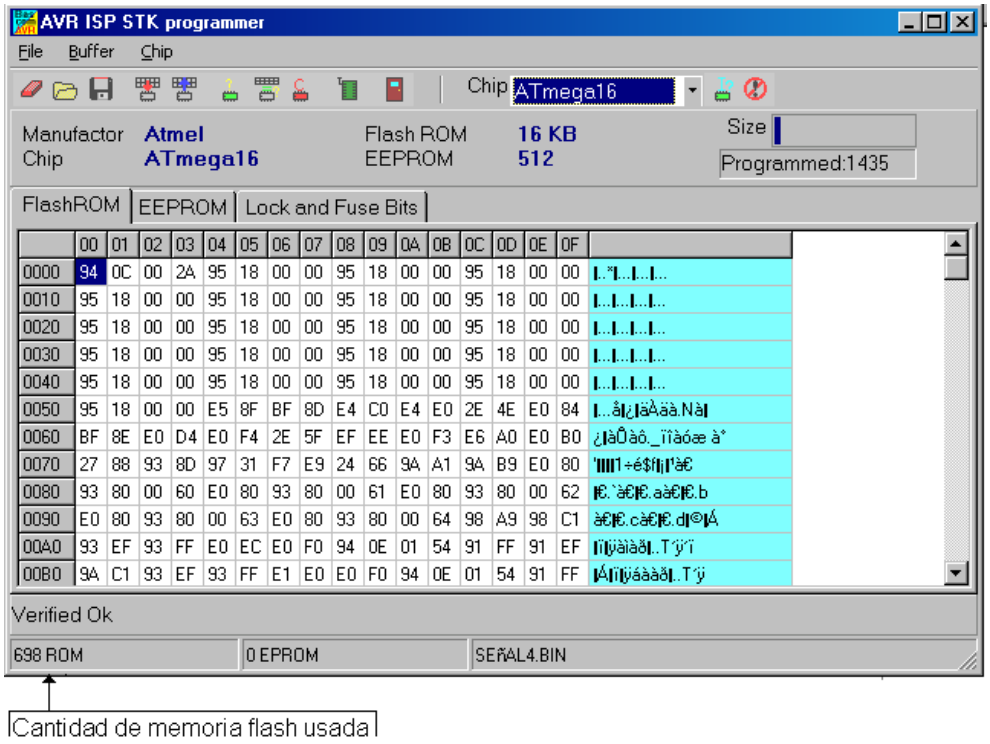


Figura 3.8. Pantalla del programador del Bascom AVR

Para esto se hicieron mediciones del *Timer* con diferentes valores, con prescaler 0 y prescaler 8 a fin de tener una idea clara de cómo se manejarían los tiempos por este compilador. En la Tabla 3.1 se muestran los resultados de estas mediciones hechas con la ayuda de un osciloscopio, donde TCNT0 es el valor que se carga en el registro para obtener el valor que se encuentra en la columna de tiempo en la tabla.

TCNT0	Tiempo en µs	
255	2.4	Prescaler 0
150	13.4	
96	20	
50	25.8	
0	31.8	
255	2	Prescaler 8
203	51.6	
196	58.8	
150	104	
96	157	
0	248	

Tabla 3.1. Valores generados a partir del Timer0 del AVR

TCNT0	Tiempo en µs
0	31.875
18	29.625
51	25.5
69	23.25
96	19.875
105	18.75
132	15.375
150	13.125
183	9
231	3
255	0

Tabla 3.2. Valores teóricos para el Timer0 del AVR con prescaler 0

En cambio en la Tabla 3.2 se muestra una lista de valores teóricos para el *Timer0* considerando un cristal de 8MHz. De estas mediciones se puede apreciar que para los valores marcados existe cierta diferencia que es normal.

Ahora el verdadero problema está en que se trató de lograr la misma señal de la Figura 3.5, pero con la utilización del *Timer0,* lo cual no se logró, la señal que se produjo era una señal inestable que visiblemente no representaba nada, ya que en la pantalla del televisor se mostraba distorsión.

Como conclusión de lo expuesto se tiene que, sí al hacer esta señal tan simple que no requiere de gran cantidad de procesamiento, se tienen estos resultados, ¿qué sucedería el instante en el que se tenga que hacer mayor cantidad de procesamiento?. Definitivamente el compilador no dá garantías de que el tiempo que se cree ocupar sea el real y eso lo vemos por medio de la imagen de la Figura 3.9, donde se aprecia que el valor de la señal para una línea es de 64 µs, cuando en el programa se colocaron los valores que se encuentran en la Figura 3.5 dando una diferencia de 2 µs que es bastante, considerando que es una aplicación tan sensible.

Por lo tanto este compilador no es una buena opción para realizar este proyecto de tal forma que se realizaron pruebas con otro compilador.

3.3.2 Pruebas con el compilador Winavr

Al igual que con el otro compilador se tratará de generar la imagen mostrada en la Figura 3.5 utilizando retardos para lograrlo. Los resultados se muestran en la Figura 3.9.

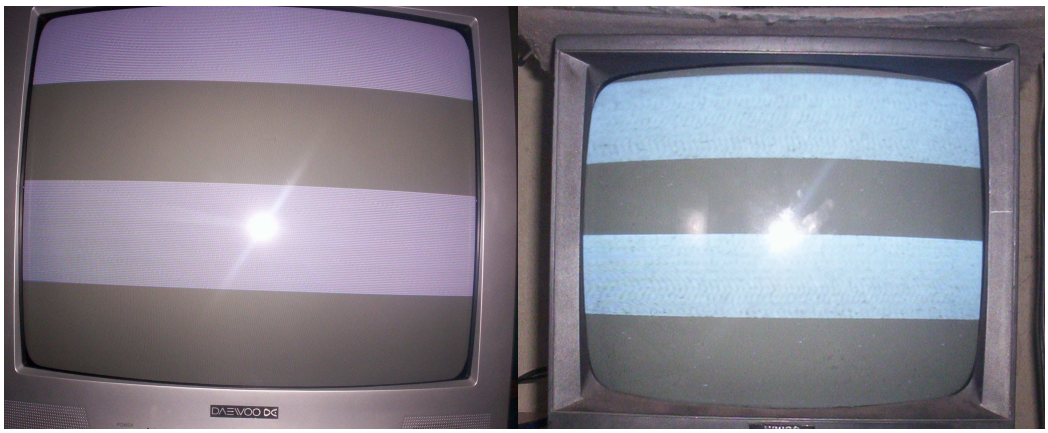


Figura 3.9. Resultado del programa compilado con el Winavr

El programa hecho con el mismo principio produjo mejores resultados en la señal obtenida por este compilador, esto se muestra en la imagen obtenida con la ayuda del osciloscopio en la Figura 3.10.

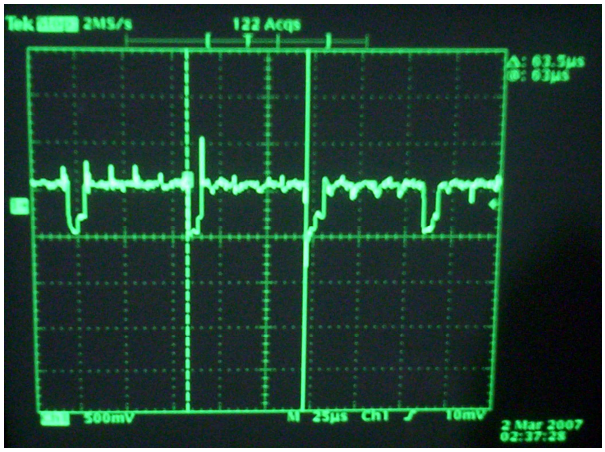


Figura 3.10. Imagen de la señal generada

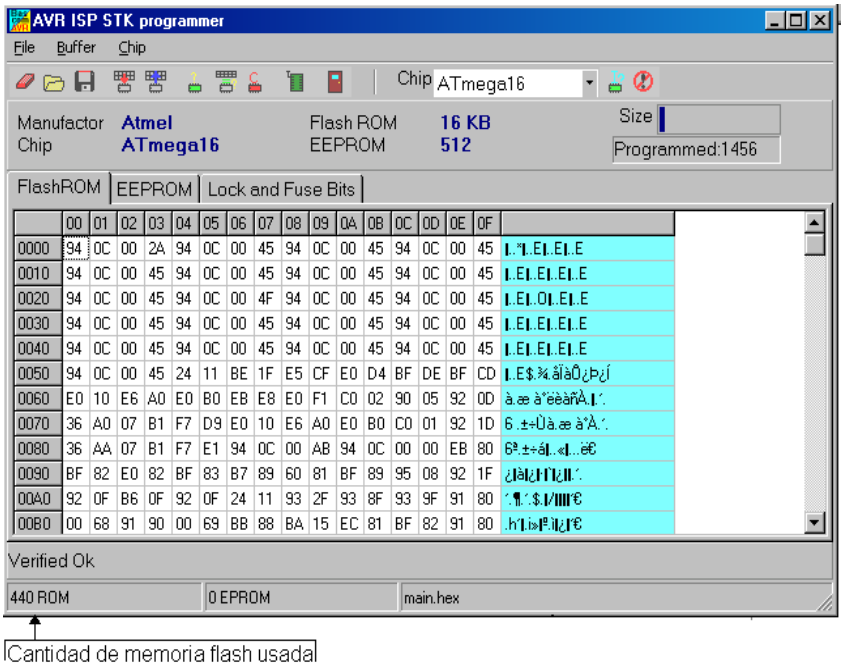


Figura 3.11. Pantalla del programador del Bascom AVR con el programa compilado en el Winavr

El otro punto a considerar era la cantidad de espacio que ocupaba el programa. En el caso de este compilador sucede que no hay manera de realizar la programación del AVR como se lo hacía con el otro compilador, por tal motivo lo que

se hace es obtener el archivo .HEX del programa compilado y de esta forma se utiliza la interface de programación del compilador Bascom AVR. El fruto de dicha operación se muestra en la Figura 3.11.

La diferencia en la cantidad de código también existe, como se puede notar en la imagen. En este punto se comparan los resultados obtenidos de esta prueba y se opta por el compilador Winavr para la realización de este prototipo, ya que los resultados de las señales obtenidos muestran el mejor desempeño de este compilador en esta aplicación.

3.4 IMPLEMENTACIÓN DEL SOFTWARE

Para comenzar la implementación del software se empezará indicando que después de realizar pruebas para generar la señal de video, ahora con la ayuda del *Timer0* se obtuvieron buenos resultados, pero de todas las mediciones realizadas se encontró que los valores de la cuenta hecha por el *Timer0* varían en décimas de microsegundo, que para esta aplicación puede constituir un problema.

La parte más sensible de este proyecto es la generación de caracteres que sean visibles en la pantalla de televisión, y de aquí partiremos para hacer todas las pruebas sobre la implementación del software.

Lo primero es tratar de idear una forma en la cual se puedan escribir los caracteres en la pantalla utilizando al máximo las facilidades que da el compilador en lenguaje C. Una de estas facilidades es la de poder calcular el tiempo que se necesita para realizar cierto proceso de acuerdo al número de instrucciones que se necesitan. El ciclo de máquina utilizando un cristal de 16 MHz es de 62.5 ns, con esto en mente se ha observado que en promedio cada instrucción en lenguaje C ocupa 4 ciclos de máquina.

3.4.1 Generación de los caracteres

El problema con la generación de los caracteres es que únicamente se puede controlar el tiempo ya que el haz de electrones tiene un patrón definido en su desplazamiento dentro de la pantalla del televisor. Por eso se trató, en primera instancia, de generar los caracteres creando una señal de 63 µs por medio de interrupciones que serían provocadas cada cierto tiempo fijo. La idea es dividir este tiempo de 63 µs, en 14 instantes de 4.5 µs, por medio de un contador se ubicará la zona en la cual deben leerse los datos a ser presentados y cuáles serían las secciones en las que hay que generar el sincronismo. En la Figura 3.12 se trata de explicar la idea de manera grafica.

Figura 3.12. Explicación gráfica de lo que se intenta hacer para generar los caracteres

Con esta idea en mente se procedió a realizar pruebas y se obtuvo que una señal de esta clase no tarda 63 µs, sino que tiene un valor de 510 µs. Analizando el problema y haciendo otras mediciones ya no con 14 divisiones sino reduciendo este valor y aumentando el tiempo en el que se ejecuta la interrupción, se obtuvo lo que se indica en la Figura 3.13.

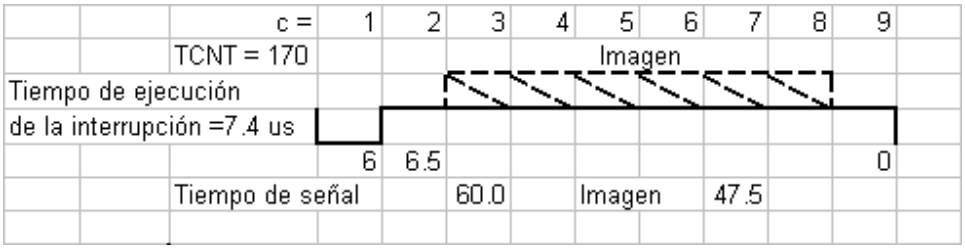


Figura 3.13. Gráfico de la señal obtenida variando los parámetros de la misma

Con estas pruebas, se vio que existen problemas cuando se hace que el tiempo del *Timer0* sea corto; no permite trabajar adecuadamente ya que los registros se sobrescriben y la pila del *stack*[7] se desborda. Esto hace que no se puedan controlar los procesos.

Para poder solucionar este problema se ideó otra forma de generar la señal y tener la posibilidad de procesar otras instrucciones sin tener que atender a la interrupción generada por el *Timer0* a cada instante, es la siguiente: lo que se hará es generar una cuenta con el *Timer0* de 64 µs, para que al cabo de ese tiempo se atienda la subrutina de la interrupción asociada al *Timer0*. Cuando se atienda esta subrutina se generará el sincronismo que sea necesario de acuerdo a un contador de líneas que también se encuentra ahí. Con esto se deja el espacio en el que se regresa a la rutina principal para ejecutar las tareas asignadas. Con este concepto se hicieron las pruebas y los resultados se muestran en la Figura 3.14.

En las imágenes mostradas se observa que la separación entre caracteres es grande y que solo se pueden observar ocho letras, que resulta una cantidad muy pequeña para poder desplegar cualquier mensaje. En cuanto a la señal que que generó estos caracteres, se puede apreciar en la Figura 3.15, una señal de 63.4 µs, que permite generar los caracteres mostrados, notando cómo se distingue dentro de la misma los niveles de voltaje que representan a los caracteres.

[7] Registro de almacenamiento de valores de las variables dentro del AVR.[29]

Figura 3.14. Imágenes de los caracteres generados

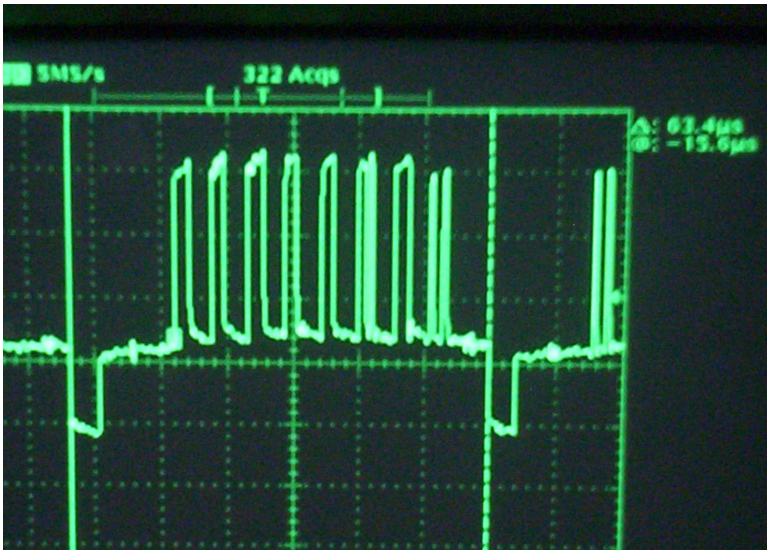


Figura 3.15. Imagen de la señal que genera los caracteres antes mostrados

Para poder reducir los espacios entre caracteres, lo que se hará es preparar los datos a ser publicados de una manera secuencial, a medida que avance la lectura de estos se genere la imagen de los caracteres a ser mostrados, ya que para realizar la lectura de los datos se ha leído un registro, en el cual se encuentran números que están asociados a una tabla con todos los caracteres que son localizados por medio de dichos números. Esta tabla contiene todos los valores necesarios para mostrar el carácter deseado. Una vez realizadas las modificaciones en el programa se obtiene el resultado mostrado en la Figura 3.16.

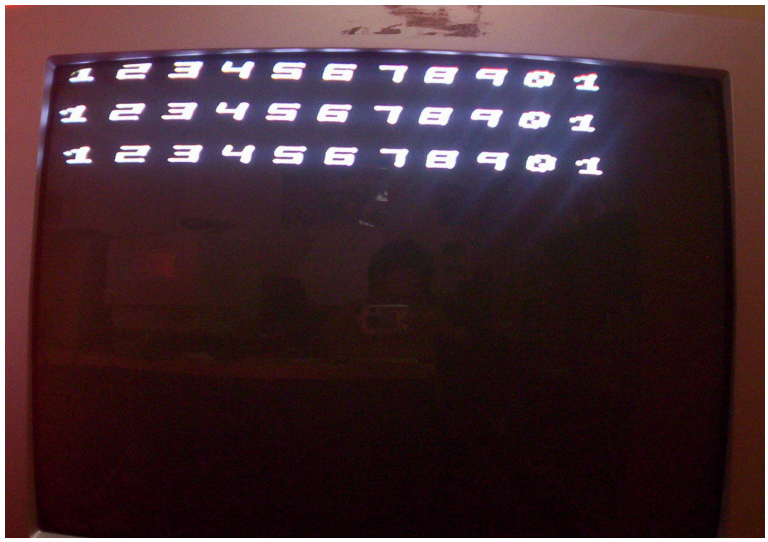


Figura 3.16. Imagen del programa modificado

Con todas las modificaciones realizadas hasta el momento, se han logrado presentar 11 caracteres visibles y un carácter que se encuentra oculto en el lado izquierdo de la pantalla; como se puede apreciar hay algunas líneas que pueden ser utilizadas con diferente información una vez que sea preparada. Hay otro espacio que se encuentra en el lado derecho de la pantalla, se intentó que se muestre un carácter más pero la imagen comenzó a vibrar, por lo que hay que hacer otras modificaciones y procurar que el texto se desplace para compensar la poca cantidad de caracteres visibles.

En las imágenes mostradas en la Figura 3.17 se puede apreciar el efecto de otros cambios en el programa y a su vez lograr desplazar los caracteres hacia la izquierda.

Una vez que se han realizado las pruebas con el texto, lo que falta por probar es que esté se pueda observar sobre una imagen de fondo. Dicha imagen proveerá el sincronismo que será leído por el programa y de esta forma se sobrepondrá la señal que contiene los caracteres en la imagen, sin que el conjunto se distorsione.

Figura 3.17. Imagen que muestra como se desplaza el texto sobre la pantalla

Para lograr este efecto se han hecho varias pruebas que se muestran en las Figura 3.18, en las que se ve que el texto tiene un error en el ajuste del sincronismo vertical.

Figura 3.18. Problemas con el sincronismo vertical

En la Figura 3.19 se puede apreciar cómo los caracteres vibran en el sentido horizontal, teniendo que corregir el sincronismo horizontal.

Figura 3.19. Problemas con el sincronismo horizontal

En la Figura 3.20 se puede ver cómo la señal con los sincronismos corregidos se muestra y se aprecia su contenido.

Figura 3.20. Imágenes con los caracteres sobrepuestos en el video

Con todas estas pruebas realizadas, se puede observar que los caracteres que se muestran en la pantalla tienen una definición aceptable y que se puede apreciar un mensaje sobre dicha imagen.

Lo que falta por probar es la recepción de los datos por el GPS y su publicación en la pantalla de televisión.

3.4.2 Procesamiento de los datos del GPS

En esta parte del programa se da el tratamiento adecuado a los datos recibidos del GPS para poder ubicar la información que será útil para cumplir con los propósitos de este prototipo.

El receptor GPS envía 4 tipos de tramas de datos, de las cuales nos interesa la trama RMC que entrega la información básica sobre la posición, velocidad y tiempo. Para explicar el procesamiento de esta trama se ilustra con un ejemplo. La trama que interesa procesar es similar a:

*$GPRMC,160644,A,0011.7021,S,07826.1680,W,022.4,084.4,130205,,,A*7B*. [8]

Se inicia el análisis de la trama de la siguiente forma:

- *$GPRMC*, Se comprueban estos caracteres para empezar el análisis de la información.
- *160644* , La información de este campo es la siguiente:
  - *16:* representa la hora en el tiempo universal coordinado y para obtener la hora local se debe restar 5 que es la diferencia entre la hora local y la hora del tiempo universal coordinado,
  - *06:* representa los minutos
  - *44:* representa los segundos
- ,*A*, Si el caracter es A indica que existen los datos de la posición y velocidad en la trama, caso contrario si se encontrara una ,*V*, indica que los datos de longitud y latitud no son fiables.
- *0011.7021*,*S*, Aquí se encuentran los datos de la latitud donde:

[8] Las comas indican la separación entre los distintos campos de datos de la trama.

  - 00 representa los grados y *11.7021* representa los minutos.
  - *S*, indica el hemisferio sea norte, N, o sur, S.
- *07826.1680,W*, Aquí se encuentran los datos de la longitud donde:
  - *078* representa los grados y *26.1680* representa los minutos.
  - *W*, indica el hemisferio sea occidental, W, u oriental, O.
- ,*022.4*, Velocidad en nudos que para obtener la velocidad en Km/h se debe multiplicar por 1.8.
- *084.4*, Angulo de elevación en grados.
- *130205* Aquí se encuentra la información de la fecha donde:
  - *13* representa el día
  - *02* representa el mes
  - *05* representa el año
- , *003.1* Valor de la variación magnética
- ,*A*7B* Valor usado para la comprobación de errores.

De lo anteriormente visto se utilizarán los campos que tienen la hora, la posición en latitud y longitud y el valor de la velocidad. En el AVR se harán las operaciones necesarias para lograr obtener la información que será mostrada en la pantalla de televisión.

En la Figura 3.21, se muestran las tramas que cada segundo envía el GPS, para procesarlos de tal manera que pueda elegir la trama RMC de entre las que llegan para obtener los datos.

En caso de que no existan datos para actualizar la información, que en esta prueba será la hora, se seguirá mostrando la hora anterior.

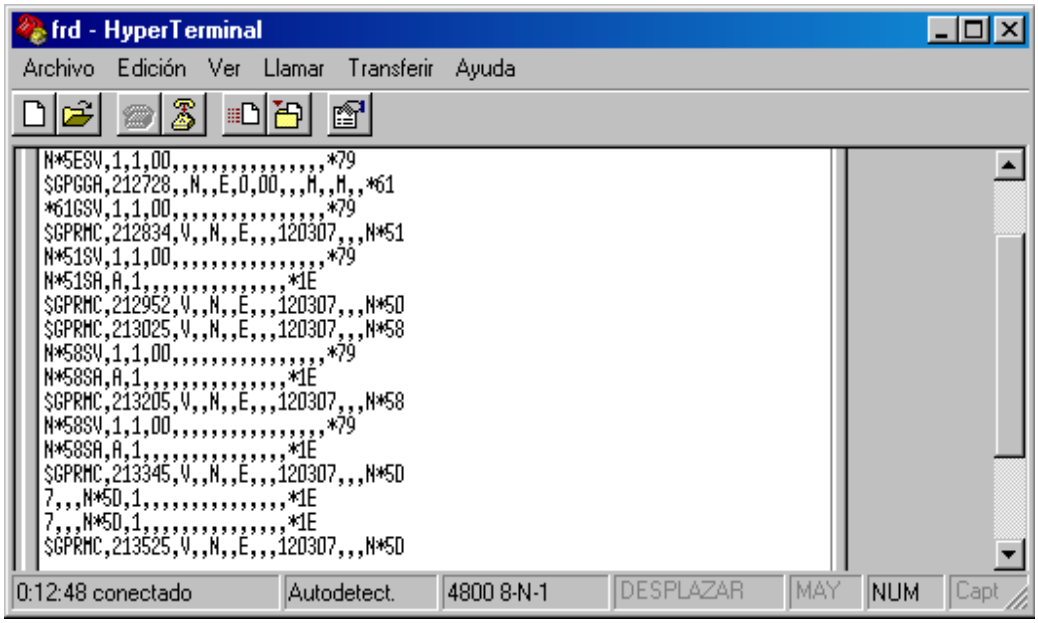


Figura 3.21. Pantalla del Hyperterminal con las tramas recibidas y procesadas del GPS

Al realizar las pruebas para visualizar la información de la hora, tomada de los datos recibidos del GPS, se tiene la Figura 3.22 que muestra la hora que es actualizada cada minuto la misma que aparece en la pantalla del televisor.

Figura 3.22. Actualización de la hora en la pantalla de televisión

Al comparar las tramas recibidas y la hora publicada se puede notar que hay tramas que llegan pero que son eliminadas, debido a que el microprocesador las descarta, pues no es la que busca.

Se han probado, con todas estas pruebas, los elementos que forman el hardware y el software para este prototipo. Una prueba en conjunto se hará en siguiente capítulo.

CAPÍTULO 4. PRUEBAS, RESULTADOS Y COSTOS DEL PROTOTIPO

4.1 INTRODUCCIÓN

El objetivo de este capítulo es presentar un ejemplo de cómo este prototipo funciona con todos los elementos antes descritos sobre una ruta previamente definida.

Una vez que se han realizado las pruebas y se han evaluado los resultados, se procederá a realizar un cálculo de los costos del prototipo.

4.2 PRUEBA DEL PROTOTIPO

Para realizar esta prueba se ha seleccionado la ruta Estación de la Ecovía Río Coca el Quinche, en la cual se han seleccionado nueve puntos de interés incluidos el origen y el destino que indican las poblaciones más grandes ubicadas a lo largo de este trayecto. Estos puntos de interés son:

- Estación Río Coca, inicio de la ruta.
- Cumbayá
- Tumbaco
- El Arenal
- Pifo
- Tababela (Sector del nuevo Aeropuerto)
- Yaruquí
- Checa
- Quinche, fin de la ruta.

4.2.1 Datos de la ruta

En cada uno de los nueve puntos seleccionados es necesario conocer su ubicación geográfica en base a su latitud y longitud. Para esto se utiliza el Google Earth como la herramienta que posibilita esta tarea. En la Figura 4.1 se muestra la imagen satelital de los puntos de la ruta seleccionada.

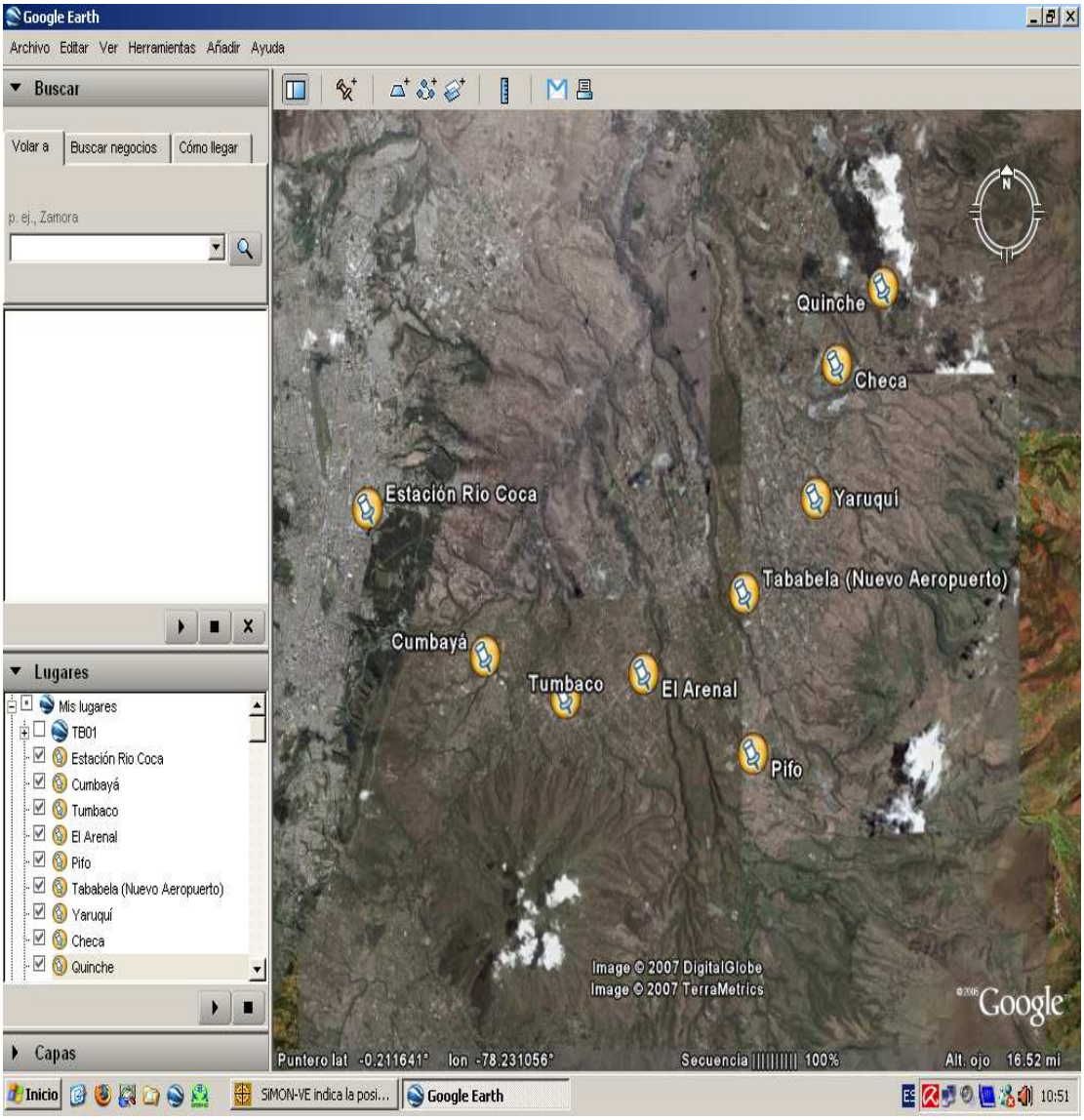


Figura 4.1. Vista satelital de la ruta seleccionada para el ejemplo.

En la Tabla 4.1 se muestra cada uno de los puntos seleccionados con sus valores de latitud y longitud.

Etiqueta	Punto	Latitud	Longitud
P1	Estación Río Coca	0.163814° S	78.471534°W
P2	Cumbayá	0.200608° S	78.431273°W
P3	Tumbaco	0.211004° S	78.403756°W
P4	El Arenal	0.204945° S	78.377032°W
P5	Pifo	0.224848° S	78,338944°W
P6	Tababela	0.184955° S	78.342638°W
P7	Yaruquí	0.161276° S	78.318143°W
P8	Checa	0.128142° S	78.310976°W
P9	Quinche	0.108942° S	78.295148°W

Tabla 4.1. Valores de Latitud y Longitud en los puntos de interés de la ruta

4.2.2 Cálculo del área en cada punto

Como se mencionó antes, a partir de cada punto de interés seleccionado, en la ruta se va a calcular un área dentro de la cual el o los mensajes asociados a cada posición son válidos. Para esto se considera lo que se dijo anteriormente, que 1' representa 1840 m, con lo cual se establece un área de 3,385,600 m^2 como área válida para los mensajes asociados.

Distancia	Valor (Km.)
P1 - P2	6.07
P2 - P3	3.25
P3 - P4	3.03
P4 - P5	4.74
P5 - P6	4.39
P6 - P7	3.82
P7 - P8	3.78
P8 - P9	2.73
Total	31.81

Tabla 4.2. Distancias entre los distintos puntos de la ruta

Al medir las distancias entre los puntos seleccionados para la ruta, que se muestra en la Tabla 4.2[9], se puede observar que las distancias entre algunos puntos provocarán que exista un error al momento de presentar los datos, ya que si se considera el área alrededor de cada punto de 1’ de radio, se produciría una superposición de áreas como se ilustra en la Figura 4.2, por lo que para aplicar este criterio se necesitaría una distancia mínima entre los puntos de 2’ que equivaldría a 3.68 Km.

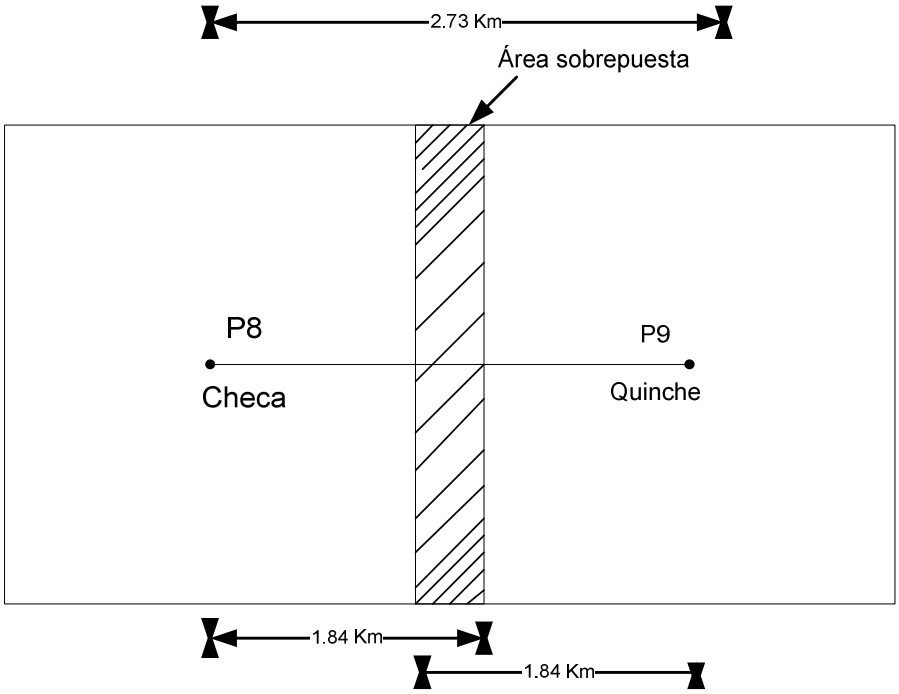


Figura 4.2. Superposición de áreas

Por esta razón se calculará el área a partir de 0.5’ lo que representa 920 m, que define un área alrededor de 846,400 m^2, permitiendo así que no exista superpoción de las áreas como se muestra en la Figura 4.3.

[9] Datos tomados del Google Earth

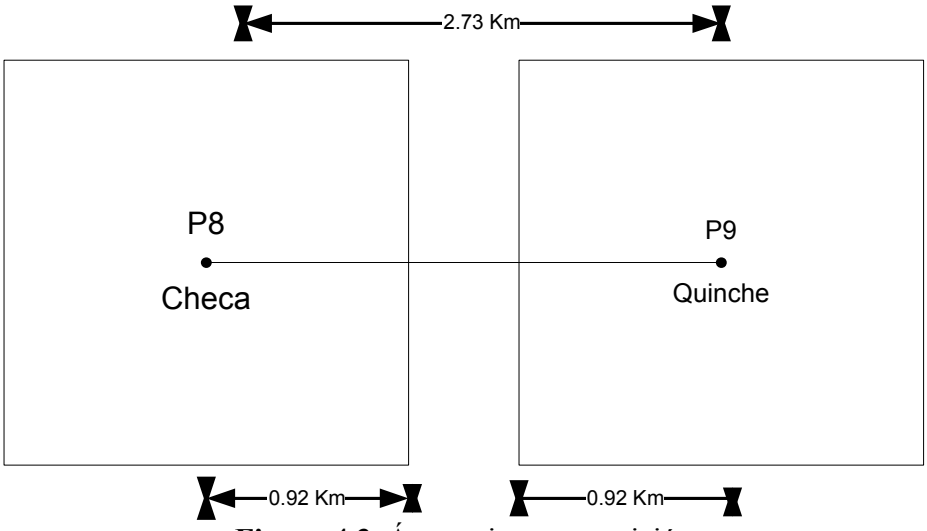


Figura 4.3. Áreas sin superposición

4.2.3 Tratamiento de los datos

Para seleccionar un grupo de mensajes a ser presentados, se necesita realizar el proceso antes descrito en los datos de la posición. Como se puede observar en la Tabla 4.1, los valores que se encuentran allí no tienen alguna forma de variación que a simple vista permita realizar los cálculos necesarios. Los datos se muestran en la Tabla 4.3 con otro formato para apreciarlos mejor.

Etiqueta	Punto	Latitud	Longitud
P1	Estación Río Coca	0°9'49.73" S	78°28'17.52" W
P2	Cumbayá	0°12'2.19" S	78°25'52.58" W
P3	Tumbaco	0°12'39.61" S	78°24'13.52" W
P4	El Arenal	0°12'17.8" S	78°22'38.35" W
P5	Pifo	0°13'39.45" S	78°20'20.2" W
P6	Tababela	0°11'5.54" S	78°20'33.5" W
P7	Yaruquí	0°9'40.59" S	78°19'5.31" W
P8	Checa	0°7'41.31" S	78°18'39.51" W
P9	Quinche	0°6'32.19" S	78°17'42.53" W

Tabla 4.3. Datos de Latitud y Longitud presentados en otro formato

Partiendo de los datos de la Tabla 4.3 se presenta un ejemplo matemático mediante el cual se describe cómo se realiza la validación de los datos para las áreas asociadas a cada punto.

El punto P9 (Quinche) tiene como latitud 0℃'32,19'' S y como longitud 78℃17'42,53'' W. En esta forma de presentar los dat os es evidente que si se produce una variación de 0.5'. Los valores de cada coordenada cambiarían entre un mínimo y un máximo. Estos valores se encuentran en la Tabla 4.4.

P9 El Quinche			
Latitud máxima	0℃'2.19'' S	Longitud máxima	78℃17'2 8.53'' W
Latitud del punto	0℃'32.19'' S	Longitud del punto	78℃17'42.53'' W
Latitud mínima	0℃'2.19'' S	Longitud mínima	78℃17'5 6.53'' W

Tabla 4.4. Variación de la Latitud y Longitud para el punto P9

Para encontrar el valor que representa esta variación en el formato que utiliza el receptor, se hará la transformación, luego se realizará la diferencia entre dos valores, así se sabrá cuál es el valor, que permitirá reconocer un dato como aceptable dentro del área determinada para la latitud y la longitud. Esto se puede apreciar en la Tabla 4.5.

P9 El Quinche			
Latitud máxima	0.100608666°S	Longitud máxima	78.29813611°W
Latitud del punto	0.108942°S	Longitud del punto	78 .295148°W
Latitud mínima	0.11695333°S	Longitud mínima	78.2920583°W
Diferencia entre latitudes	0.00803	Diferencia entre longitudes	0.00309

Tabla 4.5. Deferencia entre Latitudes y Longitudes

Estos valores serán los que se utilicen al momento de realizar una diferencia, entre el punto recibido de latitud y el almacenado, para saber si una latitud está dentro del área establecida, esperando que el resultado sea menor o igual al valor de comparación. Lo mismo se hará con la longitud, y si estas dos comparaciones son

afirmativas, el punto recibido es considerado dentro del área al rededor del punto de interés y se procederá a mostrar los mensajes grabados para dicha posición.

4.2.4 Mensajes a mostrar

Una vez realizado el proceso descrito anteriormente se procede a preparar los mensajes a ser mostrados. Se presentarán 3 mensajes; para el punto 7 por ejemplo se verían así:

- Mensaje 1: Estamos en Yaruquí
- Mensaje 2: Viajamos a 50 Km/h
- Mensaje 3: Llegaremos al Quinche en aprox. 30 min.

De los mensajes indicados, arriba se tiene que las palabras que están subrayadas son las que irán cambiando a medida que el vehiculo se desplace por la ruta seleccionada. En cambio los otros caracteres del mensaje son parte del mensaje almacenado para cada posición.

Con toda esta información lista y almacenada se realiza la prueba del prototipo.

4.2.5 Condiciones de la Prueba

Para esta prueba lo que se hizo es simular el ingreso de datos, esto quiere decir que en lugar de recibir los datos del recetor GPS se envían datos preparados para que sen recibidos por el AVR. Estos datos preparados tienen la misma forma que los datos que enviaría el GPS, es decir tienen 4 tramas de datos de las cuales hay que escoger una que es la trama RMC, ésta contiene toda la información que se requiere procesar.

Adicionalmente a las fotografías se adjuntan las pantallas capturadas de la comunicación hecha entre la computadora y el AVR.

4.3 RESULTADOS

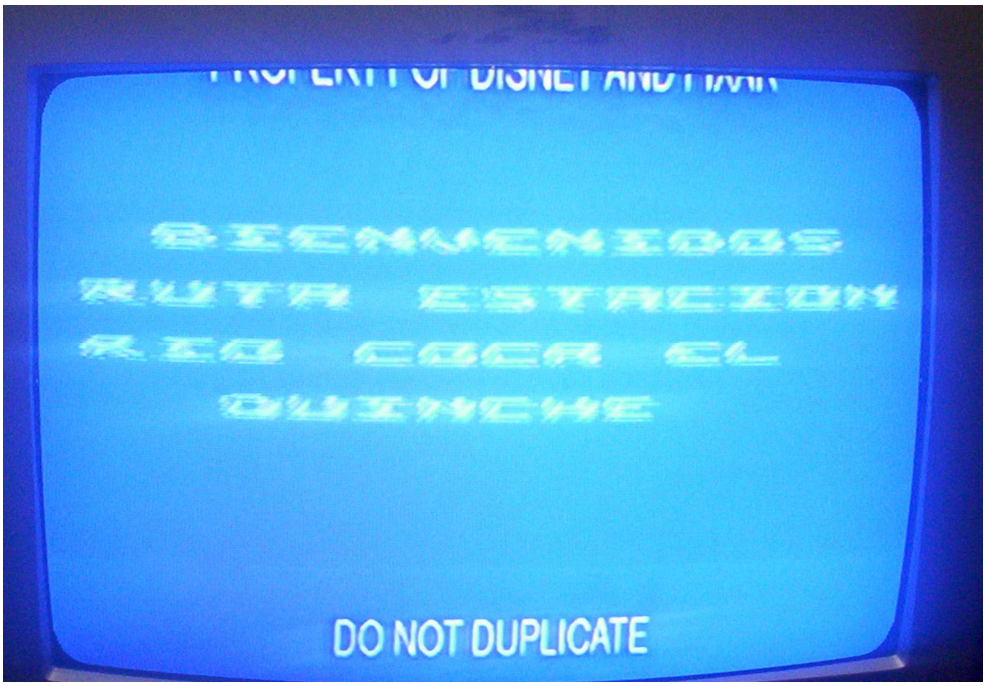


Figura 4.4. Mensaje de bienvenida a los usuarios de la unidad

En la Figura 4.4 se muestra la pantalla que da la bienvenida a los usuarios de la unidad.

Los datos preparados para cada punto se muestran a continuación, los datos que deben ser procesados se encuentran subrayados.

- Para un punto donde no existen datos, como se muestra en la Figura 4.5
  - $GPGGA,160425,,N,,E,0,00,,,M,,M,,*69
  - $GPGSA,A,1,,,,,,,,,,,,,,*1E
  - $GPGSV,1,1,00,,,,,,,,,,,,,,*79
  - $GPRMC,160425,V,,N,,E,,,130205,,,N*59

- Para un punto donde no existen datos que correspondan a algún punto de la ruta, se tiene el resultado en la Figura 4.6.
    - $GPGGA,160642,0011.7048,S,07826.1694,W,1,04,03.6,02415.4,M,017.9,M,,*48
    - $GPGSA,A,2,24,28,07,20,,,,,,,,,03.8,03.6,01.0*05
    - $GPGSV,2,1,07,04,58,268,39,07,23,343,49,17,05,221,00,20,20,060,51*7D
    - $GPRMC,180650,A,00.168814,S,078.479534,W,000.0,000.0,130205,,,A*72

- Para el punto P9 se coloca el límite máximo
    - $GPGGA,160642,0011.7048,S,07826.1694,W,1,04,03.6,02415.4,M,017.9,M,,*48
    - $GPGSA,A,2,24,28,07,20,,,,,,,,,03.8,03.6,01.0*05
    - $GPGSV,2,1,07,04,58,268,39,07,23,343,49,17,05,221,00,20,20,060,51*7D
    - $GPRMC,180650,A,00.100608,S,078.298136,W,000.0,000.0,130205,,,A*72

- Para el punto P9 se coloca el limite mínimo
    - $GPGGA,160642,0011.7048,S,07826.1694,W,1,04,03.6,02415.4,M,017.9,M,,*48
    - $GPGSA,A,2,24,28,07,20,,,,,,,,,03.8,03.6,01.0*05
    - $GPGSV,2,1,07,04,58,268,39,07,23,343,49,17,05,221,00,20,20,060,51*7D
    - $GPRMC,192000,A,00.116953,S,078.292205,W,000.0,000.0,130506,,,A*72

- Para el punto P7 que indica el punto que pertenece a Yaruquí

- $GPGGA,160642,0011.7048,S,07826.1694,W,1,04,03.6,02415.4,M,017.9,M,,*48
- $GPGSA,A,2,24,28,07,20,,,,,,,,,03.8,03.6,01.0*05
- $GPGSV,2,1,07,04,58,268,39,07,23,343,49,17,05,221,00,20,20,060,51*7D
- $GPRMC,180650,A,00.161276,S,078.18143,W,000.0,000.0,130205,,,A*72

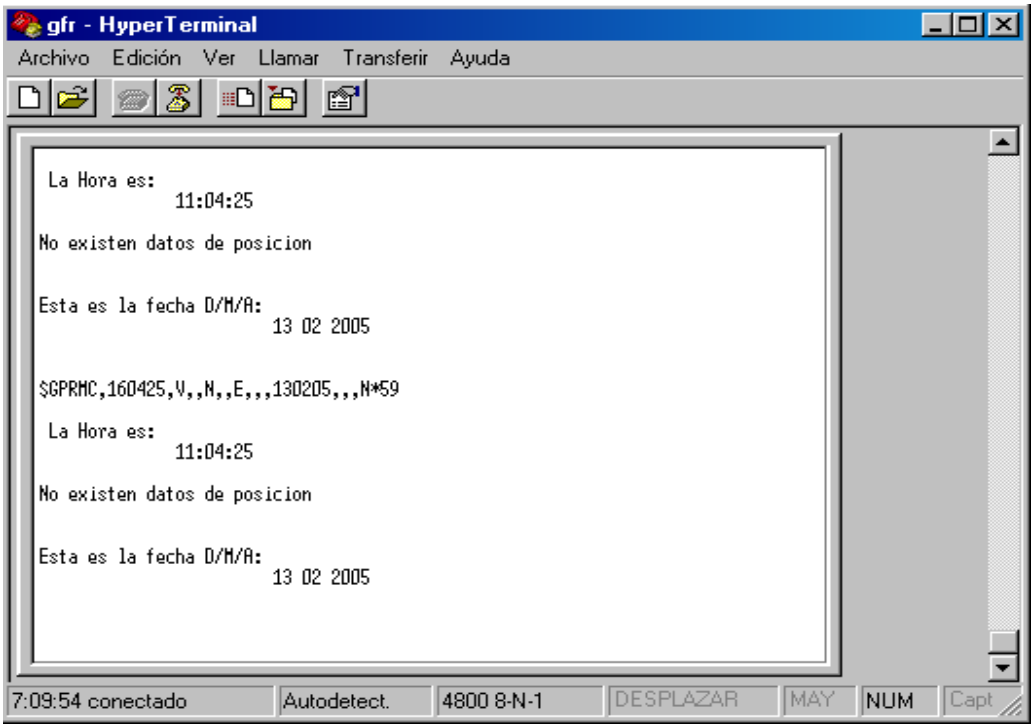


Figura 4.5. Pantalla que muestra que en la trama recibida no ha datos de posición.

En la Figura 4.7 se muestra la pantalla capturada para el valor límite superior del punto P9 que se indica en la Tabla 4.5. En cambio en la Figura 4.8 se muestra la pantalla capturada para el límite inferior del punto P9.

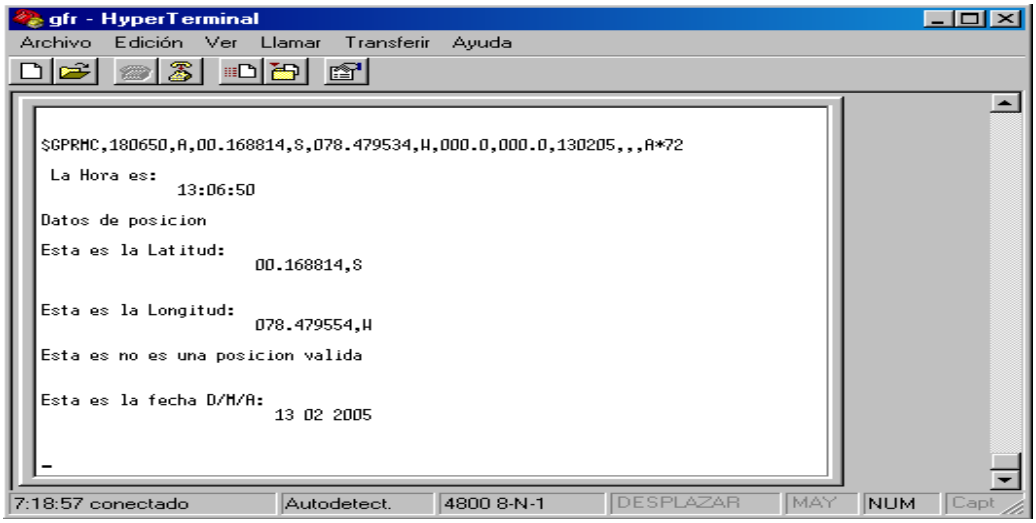


Figura 4.6. Pantalla que muestra un punto con datos que no pertenecen a la ruta.

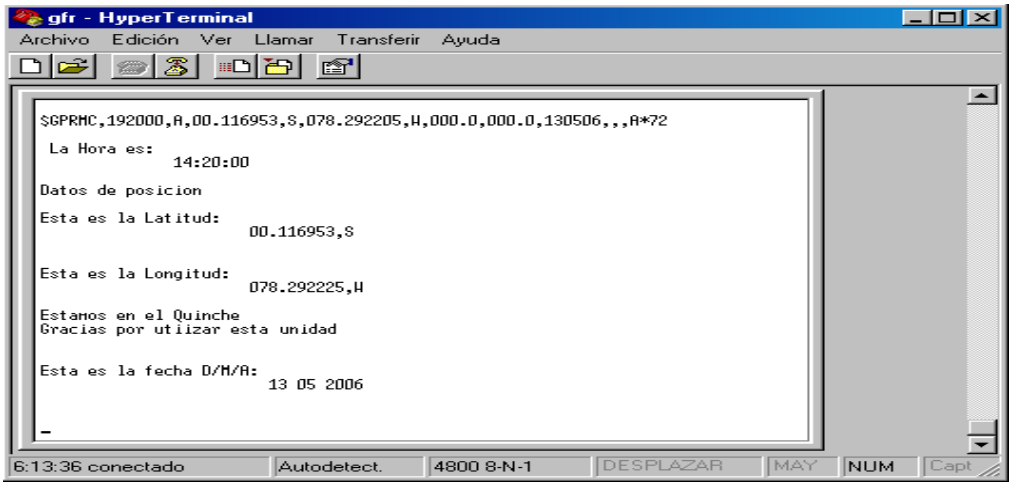


Figura 4.7. Pantalla que muestra la localización del punto P9 para el punto máximo

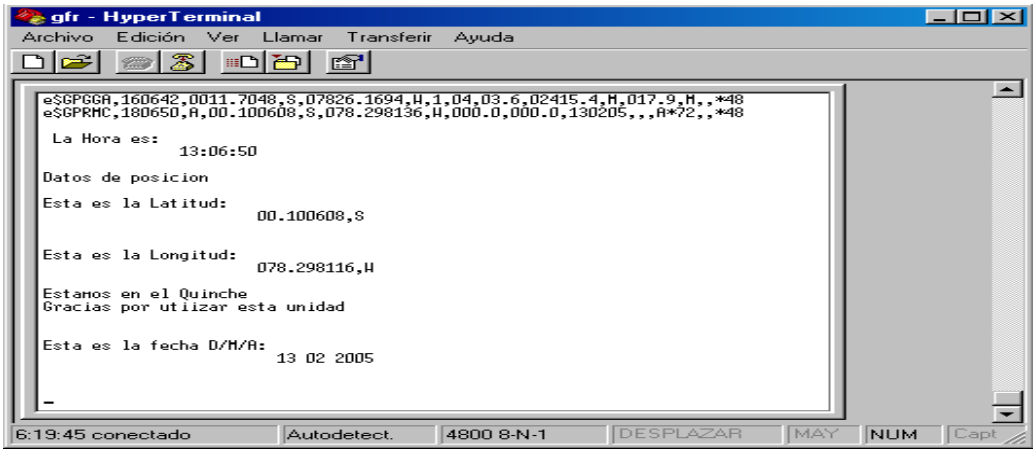


Figura 4.8. Pantalla que muestra la localización del punto P9 para el punto mínimo

Ahora se escoge el punto P7 que corresponde a Yaruquí, para poder observar los mensajes que se muestran en la pantalla del televisor. En la Figura 4.9 se muestra la pantalla capturada de la posición P7.

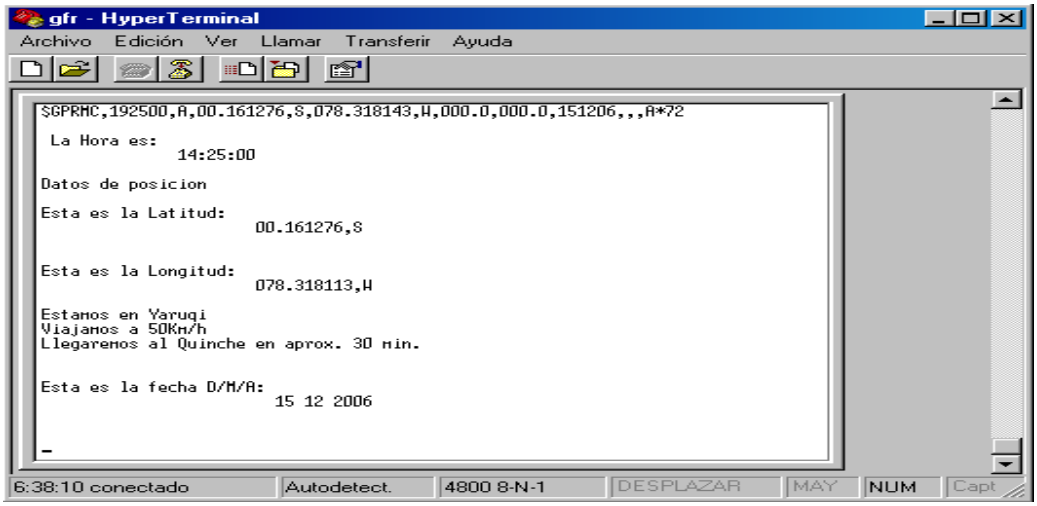


Figura 4.9. Pantalla que muestra la localización del punto P7

Figura 4.10. Pantalla que muestra el primer mensaje mostrado en P7

Ahora se presentaran imágenes de los mensajes desplegados en la pantalla de televisión. En las Figura 4.10, 4.11, 4.12.

Figura 4.11. Pantalla que muestra el segundo mensaje mostrado en P7

Figura 4.12 Pantalla que muestra el tercer mensaje mostrado en P7

Para concluir con estas pruebas indicaremos el mensaje que aparece cuando la unidad llega a su destino final en la Figura 4.13.

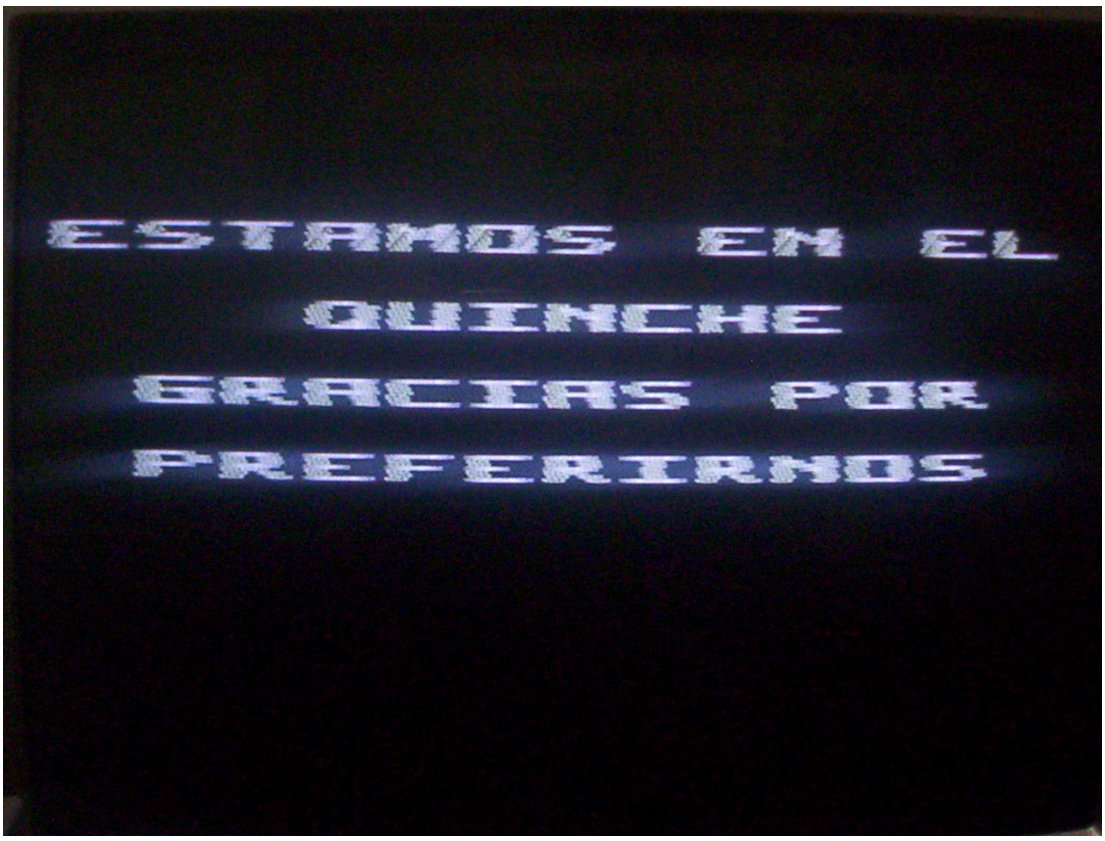


Figura 4.12 Pantalla que muestra el mensaje al final del viaje

4.4 COSTOS

Los costos para este prototipo se dividen en dos grupos: primero corresponde a los elementos utilizados para la implementación del hardware, el segundo contiene el detalle de la mano de obra en la que se indica el tiempo empleado en cada parte del proceso.

En la Tabla 4.6 se detallan los elementos ocupados para la implementación del hardware.

MATERIALES				
Descripción	Cantidad	Detalle	Precio unidad	Total
Polarized Capacitor (Radial)	8	10uF	$ 0.14	$ 1.12
Capacitor	10	0.1uF	$ 0.06	$ 0.60
Capacitor	2	33pF	$ 0.10	$ 0.20
Polarized Capacitor (Radial)	1	1uF	$ 0.10	$ 0.10
Capacitor	4	100nF	$ 0.10	$ 0.40
Polarized Capacitor (Radial)	1	1000uF	$ 0.15	$ 0.15
Default Diode	5		$ 0.10	$ 0.50
Typical INFRARED GaAs LED	4		$ 0.10	$ 0.40
Header, 25-Pin	2		$ 1.50	$ 3.00
Header, 16-Pin	1		$ 1.50	$ 1.50
Header, 20-Pin, Dual row	1		$ 1.50	$ 1.50
Inductor	1	100mH	$ 0.20	$ 0.20
NPN , PNP	2		$ 0.20	$ 0.40
Potentiometer	1	5K	$ 1.00	$ 1.00
Resistor	6	10K	$ 0.03	$ 0.18
Resistor	2	1K	$ 0.03	$ 0.06
Resistor	1	15	$ 0.03	$ 0.03
Resistor	3	4.7K	$ 0.03	$ 0.09
Resistor	1	47K	$ 0.03	$ 0.03
Resistor	1	330	$ 0.03	$ 0.03
Potentiometer	1	1K	$ 0.40	$ 0.40
Resistor	1	100K	$ 0.03	$ 0.03
Switch	1		$ 1.00	$ 1.00
Hex Inverter Schmitt Trigger	1		$ 0.80	$ 0.80
ATMEGA 16	1		$ 9.90	$ 9.90
Max232	1		$ 3.00	$ 3.00
LM1881	1		$ 4.50	$ 4.50
Regulator Voltage	1	7808	$ 0.60	$ 0.60
Regulator Voltage	1	7805	$ 0.60	$ 0.60
Hex Schmitt-Trigger Inverter	1		$ 0.60	$ 0.60
Socket	4		$ 0.30	$ 1.20
Crystal Oscillator	1		$ 0.90	$ 0.90
Receptor GPS	1		$ 80.00	$ 80.00
Placa del prototipo	1		$ 35.00	$ 35.00
Caja plástica para el equipo	1		$ 6.50	$ 6.50
ZOCALO 40 pines	1		$ 0.40	$ 0.40
ZOCALO 28 pines	1		$ 0.40	$ 0.40
ZOCALO 16 pines	4		$0.40	$ 1.60
			Total	$ 158.92

Tabla 4.6. Costos de los elementos utilizados para implementar el prototipo

En la Tabla 4.7 se indican todos los rubros relacionados con la mano de obra que se utilizó para este proyecto.

PRECIO DESCOMPUESTO			
MANO DE OBRA			
Horas	Detalle	Precio por hora	Total
320	Investigación	5.00	1,600.00
25	Diseño del Hardware	5.00	125.00
15	Implementación del Hardware	5.00	75.00
30	Diseño del Software	5.00	150.00
200	Implementación del Software	5.00	1,000.00
150	Pruebas	5.00	750.00
120	Redacción de la tesis	5.00	600.00
TOTAL			$ 4,300.00

Tabla 4.7. Costos de la mano de obra para implementar el prototipo

Considerando lo anterior, se tiene como valor total de la implementación de este prototipo la cantidad de cuatro mil cuatrocientos cincuenta y dos con cuarenta y dos centavos, descrito en la Tabla 4.8.

VALOR TOTAL DEL PROTOTIPO	
Detalle	**Total**
Materiales	$ 158.92
Mano de obra	$ 4,300.00
TOTAL	**$ 4,452.42**

Tabla 4.8. Costo total del prototipo

En el cálculo del tiempo se consideró una jornada de 8 horas diarias y en base a este criterio se tiene que un total de 108 días laborables, en los cuales se desarrolló este proyecto.

Figura 4.13. Vista frontal del equipo terminado

Figura 4.14. Vista lateral del equipo terminado.

CAPÍTULO 5. CONCLUSIONES Y RECOMENDACIONES

5.1 CONCLUSIONES

- Los AVR's abren nuevas posibilidades en el trabajo con microcontroladores por sus múltiples capacidades y por ser económicamente accesibles para el desarrollo de grandes proyectos.

- La capacidad de almacenamiento también es un aspecto importante a considerar, ya que existen AVR's con capacidades de almacenamiento desde 2Kbytes hasta 128Kbytes en memoria flash, de 128 bytes hasta 4Kbytes en memoria EEPROM, que pueden ser utilizados separadamente o en conjunto de acuerdo a las necesidades.

- El otro aspecto interesante que se puede destacar de los AVR es su versatilidad con las velocidades a las que puede trabajar que van desde 1MHz hasta 16 MHz y adicionalmente la capacidad de generar algunas de estas velocidades con su cristal interno como son las velocidades de 1, 4, 8Mhz.

- Algo muy importante de destacar es la posibilidad de la programación por sistema de los AVR, ya que evita dos cosas fastidiosas al momento de trabajar con microcontroladores, la primera el tener que comprar equipos costosos o armar programadores que ocupan gran cantidad de espacio y lo segundo, es la imposibilidad de hacer cambios instantáneos, ya que no se puede integrar el programador al circuito que se está desarrollando sin perjuicio del mismo.

- Como parte de esto mismo, la fabricación de este programador también representa una gran ventaja ya que tan solo se necesita de un circuito integrado, el 74244 que es un buffer, además un diodo una resistencia y un

condensador que pueden ser montados sobre un conector DB-25 para ser utilizado en el puerto paralelo de la computadora. Este montaje es sumamente pequeño y realmente portátil, de ahí las salidas de este programador se conectan a los pines indicados del AVR y se puede realizar la programación.

- Junto con un programador tan versátil, debe haber una buena capacidad en los espacios de memoria para ser grabados y borrados por muchas ocasiones y los AVR la tienen, según las especificaciones se pueden tener desde 1000 hasta 10,000 ciclos de escritura y borrado, en la práctica se comprueba que es verdad.

- En cuanto a los compiladores que existen para el AVR solo se puede hablar de dos: el BascomAVR y el WInavr, de los que se puede comentar lo siguiente:
    - Bascom AVR, es un compilador diseñado para que el programador se concentre en el problema que quiere resolver, pues tiene muchas subrutinas elaboradas y probadas que permiten manejar dispositivos periféricos con una facilidad impresionante, con abundante información para entender su funcionamiento y gran cantidad de ejemplos desarrollados, con todo y estas ventajas no fue utilizado en este proyecto ya que el costo de todas estas facilidades es el tiempo de procesamiento, que en el caso de aplicaciones como la del presente trabajo es crítico.
    - En el caso del Winavr, es que es un compilador hecho para controlarlo todo y esto implica trabajar con lenguaje C. Este compilador tiene poca información disponible, pero de a poco va aumentando a medida que es más difundida su utilización, tanto en sus ayudas como en otros documentos, y la que existe no necesariamente está en inglés. El programador en este lenguaje realmente se ve limitado ya que el conjunto de instrucciones que posee es elemental y en cuanto a ejemplos implementados se tienen muy pocos. La ventaja de este

compilador es que se llega a tener un control bastante claro del tiempo y de las acciones que ejecuta.

- La razón del uso del compilador Winavr se debe a que en promedio una instrucción realizada en este compilador se toma cuatro ciclos de máquina, con lo cual se puede realmente controlar el tiempo de ejecución del programa implementado. Además evita el engorroso proceso de trabajar con lenguaje Assembler, pero con el mismo control que se tendría en este lenguaje.

- La generación de caracteres es un proceso que requiere de una gran precisión y alto desempeño por lo cual microcontroladores como los AVR's muestran sus bondades al momento de realizar esta clase de proyectos.

- El buscar la forma más eficiente de generar los caracteres demostró, la importancia del tiempo en esta aplicación, por lo que se optó por suspender todo proceso en el instante de generar los caracteres, ya que cualquier otra tarea realizada simultáneamente afecta la imagen generada.

- El problema de la sincronización de las señales mostradas en la pantalla del televisor representa una parte importante, por lo que el circuito integrado LM1881 constituye una herramienta importante para que la señal de video y la señal generada se sumen y no se distorsione la imagen final.

- La tabla de caracteres que se creó para esta aplicación se característica por lo siguiente:
    - Son caracteres de 7 x 5 píxeles.
    - Están guardados como una imagen reflejada ya que al instante de ser presentados se produce un desplazamiento hacia la derecha que los mostrará adecuadamente.

- La mayoría de los caracteres utilizados son letras mayúsculas y unas pocas letras minúsculas utilizadas para abreviaturas, algunos símbolos adicionales y los dígitos del 0 al 9, como se puede apreciar en el Anexo 5.

- La información que provee el sistema GPS hace posible la implementación de aplicaciones como ésta a un costo razonable, en cuanto a equipos que permiten obtener esta información. Lo que abre un campo muy amplio para el desarrollo de productos o servicios en distintas áreas.

- La información de la posición recibida es la base del funcionamiento de este prototipo, ya que en base a ella se puede considerar si una determinada lectura recibida del receptor GPS está o no dentro del área considerada como válida para el punto de interés almacenado.

- El dato de la velocidad, para este proyecto es tan solo información adicional para los usuarios ya que no se utiliza en ningún cálculo. Por lo tanto el tiempo aproximado que se presenta como información, es un tiempo almacenado y que está estimado conforme a los tiempos que normalmente se utilizan para llegar al destino desde cada uno de los puntos de interés.

- Cuando se analiza el costo de este proyecto se observan cantidades que distan mucho de ser de “bajo costo”, pero estamos hablando de un prototipo que sujeto a depuración y producción en serie se pueden ser abaratado.

- En cuanto al tamaño del equipo, como se puede apreciar en las fotografías al final del capitulo 4, es bastante grande pero se puede reducir considerando que la tarjeta de desarrollo sobre la cual se implementó el prototipo tiene funciones que no son utilizadas, por lo tanto es posible reducir el tamaño del equipo final.

5.2 RECOMENDACIONES

- Se debe considerar la posibilidad de realizar un trabajo similar teniendo en cuenta la alternativa de generar los caracteres a color, de diferentes tamaños, utilizando las facilidades de circuitos integrados que son capaces de generar caracteres para televisión, teniendo en cuenta aplicaciones más amplias que resultarían económicamente más asequibles.

- Se recomienda que se investigue con personas que trabajan en el área de transporte, para conocer cuáles serían las necesidades a fin de que los nuevos profesionales definan soluciones prácticas y realizables.

- Por cuanto se asumió que la información presentada es de utilidad para todos los usuarios, en la práctica debe verificarse de acuerdo al medio en el cual se trata de aplicar, ello se recomienda investigar cómo mejorar la idea presentada.

- Otro factor a considerar es el criterio de los usuarios, de qué tan útil les parecería la implementación de esta clase de servicio en las unidades de transporte y qué tipo de información les seria de provecho, para que de esta forma se pueda llegar a suplir esta necesidad de información.

- Para cualquier persona que intenta desarrollar proyectos donde los recursos o las herramientas a ocuparse no son conocidas, se recomienda buscar la forma de sistematizar la información que obtenga acerca de cada parte que esté investigando, que tome todo el tiempo necesario para ver los beneficios y desventajas que se le presenten, para que de esta forma pueda enfocar sus esfuerzos hacia profundizar en la utilización de ese recuso o herramienta; caso contrario busque otra alternativa para la solución.

- Se hace necesario que en la formación académica se agreguen materias, que permitan conocer las posibilidades que existen en la actualidad para generar soluciones a problemas como éste, y que den una visión más amplia de lo que e puede hacer con el conocimiento adquirido.

- La recomendación más importante, va en el sentido de que se debe impulsar el desarrollo de soluciones para las distintas necesidades que existen, no solo en el área del transporte sino en otros campos, en los cuales se puede ingresar brindando soluciones prácticas. Pero para esto se debe orientar la formación como ingenieros, a la búsqueda y solución de problemas y de esta forma crear una cultura de desarrollo de tecnología.

REFERENICAS BIBLIOGRAFICAS

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

[2] http://en.wikibooks.org/wiki/Atmel_AVR

[3] http://winavr.scienceprog.com/general-avr/avr- microcontroller-memory-map.html

[4] Data sheet AVR ATmega8. Atmel Corporation. Disponible en: www.atmel.com.

[5] http://www.mikrocontroller.net

[6] http://www.asifunciona.com/electronica/af_gps/af_gps_4.htm

[7] http://www.gutovnik.com/como_func_sist_gps.htm

[8] http://www.solred.com.ar/cinave/papers/funcgps.htm

[9] http://www.elgps.com/documentos/comofuncionagps/comofuncionagps.html

[10] http://home.mira.net/~gnb/gps/nmea.html

[11] http://www.docum.com/tekno/entrela.htm

[12] http://www.ifent.org/Electronica/TVBN/CONCEPTOSTELE.html

[13] http://omega.ilce.edu.mx:3000/sites/ciencia/volumen3/ciencia3/112/htm/sec_24.htm

[14] http://es.wikipedia.org/wiki/PAL

[15] http://www.cybercollege.com/span/tvp009.htm

[16] http://es.wikipedia.org/wiki/SECAM

[17] http://es.wikipedia.org/wiki/NTSC

[18] http://ntsc-tv.com/ntsc-top-03.htm

[19] **http://www.epanorama.net/documents/video/rs170.html**

[20] **http://www.desi.iteso.mx/telecom/siscom/tv/informacion/senal_de_video.html**

[21] **http://www.eyetap.org/ece385/lab5.htm**

[22] **Logacho, David. Generador digital de caracteres electrónicos para video, Quito. 1977.**

[23] **http://es.wikipedia.org/wiki/TELETEXTO**

[24] **http://es.wikipedia.org/wiki/CLOSE_CAPTION**

[25] **http://es.wikipedia.org/wiki/Conector_RCA**

[26] **http://earth.google.es/userguide.html**

[27] **http://es.wikipedia.org/wiki/Streaming**

[28] **http://www.deluo.com/Merchant2/merchant.mv?Screen=PROD&Store_Code=DE&Product_Code=GPSU&Category_Code=FSP**

[29] Data sheet AVR ATmega16. Atmel Corporation. Disponible en: **www.atmel.com**

[30] Data sheet LM1881. National Semiconductor Corporation. Disponible en **http://www.national.com**

ANEXO 1
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 QFN/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

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

ATmega16
ATmega16L

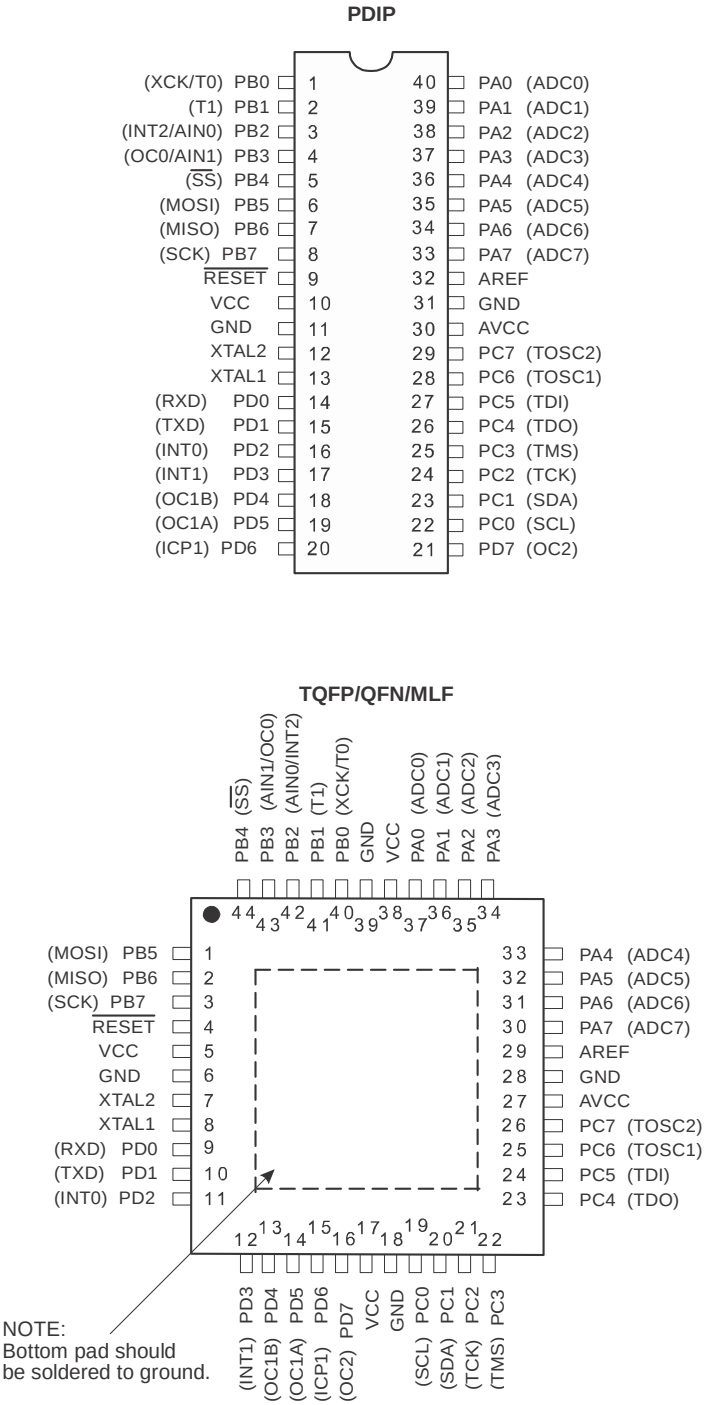
Summary

2466NS–AVR–10/06

Note: This is a summary document. A complete document is available on our Web site at www.atmel.com.

Pin Configurations

Figure 1. Pinout ATmega16



Disclaimer

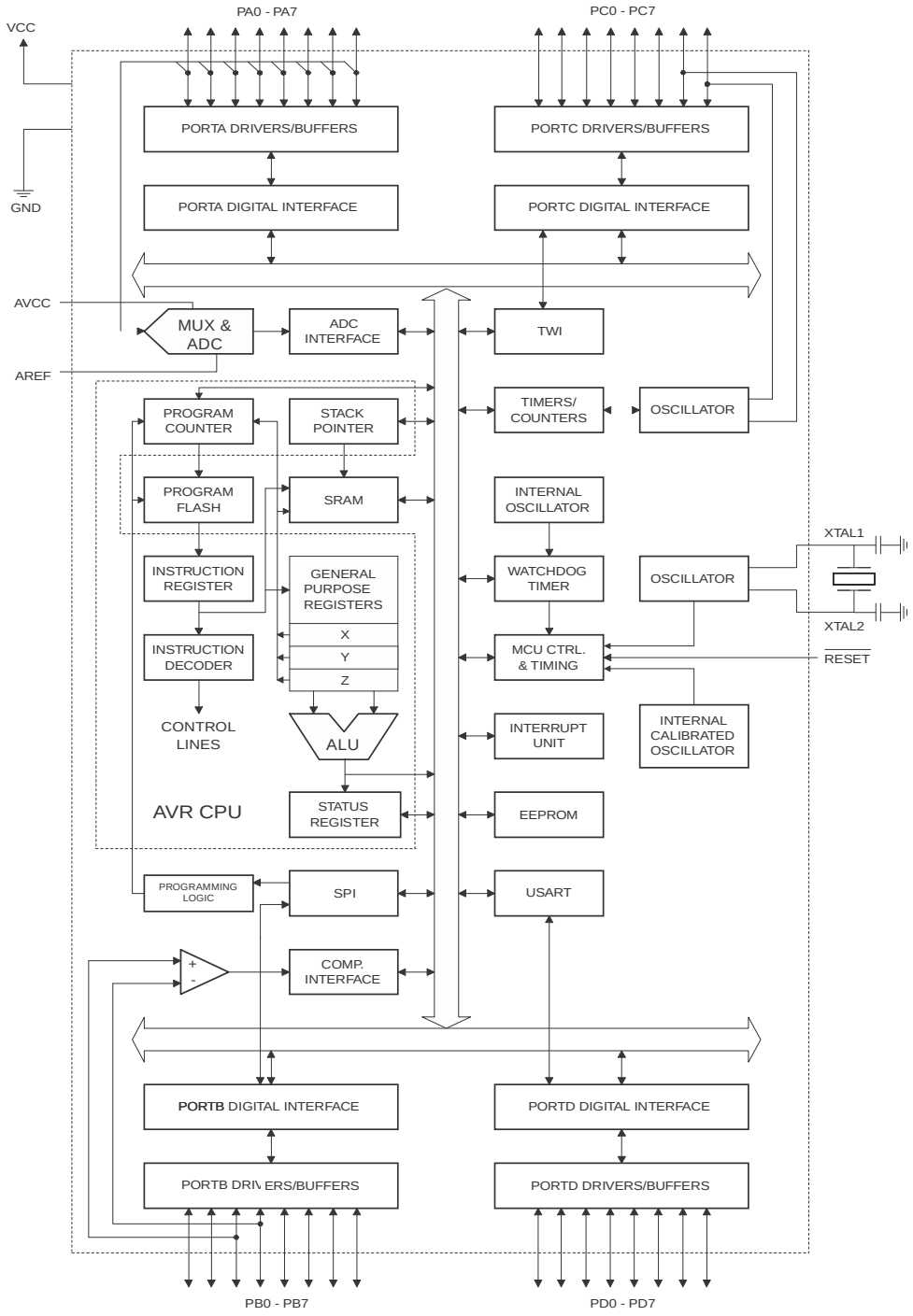
Typical values contained in this datasheet are based on simulations and characterization of other AVR microcontrollers manufactured on the same process technology. Min and Max values will be available after the device is characterized.

Overview

The ATmega16 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega16 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

Block Diagram

Figure 2. Block Diagram



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

The ATmega16 provides the following features: 16K bytes of In-System Programmable Flash Program memory with Read-While-Write capabilities, 512 bytes EEPROM, 1K byte SRAM, 32 general purpose I/O lines, 32 general purpose working registers, a JTAG interface for Boundary-scan, On-chip Debugging support and programming, three flexible Timer/Counters with compare modes, Internal and External Interrupts, a serial programmable USART, a byte oriented Two-wire Serial Interface, an 8-channel, 10-bit ADC with optional differential input stage with programmable gain (TQFP package only), a programmable Watchdog Timer with Internal Oscillator, an SPI serial port, and six software selectable power saving modes. The Idle mode stops the CPU while allowing the USART, Two-wire interface, A/D Converter, SRAM, Timer/Counters, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next External Interrupt or Hardware Reset. In Power-save mode, the Asynchronous Timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except Asynchronous Timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low-power consumption. In Extended Standby mode, both the main Oscillator and the Asynchronous Timer continue to run.

The device is manufactured using Atmel's high density nonvolatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed in-system through an SPI serial interface, by a conventional nonvolatile memory programmer, or by an On-chip Boot program running on the AVR core. The boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega16 is a powerful microcontroller that provides a highly-flexible and cost-effective solution to many embedded control applications.

The ATmega16 AVR is supported with a full suite of program and system development tools including: C compilers, macro assemblers, program debugger/simulators, in-circuit emulators, and evaluation kits.

Pin Descriptions

VCC Digital supply voltage.

GND Ground.

Port A (PA7..PA0) Port A serves as the analog inputs to the A/D Converter.

Port A also serves as an 8-bit bi-directional I/O port, if the A/D Converter is not used. Port pins can provide internal pull-up resistors (selected for each bit). The Port A output buffers have symmetrical drive characteristics with both high sink and source capability. When pins PA0 to PA7 are used as inputs and are externally pulled low, they will source current if the internal pull-up resistors are activated. The Port A pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Port B (PB7..PB0)

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

Port B also serves the functions of various special features of the ATmega16 as listed on page 56.

Port C (PC7..PC0)

Port C is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port C output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running. If the JTAG interface is enabled, the pull-up resistors on pins PC5(TDI), PC3(TMS) and PC2(TCK) will be activated even if a reset occurs.

Port C also serves the functions of the JTAG interface and other special features of the ATmega16 as listed on page 59.

Port D (PD7..PD0)

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

Port D also serves the functions of various special features of the ATmega16 as listed on page 61.

$\overline{\text{RESET}}$

Reset Input. A low level on this pin for longer than the minimum pulse length will generate a reset, even if the clock is not running. The minimum pulse length is given in Table 15 on page 36. Shorter pulses are not guaranteed to generate a reset.

XTAL1

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

XTAL2

Output from the inverting Oscillator amplifier.

AVCC

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

AREF

AREF is the analog reference pin for the A/D Converter.

Resources

A comprehensive set of development tools, application notes and datasheets are available for download on http://www.atmel.com/avr.

Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
$3F ($5F)	SREG	I	T	H	S	V	N	Z	C	7
$3E ($5E)	SPH	–	–	–	–	–	SP10	SP9	SP8	10
$3D ($5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	10
$3C ($5C)	OCR0	Timer/Counter0 Output Compare Register								83
$3B ($5B)	GICR	INT1	INT0	INT2	–	–	–	IVSEL	IVCE	46, 67
$3A ($5A)	GIFR	INTF1	INTF0	INTF2	–	–	–	–	–	68
$39 ($59)	TIMSK	OCIE2	TOIE2	TICIE1	OCIE1A	OCIE1B	TOIE1	OCIE0	TOIE0	83, 114, 132
$38 ($58)	TIFR	OCF2	TOV2	ICF1	OCF1A	OCF1B	TOV1	OCF0	TOV0	84, 115, 132
$37 ($57)	SPMCR	SPMIE	RWWSB	–	RWWSRE	BLBSET	PGWRT	PGERS	SPMEN	250
$36 ($56)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	178
$35 ($55)	MCUCR	SM2	SE	SM1	SM0	ISC11	ISC10	ISC01	ISC00	30, 66
$34 ($54)	MCUCSR	JTD	ISC2	–	JTRF	WDRF	BORF	EXTRF	PORF	39, 67, 229
$33 ($53)	TCCR0	FOC0	WGM00	COM01	COM00	WGM01	CS02	CS01	CS00	81
$32 ($52)	TCNT0	Timer/Counter0 (8 Bits)								83
$31[1] ($51)[1]	OSCCAL	Oscillator Calibration Register								28
	OCDR	On-Chip Debug Register								225
$30 ($50)	SFIOR	ADTS2	ADTS1	ADTS0	–	ACME	PUD	PSR2	PSR10	55,86,133,199,219
$2F ($4F)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	FOC1A	FOC1B	WGM11	WGM10	109
$2E ($4E)	TCCR1B	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	112
$2D ($4D)	TCNT1H	Timer/Counter1 – Counter Register High Byte								113
$2C ($4C)	TCNT1L	Timer/Counter1 – Counter Register Low Byte								113
$2B ($4B)	OCR1AH	Timer/Counter1 – Output Compare Register A High Byte								113
$2A ($4A)	OCR1AL	Timer/Counter1 – Output Compare Register A Low Byte								113
$29 ($49)	OCR1BH	Timer/Counter1 – Output Compare Register B High Byte								113
$28 ($48)	OCR1BL	Timer/Counter1 – Output Compare Register B Low Byte								113
$27 ($47)	ICR1H	Timer/Counter1 – Input Capture Register High Byte								114
$26 ($46)	ICR1L	Timer/Counter1 – Input Capture Register Low Byte								114
$25 ($45)	TCCR2	FOC2	WGM20	COM21	COM20	WGM21	CS22	CS21	CS20	127
$24 ($44)	TCNT2	Timer/Counter2 (8 Bits)								129
$23 ($43)	OCR2	Timer/Counter2 Output Compare Register								129
$22 ($42)	ASSR	–	–	–	–	AS2	TCN2UB	OCR2UB	TCR2UB	130
$21 ($41)	WDTCR	–	–	–	WDTOE	WDE	WDP2	WDP1	WDP0	41
$20[2] ($40)[2]	UBRRH	URSEL	–	–	–	UBRR[11:8]				165
	UCSRC	URSEL	UMSEL	UPM1	UPM0	USBS	UCSZ1	UCSZ0	UCPOL	164
$1F ($3F)	EEARH	–	–	–	–	–	–	–	EEAR8	17
$1E ($3E)	EEARL	EEPROM Address Register Low Byte								17
$1D ($3D)	EEDR	EEPROM Data Register								17
$1C ($3C)	EECR	–	–	–	–	EERIE	EEMWE	EEWE	EERE	17
$1B ($3B)	PORTA	PORTA7	PORTA6	PORTA5	PORTA4	PORTA3	PORTA2	PORTA1	PORTA0	64
$1A ($3A)	DDRA	DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0	64
$19 ($39)	PINA	PINA7	PINA6	PINA5	PINA4	PINA3	PINA2	PINA1	PINA0	64
$18 ($38)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	64
$17 ($37)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	64
$16 ($36)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	64
$15 ($35)	PORTC	PORTC7	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	65
$14 ($34)	DDRC	DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	65
$13 ($33)	PINC	PINC7	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	65
$12 ($32)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	65
$11 ($31)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	65
$10 ($30)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	65
$0F ($2F)	SPDR	SPI Data Register								140
$0E ($2E)	SPSR	SPIF	WCOL	–	–	–	–	–	SPI2X	140
$0D ($2D)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	138
$0C ($2C)	UDR	USART I/O Data Register								161
$0B ($2B)	UCSRA	RXC	TXC	UDRE	FE	DOR	PE	U2X	MPCM	162
$0A ($2A)	UCSRB	RXCIE	TXCIE	UDRIE	RXEN	TXEN	UCSZ2	RXB8	TXB8	163
$09 ($29)	UBRRL	USART Baud Rate Register Low Byte								165
$08 ($28)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	200
$07 ($27)	ADMUX	REFS1	REFS0	ADLAR	MUX4	MUX3	MUX2	MUX1	MUX0	215
$06 ($26)	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	217
$05 ($25)	ADCH	ADC Data Register High Byte								218
$04 ($24)	ADCL	ADC Data Register Low Byte								218
$03 ($23)	TWDR	Two-wire Serial Interface Data Register								180
$02 ($22)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	180

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
$01 ($21)	TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	179
$00 ($20)	TWBR	Two-wire Serial Interface Bit Rate Register								178

Notes:
1. When the OCDEN Fuse is unprogrammed, the OSCCAL Register is always accessed on this address. Refer to the debugger specific documentation for details on how to use the OCDR Register.
2. Refer to the USART description for details on how to access UBRRH and UCSRC.
3. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.
4. Some of the Status Flags are cleared by writing a logical one to them. Note that the CBI and SBI instructions will operate on all bits in the I/O Register, writing a one back into any flag read as set, thus clearing the flag. The CBI and SBI instructions work with registers $00 to $1F only.

Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl,K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	Rd ← Rd - K - C	Z,C,N,V,H	1
SBIW	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	Rd ← Rd • Rr	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	Rd ← Rd • K	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	One's Complement	Rd ← $FF – Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← $00 – Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • ($FF - K)	Z,N,V	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd – 1	Z,N,V	1
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V	1
SER	Rd	Set Register	Rd ← $FF	None	1
MUL	Rd, Rr	Multiply Unsigned	R1:R0 ← Rd x Rr	Z,C	2
MULS	Rd, Rr	Multiply Signed	R1:R0 ← Rd x Rr	Z,C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	R1:R0 ← Rd x Rr	Z,C	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
FMULS	Rd, Rr	Fractional Multiply Signed	R1:R0 ← (Rd x Rr) << 1	Z,C	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
BRANCH INSTRUCTIONS					
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC ← Z	None	2
JMP	k	Direct Jump	PC ← k	None	3
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	3
ICALL		Indirect Call to (Z)	PC ← Z	None	3
CALL	k	Direct Subroutine Call	PC ← k	None	4
RET		Subroutine Return	PC ← STACK	None	4
RETI		Interrupt Return	PC ← STACK	I	4
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1 / 2 / 3
CP	Rd,Rr	Compare	Rd – Rr	Z, N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	Rd – Rr – C	Z, N,V,C,H	1
CPI	Rd,K	Compare Register with Immediate	Rd – K	Z, N,V,C,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) PC ← PC + 2 or 3	None	1 / 2 / 3
SBRS	Rr, b	Skip if Bit in Register is Set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1 / 2 / 3
SBIC	P, b	Skip if Bit in I/O Register Cleared	if (P(b)=0) PC ← PC + 2 or 3	None	1 / 2 / 3
SBIS	P, b	Skip if Bit in I/O Register is Set	if (P(b)=1) PC ← PC + 2 or 3	None	1 / 2 / 3
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC←PC+k + 1	None	1 / 2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC←PC+k + 1	None	1 / 2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1 / 2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1 / 2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1 / 2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1 / 2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1 / 2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1 / 2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1 / 2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1 / 2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1 / 2
BRLT	k	Branch if Less Than Zero, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1 / 2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1 / 2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1 / 2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1 / 2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1 / 2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1 / 2
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1 / 2

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

Mnemonics	Operands	Description	Operation	Flags	#Clocks
CLH		Clear Half Carry Flag in SREG	H ← 0	H	1
MCU CONTROL INSTRUCTIONS					
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep function)	None	1
WDR		Watchdog Reset	(see specific descr. for WDR/timer)	None	1
BREAK		Break	For On-Chip Debug Only	None	N/A

Ordering Information

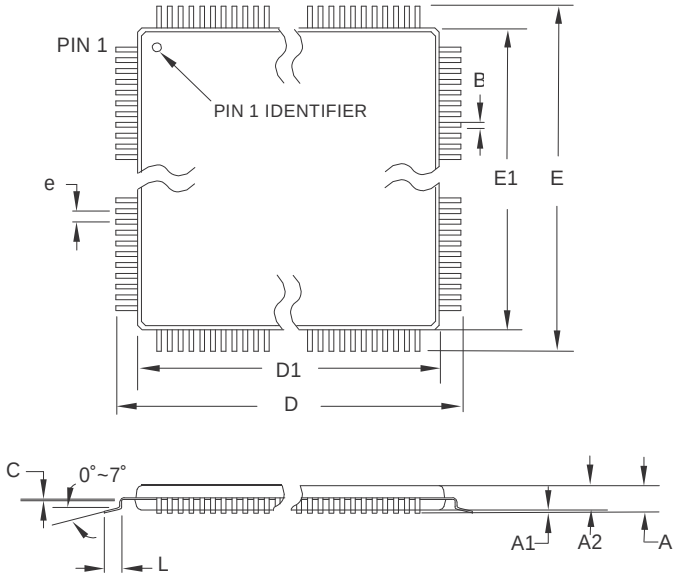
Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
8	2.7 - 5.5V	ATmega16L-8AC ATmega16L-8PC ATmega16L-8MC	44A 40P6 44M1	Commercial (0°C to 70°C)
		ATmega16L-8AI ATmega16L-8AU[1] ATmega16L-8PI ATmega16L-8PU[1] ATmega16L-8MI ATmega16L-8MU[1]	44A 44A 40P6 40P6 44M1 44M1	Industrial (-40°C to 85°C)
16	4.5 - 5.5V	ATmega16-16AC ATmega16-16PC ATmega16-16MC	44A 40P6 44M1	Commercial (0°C to 70°C)
		ATmega16-16AI ATmega16-16AU[1] ATmega16-16PI ATmega16-16PU[1] ATmega16-16MI ATmega16-16MU[1]	44A 44A 40P6 40P6 44M1 44M1	Industrial (-40°C to 85°C)

Note: 1. Pb-free packaging alternative, complies to the European Directive for Restriction of Hazardous Substances (RoHS directive). Also Halide free and fully Green.

Package Type	
44A	44-lead, Thin (1.0 mm) Plastic Gull Wing Quad Flat Package (TQFP)
40P6	40-pin, 0.600” Wide, Plastic Dual Inline Package (PDIP)
44M1	44-pad, 7 x 7 x 1.0 mm body, lead pitch 0.50 mm, Quad Flat No-Lead/Micro Lead Frame Package (QFN/MLF)

Packaging Information

44A



COMMON DIMENSIONS
(Unit of Measure = mm)

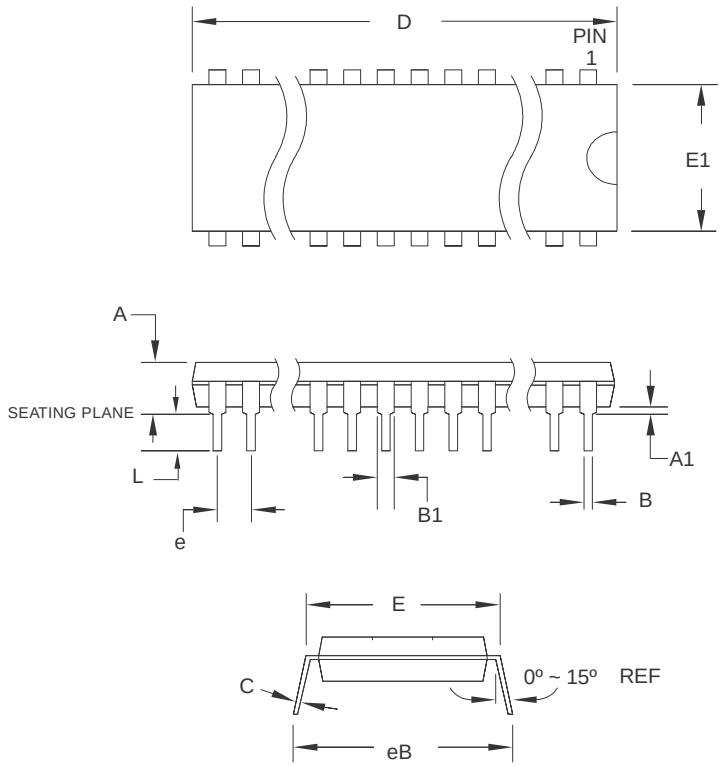
SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	1.20	
A1	0.05	–	0.15	
A2	0.95	1.00	1.05	
D	11.75	12.00	12.25	
D1	9.90	10.00	10.10	Note 2
E	11.75	12.00	12.25	
E1	9.90	10.00	10.10	Note 2
B	0.30	–	0.45	
C	0.09	–	0.20	
L	0.45	–	0.75	
e	0.80 TYP			

Notes:
1. This package conforms to JEDEC reference MS-026, Variation ACB.
2. Dimensions D1 and E1 do not include mold protrusion. Allowable protrusion is 0.25 mm per side. Dimensions D1 and E1 are maximum plastic body size dimensions including mold mismatch.
3. Lead coplanarity is 0.10 mm maximum.

10/5/2001

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**44A,** 44-lead, 10 x 10 mm Body Size, 1.0 mm Body Thickness, 0.8 mm Lead Pitch, Thin Profile Plastic Quad Flat Package (TQFP)	44A	B

40P6



COMMON DIMENSIONS
(Unit of Measure = mm)

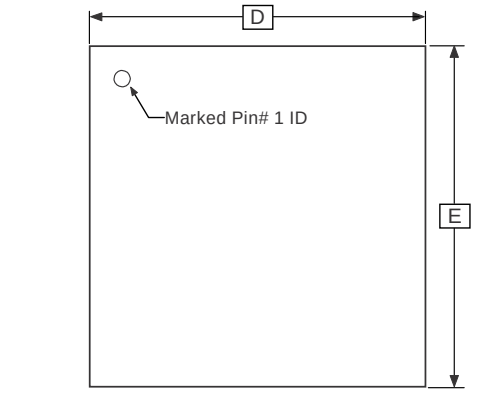
SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	4.826	
A1	0.381	–	–	
D	52.070	–	52.578	Note 2
E	15.240	–	15.875	
E1	13.462	–	13.970	Note 2
B	0.356	–	0.559	
B1	1.041	–	1.651	
L	3.048	–	3.556	
C	0.203	–	0.381	
eB	15.494	–	17.526	
e	2.540 TYP			

Notes: 1. This package conforms to JEDEC reference MS-011, Variation AC.
2. Dimensions D and E1 do not include mold Flash or Protrusion. Mold Flash or Protrusion shall not exceed 0.25 mm (0.010").

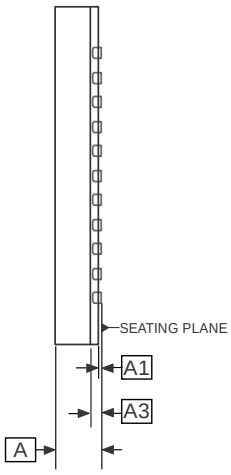
09/28/01

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**40P6**, 40-lead (0.600"/15.24 mm Wide) Plastic Dual Inline Package (PDIP)	40P6	B

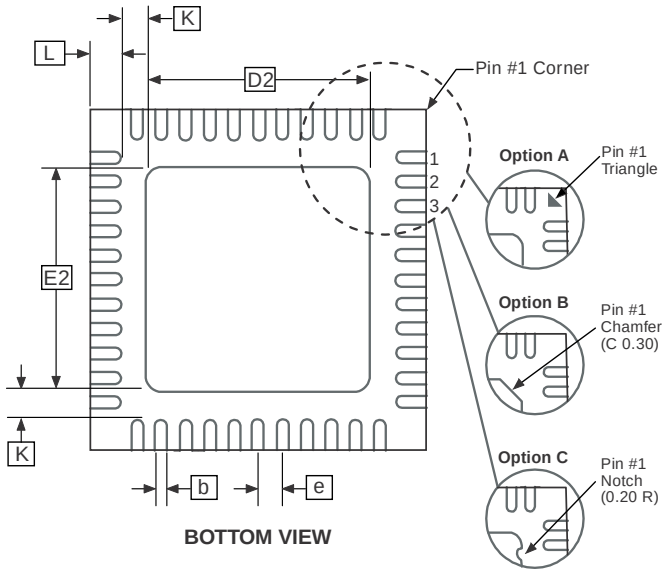
44M1



TOP VIEW



SIDE VIEW



BOTTOM VIEW

Note: JEDEC Standard MO-220, Fig. 1 (SAW Singulation) VKKD-3.

COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	0.80	0.90	1.00	
A1	–	0.02	0.05	
A3		0.25 REF		
b	0.18	0.23	0.30	
D	6.90	7.00	7.10	
D2	5.00	5.20	5.40	
E	6.90	7.00	7.10	
E2	5.00	5.20	5.40	
e		0.50 BSC		
L	0.59	0.64	0.69	
K	0.20	0.26	0.41	

5/27/06

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**44M1**, 44-pad, 7 x 7 x 1.0 mm Body, Lead Pitch 0.50 mm, 5.20 mm Exposed Pad, Micro Lead Frame Package (MLF)	44M1	G

Errata

The revision letter in this section refers to the revision of the ATmega16 device.

ATmega16(L) Rev. M

- **First Analog Comparator conversion may be delayed**
- **Interrupts may be lost when writing the timer registers in the asynchronous timer**
- **IDCODE masks data from TDI input**

1. **First Analog Comparator conversion may be delayed**

   If the device is powered by a slow rising V_{CC}, the first Analog Comparator conversion will take longer than expected on some devices.

   Problem Fix/Workaround

   When the device has been powered or reset, disable then enable theAnalog Comparator before the first conversion.

2. **Interrupts may be lost when writing the timer registers in the asynchronous timer**

   If one of the timer registers which is synchronized to the asynchronous timer2 clock is written in the cycle before a overflow interrupt occurs, the interrupt may be lost.

   Problem Fix/Workaround

   Always check that the Timer2 Timer/Counter register, TCNT2, does not have the value 0xFF before writing the Timer2 Control Register, TCCR2, or Output Compare Register, OCR2

3. **IDCODE masks data from TDI input**

   The JTAG instruction IDCODE is not working correctly. Data to succeeding devices are replaced by all-ones during Update-DR.

   Problem Fix / Workaround

   – If ATmega16 is the only device in the scan chain, the problem is not visible.

   – Select the Device ID Register of the ATmega16 by issuing the IDCODE instruction or by entering the Test-Logic-Reset state of the TAP controller to read out the contents of its Device ID Register and possibly data from succeeding devices of the scan chain. Issue the BYPASS instruction to the ATmega16 while reading the Device ID Registers of preceding devices of the boundary scan chain.

   – If the Device IDs of all devices in the boundary scan chain must be captured simultaneously, the ATmega16 must be the fist device in the chain.

ATmega16(L) Rev. L

- **First Analog Comparator conversion may be delayed**
- **Interrupts may be lost when writing the timer registers in the asynchronous timer**
- **IDCODE masks data from TDI input**

1. **First Analog Comparator conversion may be delayed**

   If the device is powered by a slow rising V_{CC}, the first Analog Comparator conversion will take longer than expected on some devices.

   Problem Fix/Workaround

   When the device has been powered or reset, disable then enable theAnalog Comparator before the first conversion.

2. **Interrupts may be lost when writing the timer registers in the asynchronous timer**

If one of the timer registers which is synchronized to the asynchronous timer2 clock is written in the cycle before a overflow interrupt occurs, the interrupt may be lost.

Problem Fix/Workaround

Always check that the Timer2 Timer/Counter register, TCNT2, does not have the value 0xFF before writing the Timer2 Control Register, TCCR2, or Output Compare Register, OCR2

3. IDCODE masks data from TDI input

The JTAG instruction IDCODE is not working correctly. Data to succeeding devices are replaced by all-ones during Update-DR.

Problem Fix / Workaround

– If ATmega16 is the only device in the scan chain, the problem is not visible.

– Select the Device ID Register of the ATmega16 by issuing the IDCODE instruction or by entering the Test-Logic-Reset state of the TAP controller to read out the contents of its Device ID Register and possibly data from succeeding devices of the scan chain. Issue the BYPASS instruction to the ATmega16 while reading the Device ID Registers of preceding devices of the boundary scan chain.

– If the Device IDs of all devices in the boundary scan chain must be captured simultaneously, the ATmega16 must be the fist device in the chain.

ATmega16(L) Rev. K

- **First Analog Comparator conversion may be delayed**
- **Interrupts may be lost when writing the timer registers in the asynchronous timer**
- **IDCODE masks data from TDI input**

1. First Analog Comparator conversion may be delayed

If the device is powered by a slow rising V_{CC}, the first Analog Comparator conversion will take longer than expected on some devices.

Problem Fix/Workaround

When the device has been powered or reset, disable then enable theAnalog Comparator before the first conversion.

2. Interrupts may be lost when writing the timer registers in the asynchronous timer

If one of the timer registers which is synchronized to the asynchronous timer2 clock is written in the cycle before a overflow interrupt occurs, the interrupt may be lost.

Problem Fix/Workaround

Always check that the Timer2 Timer/Counter register, TCNT2, does not have the value 0xFF before writing the Timer2 Control Register, TCCR2, or Output Compare Register, OCR2

3. IDCODE masks data from TDI input

The JTAG instruction IDCODE is not working correctly. Data to succeeding devices are replaced by all-ones during Update-DR.

Problem Fix / Workaround

– If ATmega16 is the only device in the scan chain, the problem is not visible.

– Select the Device ID Register of the ATmega16 by issuing the IDCODE instruction or by entering the Test-Logic-Reset state of the TAP controller to read out the contents of its Device ID Register and possibly data from

succeeding devices of the scan chain. Issue the BYPASS instruction to the ATmega16 while reading the Device ID Registers of preceding devices of the boundary scan chain.

– If the Device IDs of all devices in the boundary scan chain must be captured simultaneously, the ATmega16 must be the fist device in the chain.

ATmega16(L) Rev. J

- **First Analog Comparator conversion may be delayed**
- **Interrupts may be lost when writing the timer registers in the asynchronous timer**
- **IDCODE masks data from TDI input**

1. First Analog Comparator conversion may be delayed

If the device is powered by a slow rising V_{CC}, the first Analog Comparator conversion will take longer than expected on some devices.

Problem Fix/Workaround

When the device has been powered or reset, disable then enable theAnalog Comparator before the first conversion.

2. Interrupts may be lost when writing the timer registers in the asynchronous timer

If one of the timer registers which is synchronized to the asynchronous timer2 clock is written in the cycle before a overflow interrupt occurs, the interrupt may be lost.

Problem Fix/Workaround

Always check that the Timer2 Timer/Counter register, TCNT2, does not have the value 0xFF before writing the Timer2 Control Register, TCCR2, or Output Compare Register, OCR2

3. IDCODE masks data from TDI input

The JTAG instruction IDCODE is not working correctly. Data to succeeding devices are replaced by all-ones during Update-DR.

Problem Fix / Workaround

– If ATmega16 is the only device in the scan chain, the problem is not visible.

– Select the Device ID Register of the ATmega16 by issuing the IDCODE instruction or by entering the Test-Logic-Reset state of the TAP controller to read out the contents of its Device ID Register and possibly data from succeeding devices of the scan chain. Issue the BYPASS instruction to the ATmega16 while reading the Device ID Registers of preceding devices of the boundary scan chain.

– If the Device IDs of all devices in the boundary scan chain must be captured simultaneously, the ATmega16 must be the fist device in the chain.

ATmega16(L) Rev. I

- **First Analog Comparator conversion may be delayed**
- **Interrupts may be lost when writing the timer registers in the asynchronous timer**
- **IDCODE masks data from TDI input**

1. First Analog Comparator conversion may be delayed

If the device is powered by a slow rising V_{CC}, the first Analog Comparator conversion will take longer than expected on some devices.

Problem Fix/Workaround

When the device has been powered or reset, disable then enable theAnalog Comparator before the first conversion.

2. Interrupts may be lost when writing the timer registers in the asynchronous timer

If one of the timer registers which is synchronized to the asynchronous timer2 clock is written in the cycle before a overflow interrupt occurs, the interrupt may be lost.

Problem Fix/Workaround

Always check that the Timer2 Timer/Counter register, TCNT2, does not have the value 0xFF before writing the Timer2 Control Register, TCCR2, or Output Compare Register, OCR2

3. IDCODE masks data from TDI input

The JTAG instruction IDCODE is not working correctly. Data to succeeding devices are replaced by all-ones during Update-DR.

Problem Fix / Workaround

– If ATmega16 is the only device in the scan chain, the problem is not visible.

– Select the Device ID Register of the ATmega16 by issuing the IDCODE instruction or by entering the Test-Logic-Reset state of the TAP controller to read out the contents of its Device ID Register and possibly data from succeeding devices of the scan chain. Issue the BYPASS instruction to the ATmega16 while reading the Device ID Registers of preceding devices of the boundary scan chain.

– If the Device IDs of all devices in the boundary scan chain must be captured simultaneously, the ATmega16 must be the fist device in the chain.

ATmega16(L) Rev. H

- **First Analog Comparator conversion may be delayed**
- **Interrupts may be lost when writing the timer registers in the asynchronous timer**
- **IDCODE masks data from TDI input**

1. First Analog Comparator conversion may be delayed

If the device is powered by a slow rising V_{CC}, the first Analog Comparator conversion will take longer than expected on some devices.

Problem Fix/Workaround

When the device has been powered or reset, disable then enable theAnalog Comparator before the first conversion.

2. Interrupts may be lost when writing the timer registers in the asynchronous timer

If one of the timer registers which is synchronized to the asynchronous timer2 clock is written in the cycle before a overflow interrupt occurs, the interrupt may be lost.

Problem Fix/Workaround

Always check that the Timer2 Timer/Counter register, TCNT2, does not have the value 0xFF before writing the Timer2 Control Register, TCCR2, or Output Compare Register, OCR2

3. IDCODE masks data from TDI input

The JTAG instruction IDCODE is not working correctly. Data to succeeding devices are replaced by all-ones during Update-DR.

Problem Fix / Workaround

– If ATmega16 is the only device in the scan chain, the problem is not visible.

– Select the Device ID Register of the ATmega16 by issuing the IDCODE instruction or by entering the Test-Logic-Reset state of the TAP controller to read out the contents of its Device ID Register and possibly data from succeeding devices of the scan chain. Issue the BYPASS instruction to the ATmega16 while reading the Device ID Registers of preceding devices of the boundary scan chain.
– If the Device IDs of all devices in the boundary scan chain must be captured simultaneously, the ATmega16 must be the fist device in the chain.

Datasheet Revision History

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

Rev. 2466N-10/06

1. **Updated “Timer/Counter Oscillator” on page 31.**
2. **Updated “Fast PWM Mode” on page 102.**
3. **Updated Table 38 on page 83, Table 40 on page 84, Table 45 on page 112, Table 47 on page 113, Table 50 on page 129 and Table 52 on page 130.**
4. **Updated C code example in “USART Initialization” on page 150.**
5. **Updated “Errata” on page 343.**

Rev. 2466M-04/06

1. **Updated typos.**
2. **Updated “Serial Peripheral Interface – SPI” on page 136.**
3. **Updated Table 86 on page 222, Table 116 on page 279 ,Table 121 on page 298 and Table 122 on page 300.**

Rev. 2466L-06/05

1. **Updated note in “Bit Rate Generator Unit” on page 179.**
2. **Updated values for V_{INT} in “ADC Characteristics” on page 300.**
3. **Updated “Serial Programming Instruction set” on page 279.**
4. **Updated USART init C-code example in “USART” on page 145.**

Rev. 2466K-04/05

1. **Updated “Ordering Information” on page 11.**
2. **MLF-package alternative changed to “Quad Flat No-Lead/Micro Lead Frame Package QFN/MLF”.**
3. **Updated “Electrical Characteristics” on page 294.**

Rev. 2466J-10/04

1. **Updated “Ordering Information” on page 11.**

Rev. 2466I-10/04

1. **Removed references to analog ground.**
2. **Updated Table 7 on page 28, Table 15 on page 38, Table 16 on page 42, Table 81 on page 211, Table 116 on page 279, and Table 119 on page 296.**
3. **Updated “Pinout ATmega16” on page 2.**
4. **Updated features in “Analog to Digital Converter” on page 205.**
5. **Updated “Version” on page 230.**
6. **Updated “Calibration Byte” on page 264.**

7. Added “Page Size” on page 265.

Rev. 2466H-12/03

1. Updated “Calibrated Internal RC Oscillator” on page 29.

Rev. 2466G-10/03

1. Removed “Preliminary” from the datasheet.
2. Changed ICP to ICP1 in the datasheet.
3. Updated “JTAG Interface and On-chip Debug System” on page 36.
4. Updated assembly and C code examples in “Watchdog Timer Control Register – WDTCR” on page 43.
5. Updated Figure 46 on page 103.
6. Updated Table 15 on page 38, Table 82 on page 218 and Table 115 on page 279.
7. Updated “Test Access Port – TAP” on page 223 regarding JTAGEN.
8. Updated description for the JTD bit on page 232.
9. Added note 2 to Figure 126 on page 255.
10. Added a note regarding JTAGEN fuse to Table 105 on page 263.
11. Updated Absolute Maximum Ratings* and DC Characteristics in “Electrical Characteristics” on page 294.
12. Updated “ATmega16 Typical Characteristics” on page 302.
13. Fixed typo for 16 MHz QFN/MLF package in “Ordering Information” on page 11.
14. Added a proposal for solving problems regarding the JTAG instruction IDCODE in “Errata” on page 15.

Rev. 2466F-02/03

1. Added note about masking out unused bits when reading the Program Counter in “Stack Pointer” on page 12.
2. Added Chip Erase as a first step in “Programming the Flash” on page 291 and “Programming the EEPROM” on page 292.
3. Added the section “Unconnected pins” on page 55.
4. Added tips on how to disable the OCD system in “On-chip Debug System” on page 34.
5. Removed reference to the “Multi-purpose Oscillator” application note and “32 kHz Crystal Oscillator” application note, which do not exist.
6. Added information about PWM symmetry for Timer0 and Timer2.

7. **Added note in "Filling the Temporary Buffer (Page Loading)" on page 256 about writing to the EEPROM during an SPM Page Load.**

8. **Removed ADHSM completely.**

9. **Added Table 73, "TWI Bit Rate Prescaler," on page 183 to describe the TWPS bits in the "TWI Status Register – TWSR" on page 182.**

10. **Added section "Default Clock Source" on page 25.**

11. **Added note about frequency variation when using an external clock. Note added in "External Clock" on page 31. An extra row and a note added in Table 118 on page 296.**

12. **Various minor TWI corrections.**

13. **Added "Power Consumption" data in "Features" on page 1.**

14. **Added section "EEPROM Write During Power-down Sleep Mode" on page 22.**

15. **Added note about Differential Mode with Auto Triggering in "Prescaling and Conversion Timing" on page 208.**

16. **Added updated "Packaging Information" on page 12.**

Rev. 2466E-10/02

1. **Updated "DC Characteristics" on page 294.**

Rev. 2466D-09/02

1. **Changed all Flash write/erase cycles from 1,000 to 10,000.**

2. **Updated the following tables: Table 4 on page 26, Table 15 on page 38, Table 42 on page 85, Table 45 on page 112, Table 46 on page 112, Table 59 on page 144, Table 67 on page 168, Table 90 on page 237, Table 102 on page 261, "DC Characteristics" on page 294, Table 119 on page 296, Table 121 on page 298, and Table 122 on page 300.**

3. **Updated "Errata" on page 15.**

Rev. 2466C-03/02

1. **Updated typical EEPROM programming time, Table 1 on page 20.**

2. **Updated typical start-up time in the following tables:**

   Table 3 on page 25, Table 5 on page 27, Table 6 on page 28, Table 8 on page 29, Table 9 on page 29, and Table 10 on page 30.

3. **Updated Table 17 on page 43 with typical WDT Time-out.**

4. **Added Some Preliminary Test Limits and Characterization Data.**

   Removed some of the TBD's in the following tables and pages:

   Table 15 on page 38, Table 16 on page 42, Table 116 on page 272 (table removed in document review #D), "Electrical Characteristics" on page 294, Table 119 on page 296, Table 121 on page 298, and Table 122 on page 300.

5. **Updated TWI Chapter.**

Added the note at the end of the “Bit Rate Generator Unit” on page 179.

6. **Corrected description of ADSC bit in “ADC Control and Status Register A – ADCSRA” on page 220.**

7. **Improved description on how to do a polarity check of the ADC doff results in “ADC Conversion Result” on page 217.**

8. **Added JTAG version number for rev. H in Table 87 on page 230.**

9. **Added not regarding OCDEN Fuse below Table 105 on page 263.**

10. **Updated Programming Figures:**

    Figure 127 on page 265 and Figure 136 on page 277 are updated to also reflect that AVCC must be connected during Programming mode. Figure 131 on page 273 added to illustrate how to program the fuses.

11. **Added a note regarding usage of the “PROG_PAGELOAD ($6)” on page 283 and “PROG_PAGEREAD ($7)” on page 283.**

12. **Removed alternative algortihm for leaving JTAG Programming mode.**

    See “Leaving Programming Mode” on page 291.

13. **Added Calibrated RC Oscillator characterization curves in section “ATmega16 Typical Characteristics” on page 302.**

14. **Corrected ordering code for QFN/MLF package (16MHz) in “Ordering Information” on page 11.**

15. **Corrected Table 90, “Scan Signals for the Oscillators(1)(2)(3),” on page 237.**

Atmel Corporation

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

Regional Headquarters

Europe
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

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

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

Atmel Operations

Memory
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/
High Speed Converters/RF Datacom
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

Literature Requests
www.atmel.com/literature

2466NS–AVR–10/06

ANEXO 2
RECEPTOR GPS

Deluo GPS Lite for Laptop & PDA (Serial, USB or PDA)

Deluo GPS Lite for Laptop & PDA (Serial, USB or PDA)

SKU: 90-020

Our price: **$ 69.90**

Quantity **This product is currently out of stock**

Product Summary

A Flexible and Affordable GPS solution for your Laptop or PDA. Improve the accuracy of your navigation software with Deluo GPS Lite.

The Deluo GPS Lite is a powerful tool that provides accurate positioning and will show your position using almost any mapping application.

Tracking up to 12 satellites, the Deluo GPS Lite can be used with almost every major mapping software (NMEA 0183 v2.2 data protocol). Use it in your car, boat or RV; the Deluo GPS Lite can be used anywhere in world with no monthly service fees. The GPS unit comes with an internal magnet and Velcro® dots for versatile mounting options.

For added versatility, you can also use the Deluo GPS Lite with your PDA by purchasing a GPS adapter cable for any supported PDA models, allowing you to use a single GPS receiver with multiple devices.**

For Serial, USB or PDA Connection, please select the right adapter for you from the optional adapter list.

**** PDA use may require additional mapping software depending on your PDA's operating system.**

Not sure which GPS will best meet your needs? Click here to

compare the Deluo GPS units specs side-by-side.

This GPS receiver is based on Sony's GPS chipset, which has an advanced RFCMOS block signal processing architecture. This translates to high sensitivity, minimal power consumption and high-performance. The new LSI (Large Scale Integration) configuration is ideal for a wide range of location-based applications such as automotive, cellular handsets, handheld navigation, fleet management, and mobile computing devices.

Specifications:

Accuracy	Position: 5 to 25 meters Velocity: 0.1 m/sec Time: ±1μ sec
Acquisition	Cold start: 50 sec (Average) Warm start: 35 sec (Average) Hot start: 2 sec (Min.)
Sensitivity	Acquisition: -139 dBm Tracking: -152 dBm
Dynamics	Altitude: 18,000m (Max.) Velocity: 500m/sec (Max.) Acceleration: ±4G (Max.)
Navigation update rate	Once per second
Serial I/O	Series RS-232 signal 4800bps NMEA 0183:GGA, GSV, GSA, RMC,(optional VTG,GLL)
Coordinate Datum	WGS-84
Power supply	+5VDC and up to +9VDC
Power consumption	Typical (tracking): 120mW
Sleep mode power saving	Yes (automatically)
RTC support	Yes
Water-resistant	Yes
System Performance	Tracks up to 12 satellites L1, C/A code
Operating Temperature	-40°~ +85°C
Storage Temperature	-55°~ +100°C
Humidity	5%~95%
Dimensions	41 x 41 x 18 mm

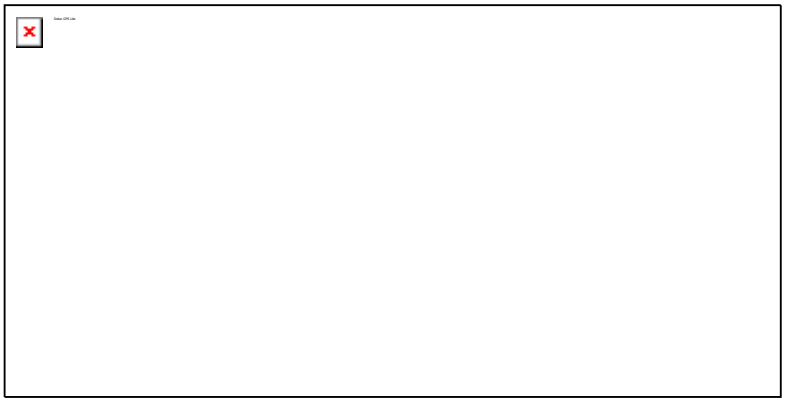
GPS compatible with almost every major mapping software, including:

- Deluo Routis
- Microsoft Streets and Trips
- Microsoft MapPoint
- Rand McNally Street Finder
- National Geographic TOPO!
- Delorme Stret Atlas
- Fugawi

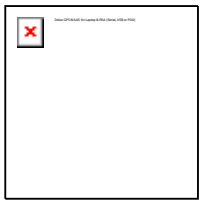
- OziExplorer
- TomTom Navigator
- Via Michelin
- Destinator
- Route 66
- and most other softwares that support NMEA data protocol (*)

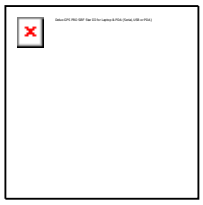
(*) Please email us at support@deluo.com regarding software compatibility

Detailed images



Deluo GPS Matrix





		Deluo GPS WAAS	Deluo GPS PRO
	GPS Processor	Sony CXD2951GA-4	SiRF Star III
	Channels	12	20
Accuracy	**Position**	2 m (Approx.)	<5 m
	Velocity	0.1 m/s	0.1 m/s
	Time	± 1μs	± 1μs
	WAAS/EGNOS enabled	**YES**	**YES**
Acquisition	**Reacquisition**	< 0.1 sec. (avg)	0.1 sec. (avg)
	Hot Start	< 2 sec. (avg)	1 sec. (min)
	Warm Start	< 33 sec. (avg)	38 sec. (avg)
	Cold Start	< 40 sec. (avg)	42 sec. (avg)
	Sensitivity	-152 dBm	-159 dBm
User protocol		NMEA-0183 v2.20	NMEA-0183 v2.20 SiRF (binary)
	GGA	YES	YES
	GSV	YES	YES
	GSA	YES	YES
	RMC	YES	YES
	VTG	OPTIONAL	OPTIONAL
	GLL	OPTIONAL	OPTIONAL
Dynamics	**Altitude**	18,000m	18,000m
	Velocity	500m/sec	515m/sec
	Acceleration	4G (max)	<4G
	Navigation update rate	1Hz	1Hz
	Coordinate Datum	WGS 84	WGS 84
	Power supply	5V	5V
	Power consumption	120mA	80 mA
	Power saving	Yes	Yes
	RTC support	Yes	Yes
	Extended warranty	Optional	Optional

ANEXO 3
ESTÁNDAR NMEA

NMEA data

Table of Contents

Disclaimer

This site is based on personal research and is believed to be accurate but there is no guarantee that any of the information is correct or suitable for any purpose. I have been told by the NMEA folks that my information is old and out of date. The current version of NMEA at the time I wrote this is 3.01 which is not described here. This site is for historical information and is not intended to be used for any official purpose. For official data please contact the NMEA web site. Please see the bottom of this article for the sources of this data.

Introduction

The National Marine Electronics Association (NMEA) has developed a specification that defines the interface between various pieces of marine electronic equipment. The standard permits marine electronics to send information to computers and to other marine equipment. A full copy of this standard is available for purchase at their web site. None of the information on this site comes from this standard and I do not have a copy. Anyone attempting to design anything to this standard should obtain an official copy.

GPS receiver communication is defined within this specification. Most computer programs that provide real time position information understand and expect data to be in NMEA format. This data includes the complete PVT (position, velocity, time) solution computed by the GPS receiver. The idea of NMEA is to send a line of data called a sentence that is totally self contained and independent from other sentences. There are standard sentences for each device category and there is also the ability to define proprietary sentences for use by the individual company. All of the standard sentences have a two letter prefix that defines the device that uses that sentence type. (For gps receivers the prefix is GP.) which is followed by a three letter sequence that defines the sentence contents. In addition NMEA permits hardware manufactures to define their own proprietary sentences for whatever purpose they see fit. All proprietary sentences begin with the letter P and are followed with 3 letters that identifies the manufacturer controlling that sentence. For example a Garmin sentence would start with PGRM and Magellan would begin with PMGN.

Each sentence begins with a '$' and ends with a carriage return/line feed sequence and can be no longer than 80 characters of visible text (plus the line terminators). The data is contained within this single line with data items separated by commas. The data itself is just ascii text and may extend over multiple sentences in certain specialized instances but is normally fully contained in one variable length sentence. The data may vary in the amount of precision contained in the message. For example time might be indicated to decimal parts of a second or location may be show with 3 or even 4 digits after the decimal point. Programs that read the data should only use the commas to determine the field boundaries and not depend on column positions. There is a provision for a checksum at the end of each sentence which may or may not be checked by the unit that reads the data. The checksum field consists of a '*' and two hex digits representing the exclusive OR of all characters between, but not including, the '$' and '*'. A checksum is required on some sentences.

There have been several changes to the standard but for gps use the only ones that are likely to be encountered are 1.5 and 2.0 through 2.3. These just specify some different sentence configurations which may be peculiar to the needs of a particular device thus the gps may need to be changed to match the devices being interfaced to. Some gps's provide the ability configure a custom set the sentences while other may offer a set of fixed choices. Many gps receivers simply output a fixed set of sentences that cannot be changed by the user. The current version of the standard is 3.01. I have no specific information on this version, but I am not aware of any GPS products that require conformance to this version.

Hardware Connection

The hardware interface for GPS units is designed to meet the NMEA requirements. They are also compatible with most computer serial ports using RS232 protocols, however strictly speaking the NMEA standard is not RS232. They recommend conformance to EIA-422. The interface speed can be adjusted on some models but the NMEA standard is 4800 baud with 8 bits of data, no parity, and one stop bit. All units that support NMEA should support this speed. Note that, at a baud rate of 4800, you can easily send enough data to more than fill a full second of time. For this reason some units only send updates every two seconds or may send some data every second while reserving other data to be sent less often. In addition some units may send data a couple of seconds old while other units may send data that is collected within the second it is sent. Generally time is sent in some field within each second so it is pretty easy to figure out what a particular gps is doing. Some sentences may be sent only during a particular action of the receiver such as while following a route while other receivers may always send the sentence and just null out the values. Other difference will be noted in the specific data descriptions defined later in the text.

At 4800 baud you can only send 480 characters in one second. Since an NMEA sentence can be as long as 82 characters you can be limited to less than 6 different sentences. The actual limit is determined by the specific sentences used, but this shows that it is easy to overrun the capabilities if you want rapid sentence response. NMEA is designed to run as a process in the background spitting out sentences which are then captured as needed by the using program. Some programs cannot do this and these programs will sample the data stream, then use the data for screen display, and then sample the data again. Depending on the time needed to use the data there can easily be a lag of 4 seconds in the responsiveness to changed data.

The NMEA standard has been around for many years (1983) and has undergone several revisions. The protocol has changed and the number and types of sentences may be different depending on the revision. Most GPS receivers understand the standard which is called: 0183 version 2. This standard dictates a transfer rate of 4800 baud. Some receivers also understand older standards. The oldest standard was 0180 followed by 0182 which transferred data at 1200 baud. An earlier version of 0183 called version 1.5 is also understood by some receivers. Some Garmin units and other brands can be set to 9600 for NMEA output or even higher but this is only recommended if you have determined that 4800 works ok and then you can try to set it faster. Setting it to run as fast as you can may improve the responsiveness of the program.

In order to use the hardware interface you will need a cable. Generally the cable is unique to the hardware model so you will need an cable made specifically for the brand and model of the unit you own. Some of the latest computers no longer include a serial port but only a USB port. Most gps receivers will work with Serial to USB adapters and serial ports attached via the pcmcia (pc card) adapter. For general NMEA use with a gps receiver you will only need two wires in the cable, data out from the gps and ground. A third wire, Data in, will be needed if you expect the receiver to accept data on this cable such as to upload waypoints or send DGPS data to the receiver.

GPS receivers may be used to interface with other NMEA devices such as autopilots, fishfinders, or even another gps receivers. They can also listen to Differential Beacon Receivers that can send data using the RTCM SC-104 standard. This data is consistent with the hardware requirements for NMEA input data. There are no handshake lines defined for NMEA.

NMEA sentences

NMEA consists of sentences, the first word of which, called a data type, defines the interpretation of the rest of the sentence. Each Data type would have its own unique interpretation and is defined in the NMEA standard. The GGA sentence (shown below) shows an example that provides essential fix data. Other sentences may repeat some of the same information but will also supply new data. Whatever device or program that reads the data can watch for the data sentence that it is interested in and simply ignore other sentences that is doesn't care about. In the NMEA standard there are no commands to indicate that the gps should do something different. Instead each receiver just sends all of the data and expects much of it to be ignored. Some receivers have commands inside the unit that can select a subset of all the sentences or, in some cases, even the individual sentences to send. There is no way to indicate anything back to the unit as to whether the sentence is being read correctly or to request a re-send of some data you didn't get. Instead the receiving unit just checks the checksum and ignores the data if the checksum is bad figuring the data will be sent again sometime later.

There are many sentences in the NMEA standard for all kinds of devices that may be used in a Marine environment. Some of the ones that have applicability to gps receivers are listed below: (all message start with GP.)

- AAM - Waypoint Arrival Alarm

- ALM - Almanac data
- APA - Auto Pilot A sentence
- APB - Auto Pilot B sentence
- BOD - Bearing Origin to Destination
- BWC - Bearing using Great Circle route
- DTM - Datum being used.
- GGA - Fix information
- GLL - Lat/Lon data
- GSA - Overall Satellite data
- GSV - Detailed Satellite data
- MSK - send control for a beacon receiver
- MSS - Beacon receiver status information.
- RMA - recommended Loran data
- RMB - recommended navigation data for gps
- RMC - recommended minimum data for gps
- RTE - route message
- VTG - Vector track an Speed over the Ground
- WCV - Waypoint closure velocity (Velocity Made Good)
- WPL - Waypoint information
- XTC - cross track error
- XTE - measured cross track error
- ZTG - Zulu (UTC) time and time to go (to destination)
- ZDA - Date and Time

In addition some gps receivers with special capabilities output these special messages.

- HCHDG - Compass output
- PSLIB - Remote Control for a DGPS receiver

The last version 2 iteration of the NMEA standard was 2.3. It added a mode indicator to several sentences which is used to indicate the kind of fix the receiver currently has. This indication is part of the signal integrity information needed by the FAA. The value can be A=autonomous, D=differential, E=Estimated, N=not valid, S=Simulator. Sometimes there can be a null value as well. Only the A and D values will correspond to an Active and reliable Sentence. This mode character has been added to the RMC, RMB, VTG, and GLL, sentences and optionally some others including the BWC and XTE sentences.

If you are interfacing a GPS unit to another device, including a computer program, you need to ensure that the receiving unit is given all of the sentences that it needs. If it needs a sentence that your GPS does not send then the interface to that unit is likely to fail. Here is a Link for the needs of some typical programs. The sentences sent by some typical receivers include:

NMEA 2.0

Name	Garmin	Magellan	Lowrance	Notes:
GPAPB	N	Y	Y	Auto Pilot B
GPBOD	Y	N	N	bearing, origin to destination - earlier G-12's do not transmit this
GPGGA	Y	Y	Y	fix data
GPGLL	Y	Y	Y	Lat/Lon data - earlier G-12's do not transmit this
GPGSA	Y	Y	Y	overall satellite reception data, missing on some Garmin models
GPGSV	Y	Y	Y	detailed satellite data, missing on some Garmin models
GPRMB	Y	Y	Y	minimum recommended data when following a route
GPRMC	Y	Y	Y	minimum recommended data

GPRTE	Y	U	U	route data, only when there is an active route. (this is sometimes bidirectional)
GPWPL	Y	Y	U	waypoint data, only when there is an active route (this is sometimes bidirectional)

NMEA 1.5 - some units do not support version 1.5. Lowrance units provide the ability to customize the NMEA output by sentences so that you can develop your own custom sentence structure.

Name	Garmin	Magellan	Notes:
GPAPA	N	Y	Automatic Pilot A
GPBOD	Y	N	bearing origin to destination - earlier G-12's do not send this
GPBWC	Y	Y	bearing to waypoint using great circle route.
GPGLL	Y	Y	lat/lon - earlier G-12's do not send this
GPRMC	Y	N	minimum recommend data
GPRMB	Y	N	minimum recommended data when following a route
GPVTG	Y	Y	vector track and speed over ground
GPWPL	Y	N	waypoint data (only when active goto)
GPXTE	Y	Y	cross track error

The NMEA 2.3 output from the Garmin Legend, Vista, and perhaps some others include the BWC, VTG, and XTE sentences.

The Trimble Scoutmaster outputs: APA, APB, BWC, GGA, GLL, GSA, GSV, RMB, RMC, VTG, WCV, XTE, ZTG.

The Motorola Encore outputs: GGA, GLL, GSV, RMC, VTG, ZDA and a proprietary sentence PMOTG.

Units based on the SiRF chipset can output: GGA, GLL, GSA, GSV, RMC, and VTG. What is actually output is based on which sentences are selected by the user or application program. See below for more details. Some implementations have enhanced the SiRF capabilities with other sentences as well by changing the firmware. For example, the u-blox receivers add ZDA and some proprietary sentences to the above list of sentences. Check your documentation for more details.

Garmin receivers send the following Proprietary Sentences:

- PGRME (estimated error) - not sent if set to 0183 1.5
- PGRMM (map datum)
- PGRMZ (altitude)
- PSLIB (beacon receiver control)

Note that Garmin converts lat/lon coordinates to the datum chosen by the user when sending this data. This is indicated in the proprietary sentence PGRMM. This can help programs that use maps with other datums but is not an NMEA standard. Be sure and set your datum to WGS84 on Garmin units when communicating to other NMEA devices.

Magellan also converts lat/lon coordinates to the datum chosen on the receiver but do not indicate this in a message. Magellan units use proprietary sentences for waypoint maintenance and other tasks. They use a prefix of PMGN for this data.

Most other units always output NMEA messages in the WGS84 datum. Be sure and check the user documentation to be sure.

It is possible to just view the information presented on the NMEA interface using a simple terminal program. If the terminal program can log the session then you can build a history of the entire session into a file. More sophisticated logging programs can filter the messages to only certain sentences or only collect sentences at prescribed intervals. Some computer programs that provide real time display and logging actually save the log in an ascii format that can be viewed with a text editor or used independently from the program that generated it.

NMEA input

Some units also support an NMEA input mode. While not too many programs support this mode it does provide a standardized way to update or add waypoint and route data. Note that there is no handshaking or commands in NMEA mode so you just send the data in the correct sentence and the unit will accept the data and add or overwrite the information in memory. If the data is not in the correct format it will simply be ignored. A carriage return/line feed sequence is required. If the waypoint name is the same you will overwrite existing data but no warning will be issued. The sentence construction is identical to what the unit downloads so you can, for example, capture a WPL sentence from one unit and then send that same sentence to another unit but be careful if the two units support waypoint names of different lengths since the receiving unit might truncate the name and overwrite a waypoint accidently. If you create a sentence from scratch you should create a correct checksum. Be sure you know and have set you unit to the correct datum. Many units support the input of WPL sentences and a few support RTE as well.

On NMEA input the receiver stores information based on interpreting the sentence itself. While some receivers accept standard NMEA input this can only be used to update a waypoint or similar task and not to send a command to the unit. Proprietary input sentences could be used to send commands. Since the Magellan upload and download maintenance protocol is based on NMEA sentences they support a modified WPL message that adds comments, altitude, and icon data.

Some marine units may accept input for alarms such as deep or shallow water based on the DPT sentence or MTW to read the water temperature. For example the Garmin Map76 supports DPT, MTW (temperature), and VHW (speed) input sentences. Other units may use NMEA input to provide initialization data via proprietary sentences, or to select which NMEA sentences to output.

Decode of selected position sentences

The most important NMEA sentences include the GGA which provides the current Fix data, the RMC which provides the minimum gps sentences information, and the GSA which provides the Satellite status data.

GGA - essential fix data which provide 3D location and accuracy data.

```
$GPGGA,123519,4807.038,N,01131.000,E,1,08,0.9,545.4,M,46.9,M,,*47
```

```
Where:
     GGA          Global Positioning System Fix Data
     123519       Fix taken at 12:35:19 UTC
     4807.038,N   Latitude 48 deg 07.038' N
     01131.000,E  Longitude 11 deg 31.000' E
     1            Fix quality: 0 = invalid
                               1 = GPS fix (SPS)
                               2 = DGPS fix
                               3 = PPS fix
                          4 = Real Time Kinematic
                          5 = Float RTK
                               6 = estimated (dead reckoning) (2.3 feature)
                          7 = Manual input mode
                          8 = Simulation mode
     08           Number of satellites being tracked
     0.9          Horizontal dilution of position
     545.4,M      Altitude, Meters, above mean sea level
```

```
46.9,M       Height of geoid (mean sea level) above WGS84
                 ellipsoid
(empty field) time in seconds since last DGPS update
(empty field) DGPS station ID number
*47          the checksum data, always begins with *
```

If the height of geoid is missing then the altitude should be suspect. Some non-standard implementations report altitude with respect to the ellipsoid rather than geoid altitude. Some units do not report negative altitudes at all. This is the only sentence that reports altitude.

GSA - GPS DOP and active satellites. This sentence provides details on the nature of the fix. It includes the numbers of the satellites being used in the current solution and the DOP. DOP (dilution of precision) is an indication of the effect of satellite geometry on the accuracy of the fix. It is a unitless number where smaller is better. For 3D fixes using 4 satellites a 1.0 would be considered to be a perfect number, however for overdetermined solutions it is possible to see numbers below 1.0.

There are differences in the way the PRN's are presented which can effect the ability of some programs to display this data. For example, in the example shown below there are 5 satellites in the solution and the null fields are scattered indicating that the almanac would show satellites in the null positions that are not being used as part of this solution. Other receivers might output all of the satellites used at the beginning of the sentence with the null field all stacked up at the end. This difference accounts for some satellite display programs not always being able to display the satellites being tracked. Some units may show all satellites that have ephemeris data without regard to their use as part of the solution but this is non-standard.

```
  $GPGSA,A,3,04,05,,09,12,,,24,,,,,2.5,1.3,2.1*39
```

```
Where:
     GSA      Satellite status
     A        Auto selection of 2D or 3D fix (M = manual)
     3        3D fix - values include: 1 = no fix
                                       2 = 2D fix
                                       3 = 3D fix
     04,05... PRNs of satellites used for fix (space for 12)
     2.5      PDOP (dilution of precision)
     1.3      Horizontal dilution of precision (HDOP)
     2.1      Vertical dilution of precision (VDOP)
     *39      the checksum data, always begins with *
```

GSV - Satellites in View shows data about the satellites that the unit might be able to find based on its viewing mask and almanac data. It also shows current ability to track this data. Note that one GSV sentence only can provide data for up to 4 satellites and thus there may need to be 3 sentences for the full information. It is reasonable for the GSV sentence to contain more satellites than GGA might indicate since GSV may include satellites that are not used as part of the solution. It is not a requirment that the GSV sentences all appear in sequence. To avoid overloading the data bandwidth some receivers may place the various sentences in totally different samples since each sentence identifies which one it is.

The field called SNR (Signal to Noise Ratio) in the NMEA standard is often referred to as signal strength. SNR is an indirect but more useful value that raw signal strength. It can range from 0 to 99 and has units of dB according to the NMEA standard, but the various manufacturers send different ranges of numbers with different starting numbers so the values themselves cannot necessarily be used to evaluate different units. The range of working values in a given gps will usually show a difference of about 25 to 35 between the lowest and highest values, however 0 is a special case and may be shown on satellites that are in view but not being tracked.

```
  $GPGSV,2,1,08,01,40,083,46,02,17,308,41,12,07,344,39,14,22,228,45*75

Where:
      GSV          Satellites in view
      2            Number of sentences for full data
      1            sentence 1 of 2
      08           Number of satellites in view

      01           Satellite PRN number
      40           Elevation, degrees
      083          Azimuth, degrees
      46           SNR - higher is better
           for up to 4 satellites per sentence
      *75          the checksum data, always begins with *
```

RMC - NMEA has its own version of essential gps pvt (position, velocity, time) data. It is called RMC, The Recommended Minimum, which will look similar to:

```
$GPRMC,123519,A,4807.038,N,01131.000,E,022.4,084.4,230394,003.1,W*6A

Where:
     RMC          Recommended Minimum sentence C
     123519       Fix taken at 12:35:19 UTC
     A            Status A=active or V=Void.
     4807.038,N   Latitude 48 deg 07.038' N
     01131.000,E  Longitude 11 deg 31.000' E
     022.4        Speed over the ground in knots
     084.4        Track angle in degrees True
     230394       Date - 23rd of March 1994
     003.1,W      Magnetic Variation
     *6A          The checksum data, always begins with *
```

Note that, as of the 2.3 release of NMEA, there is a new field in the RMC sentence at the end just prior to the checksum. For more information on this field see here.

GLL - Geographic Latitude and Longitude is a holdover from Loran data and some old units may not send the time and data active information if they are emulating Loran data. If a gps is emulating Loran data they may use the LC Loran prefix instead of GP.

```
  $GPGLL,4916.45,N,12311.12,W,225444,A,*31

Where:
     GLL          Geographic position, Latitude and Longitude
```

```
4916.46,N    Latitude 49 deg. 16.45 min. North
12311.12,W   Longitude 123 deg. 11.12 min. West
225444       Fix taken at 22:54:44 UTC
A            Data Active or V (void)
*31          checksum data
```

Note that, as of the 2.3 release of NMEA, there is a new field in the GLL sentence at the end just prior to the checksum. For more information on this field see here.

VTG - Velocity made good. The gps receiver may use the LC prefix instead of GP if it is emulating Loran output.

```
$GPVTG,054.7,T,034.4,M,005.5,N,010.2,K
```

where:

```
VTG          Track made good and ground speed
054.7,T      True track made good
034.4,M      Magnetic track made good
005.5,N      Ground speed, knots
010.2,K      Ground speed, Kilometers per hour
```

Note that, as of the 2.3 release of NMEA, there is a new field in the VTG sentence at the end just prior to the checksum. For more information on this field see here.

Receivers that don't have a magnetic deviation (variation) table built in will null out the Magnetic track made good.

Decode of some Navigation Sentences

WPL - Waypoint Location data provides essential waypoint data. It is output when navigating to indicate data about the destination and is sometimes supported on input to redefine a waypoint location. Note that waypoint data as defined in the standard does not define altitude, comments, or icon data. When a route is active, this sentence is sent once for each waypoint in the route, in sequence. When all waypoints have been reported, the RTE sentence is sent in the next data set. In any group of sentences, only one WPL sentence, or an RTE sentence, will be sent.

```
$GPWPL,4807.038,N,01131.000,E,WPTNME*5C
```

With an interpretation of:

```
WPL          Waypoint Location
4807.038,N   Latitude
01131.000,E  Longitude
WPTNME       Waypoint Name
*5C          The checksum data, always begins with *
```

AAM - Waypoint Arrival Alarm is generated by some units to indicate the Status of arrival (entering the arrival circle, or passing the perpendicular of the course line) at the destination waypoint.

```
$GPAAM,A,A,0.10,N,WPTNME*43
```

```
Where:
    AAM    Arrival Alarm
    A      Arrival circle entered
    A      Perpendicular passed
    0.10   Circle radius
    N      Nautical miles
    WPTNME Waypoint name
    *43    Checksum data
```

APB - Autopilot format B is sent by some gps receivers to allow them to be used to control an autopilot unit. This sentence is commonly used by autopilots and contains navigation receiver warning flag status, cross-track-error, waypoint arrival status, initial bearing from origin waypoint to the destination, continuous bearing from present position to destination and recommended heading-to-steer to destination waypoint for the active navigation leg of the journey.

Note: some autopilots, Robertson in particular, misinterpret "bearing from origin to destination" as "bearing from present position to destination". This is likely due to the difference between the APB sentence and the APA sentence. for the APA sentence this would be the correct thing to do for the data in the same field. APA only differs from APB in this one field and APA leaves off the last two fields where this distinction is clearly spelled out. This will result in poor performance if the boat is sufficiently off-course that the two bearings are different.

```
  $GPAPB,A,A,0.10,R,N,V,V,011,M,DEST,011,M,011,M*82
```

```
where:
    APB     Autopilot format B
    A       Loran-C blink/SNR warning, general warning
    A       Loran-C cycle warning
    0.10    cross-track error distance
    R       steer Right to correct (or L for Left)
    N       cross-track error units - nautical miles (K for kilometers)
    V       arrival alarm - circle
    V       arrival alarm - perpendicular
    011,M   magnetic bearing, origin to destination
    DEST    destination waypoint ID
    011,M   magnetic bearing, present position to destination
    011,M   magnetic heading to steer (bearings could True as 033,T)
```

BOD - Bearing - Origin to Destination shows the bearing angle of the line, calculated at the origin waypoint, extending to the destination waypoint from the origin waypoint for the active navigation leg of the journey.

```
  $GPBOD,045.,T,023.,M,DEST,START*01
```

```
where:
        BOD          Bearing - origin to destination waypoint
        045.,T       bearing 045 True from "START" to "DEST"
```

```
        023.,M      bearing 023 Magnetic from "START" to "DEST"
        DEST        destination waypoint ID
        START       origin waypoint ID
        *01         checksum
```

BWC - Bearing & Distance to Waypoint using a Great Circle route. Time (UTC) and distance & bearing to, and location of, a specified waypoint from present position along the great circle path.

```
  $GPBWC,225444,4917.24,N,12309.57,W,051.9,T,031.6,M,001.3,N,004*29
```

where:

```
        BWC         Bearing and distance to waypoint - great circle
        225444      UTC time of fix 22:54:44
        4917.24,N   Latitude of waypoint
        12309.57,W  Longitude of waypoint
        051.9,T     Bearing to waypoint, degrees true
        031.6,M     Bearing to waypoint, degrees magnetic
        001.3,N     Distance to waypoint, Nautical miles
        004         Waypoint ID
        *29         checksum
```

RMB - The recommended minimum navigation sentence is sent whenever a route or a goto is active. On some systems it is sent all of the time with null data. The Arrival alarm flag is similar to the arrival alarm inside the unit and can be decoded to drive an external alarm. Note the use of leading zeros in this message to preserve the character spacing. This is done, I believe, because some autopilots may depend on exact character spacing.

```
  $GPRMB,A,0.66,L,003,004,4917.24,N,12309.57,W,001.3,052.5,000.5,V*20
```

where:

```
           RMB          Recommended minimum navigation information
           A            Data status A = OK, V = Void (warning)
           0.66,L       Cross-track error (nautical miles, 9.99 max),
                                steer Left to correct (or R = right)
           003          Origin waypoint ID
           004          Destination waypoint ID
           4917.24,N    Destination waypoint latitude 49 deg. 17.24 min. N
           12309.57,W   Destination waypoint longitude 123 deg. 09.57 min. W
           001.3        Range to destination, nautical miles (999.9 max)
           052.5        True bearing to destination
           000.5        Velocity towards destination, knots
           V            Arrival alarm  A = arrived, V = not arrived
           *20          checksum
```

RTE - RTE is sent to indicate the names of the waypoints used in an active route. There are two types of RTE sentences. This route sentence can list all of the waypoints in the entire route or it can list only those still ahead. Because an NMEA sentence is limited to 80 characters there may need to be multiple sentences to identify all of the waypoints. The data about the waypoints themselves will be sent in subsequent WPL sentences which will be sent in future cycles of the NMEA data.

```
$GPRTE,2,1,c,0,W3IWI,DRIVWY,32CEDR,32-29,32BKLD,32-I95,32-US1,BW-32,BW-198*69
```

Where:

RTE	Waypoints in active route
2	total number of sentences needed for full data
1	this is sentence 1 of 2
c	Type c = complete list of waypoints in this route
	w = first listed waypoint is start of current leg
0	Route identifier
W3IWI,...	Waypoint identifiers (names)
*69	checksum

XTE - Measured cross track error is a small subset of the RMB message for compatibility with some older equipment designed to work with Loran. Note that the same limitations apply to this message as the ones in the RMB since it is expected to be decoded by an autopilot.

```
$GPXTE,A,A,0.67,L,N*6F
```

Where:

XTE	Cross track error, measured
A	General warning flag V = warning
	(Loran-C Blink or SNR warning)
A	Not used for GPS (Loran-C cycle lock flag)
0.67	cross track error distance
L	Steer left to correct error (or R for right)
N	Distance units - Nautical miles
*6F	checksum

Other sentences that may be useful

ALM - GPS Almanac Data contains GPS week number, satellite health and the complete almanac data for one satellite. Multiple messages may be transmitted, one for each satellite in the GPS constellation, up to maximum of 32 messages. Note that these sentences can take a long time to send so they are not generally sent automatically by the gps receiver. (Sorry I don't have an exact example of the sentence.) Note that this sentence breaks the 80 character rule.

```
$GPALM,A.B,C.D,E,F,hh,hhhh,...
```

Where:

ALM	Almanac Data being sent
A	Total number of messages
B	Message number

```
C     Satellite PRN number
D     GPS week number (0-1023)
E     Satellite health (bits 17-24 of message)
F     eccentricity
hh    t index OA, almanac reference time
hhhh  sigma index 1, inclination angle
...   OMEGADOT rate of right ascension
      SQRA(A) root of semi-major axis
      Omega, argument of perigee
      Omega index 0, longitude of ascension node
      M index 0, mean anomaly
      a index f0, clock parameter
      a index f1, clock parameter
```

HCHDG - Compass output is used on Garmin etrex summit, vista , and 76S receivers to output the value of the internal flux-gate compass. Only the magnetic heading and magnetic variation is shown in the message.

```
  $HCHDG,101.1,,,7.1,W*3C

where:
     HCHDG    Magnetic heading, deviation, variation
     101.1    heading
     ,,       deviation (no data)
     7.1,W    variation
```

ZDA - Data and Time

```
  $GPZDA,hhmmss.ss,dd,mm,yyyy,xx,yy*CC
  $GPZDA,201530.00,04,07,2002,00,00*6E

where:
       hhmmss    HrMinSec(UTC)
         dd,mm,yyy Day,Month,Year
         xx        local zone hours -13..13
         yy        local zone minutes 0..59
         *CC       checksum
```

MSK - Control for a Beacon Receiver

```
  $GPMSK,318.0,A,100,M,2*45

where:
       318.0      Frequency to use
```

```
A          Frequency mode, A=auto, M=manual
100        Beacon bit rate
M          Bitrate, A=auto, M=manual
2          frequency for MSS message status (null for no status)
*45        checksum
```

MSS - Beacon Receiver Status

```
$GPMSS,55,27,318.0,100,*66
```

```
where:
       55         signal strength in dB
       27         signal to noise ratio in dB
       318.0      Beacon Frequency in KHz
       100        Beacon bitrate in bps
       *66        checksum
```

Proprietary Sentences

Proprietary sentences can either be output from the gps or used as input to control information. They always start with P which is followed by a 3 character manufactures code and additional characters to define the sentence type.

Garmin

The following are Garmin proprietary sentences. "P" denotes proprietary, "GRM" is Garmin's manufacturer code, and "M" or "Z" indicates the specific sentence type. Note that the PGRME sentence is not set if the output is set to NMEA 1.5 mode.

```
$PGRME,15.0,M,45.0,M,25.0,M*1C
```

```
where:
     15.0,M       Estimated horizontal position error in meters (HPE)
     45.0,M       Estimated vertical error (VPE) in meters
     25.0,M       Overall spherical equivalent position error
```

```
$PGRMZ,93,f,3*21
```

```
where:
      93,f         Altitude in feet
      3            Position fix dimensions 2 = user altitude
                                           3 = GPS altitude
   This sentence shows in feet, regardless of units shown on the display.
   Note that for units with an altimeter this will be altitude computed
   by the internal altimeter.
```

```
  $PGRMM,NAD27 Canada*2F
     Currently active horizontal datum
```

PSLIB

Proprietary sentences are used to control a Starlink differential beacon receiver. (Garmin's DBR is Starlink compatible as are many others.) When the GPS receiver is set to change the DBR frequency or baud rate, the "J" sentence is replaced (just once) by (for example): $PSLIB,320.0,200*59 to set the DBR to 320 KHz, 200 baud.

```
      $PSLIB,,,J*22   Status request
      $PSLIB,,,K*23   configuration request
```

These two sentences are normally sent together in each group of sentences from the GPS. The three fields are: Frequency, bit Rate, Request Type. The value in the third field may be: J = status request, K = configuration request, or null (blank) = tuning message. The correct values for frequency range from 283.5-325.0 KHz while the bit rate can be set to 0, 25, 50, 100 or 200 bps.

Magellan

Magellan uses proprietary sentences to do all of their waypoint and route maintenance. They use the MGN prefix for their sentences. This use is documented in their interface specification and will not be repeated here. However, they also send proprietary sentences to augment the gps data just like Garmin does. Here is an example of a sentence sent by the GPS Companion product:

```
  $PMGNST,02.12,3,T,534,05.0,+03327,00*40
```

```
where:
      ST      status information
      02.12   Version number?
      3       2D or 3D
      T       True if we have a fix False otherwise
      534     numbers change - unknown
      05.0    time left on the gps battery in hours
      +03327  numbers change (freq. compensation?)
      00      PRN number receiving current focus
      *40    checksum
```

A tracklog on a Meridian is made up of propretary sentences that look like:

```
$PMGNTRK,4322.061,N,07948.473,W,00116,M,173949.42,A,,020602*67
$PMGNTRK,4322.058,N,07948.483,W,00090,M,174202.45,A,,020602*69.
```

```
where

      TRK       Tracklog
      4322.071  Latitude
```

```
      N         North or South
      07948.473 Longitude
      W         East or West
      00116     Altitude
      M         Meters or Feet
      173949.42 UTC time
      A         Active or Void
      ,,        Track Name
      020602    date
      *67       checksum
```

Motorola

The **PMOTG** is used by Motorola Oncore receivers to send a command to the receiver. This command is used to set the output of the sentence to a particular frequency in seconds (or to 0) or to switch the output formula to motorola binary, gps, or loran.

```
  $PMOTG,xxx,yyyy
```

```
where:
      xxx    the sentence to be controlled
      yyyy   the time interval (0-9999 seconds)

or $PMOTG,FOR,y

where:
      y    MPB=0, GPS=1, Loran=2
```

Rockwell International

The Rockwell chipset is used on a number of gps receivers. It outputs some proprietary sentences with the **PRWI** prefix and accepts input from some special sentences similar to the approach used by Magellan. It can also be switched to a separate binary mode using a proprietary sentence. The input sentence most used to initialize the unit is $PRWIINIT and one output sentence is $PRWIRID

```
  $PRWIRID,12,01.83,12/15/97,0003,*42

where:
     $PRWIRID
     12         12 channel unit
     01.83      software version
     12/15/97   software date
     0003       software options (HEX value)
                Bit 0 minimize ROM usage
                Bit 1 minimize RAM usage
     *42        checksum
```

An input sentence that will define which NMEA sentences are to be output from the Rockwell unit is:

```
  $PRWIILOG,GGA,A,T,1,0
```

```
where:
   $PRWIILOG
   GGA         type of sentence
   A           A=activate, V=deactivate
   T           cyclic
   1           every 1 second
   0           ??
```

The initialization sentence which can be input to speed up acquisition looks like:

```
$PRWIINIT,V,,,4308.750,N,07159.791,W,100.0,0.0,M,0.0,T,175244,230503*77
```

```
where:
   $PRWIINIT     INIT = initialization
   V             V = reset, A = no reset
   ,,          Reserved for future use
   4308.750      Latitude
   N             N = North, S = South
   07159.791     Longitude
   W             W = West, E = East
   100.0         Altitude in meters
   0.0           Speed
   M             M = m/s, N = knots, K = km/hr
   0.0           Heading
   T             T = True, M = Magnetic
   175244      UTC time (hour, min, sec)
   230503        UTC date (day, month, year)
   *77           Checksum
```

Note: Commas may be used to signify using existing data. If units are supplied then the data must be present. Speed and direction must be supplied together. Lat/Lon must be supplied together. UTC time and date must be supplied together. If heading is magnetic then lat/lon needs to be supplied along with UTC time and date.

The sentences available for the Rockwell Jupiter chipset are: GGA, GSA, GSV, VTG, RMC and some proprietary sentences.

SiRF

The SiRF line of chips support several input sentences that permit the user to customize the way the chip behaves. In addition SiRF has a binary protocol that is even more powerful permitting different implementations to behave entirely differently. However, most applications do not attempt to customize the behavior so a user will need to make sure that the any customization is compatible with the application they are planning to use. There are 5 input sentences defined that begin with

$PSRF which is followed by three digits. Each sentence takes a fix amount of input fields which must exist, no null fields, and is terminated with the standard CR/LF sequence. The checksum is required.

The sentences 100 and 102 set the serial ports. 100 sets the main port A while 102 sets the DGPS input port B. 100 has an extra field that can be used to switch the interface to binary mode. Binary mode requires 8 bits, 1 stop bit, no parity. There is a command in binary mode that will switch the interface back to NMEA. Do not use the NMEA command to switch to binary mode unless you have the ability to switch it back. You could render your gps inoperative.

```
$PSRF100,0,9600,8,1,0*0C
$PSRF102,9600,8,1,0*3C
```

where

```
   $PSRF100
   0          0=SiRF, 1=NMEA  - This is where the protocol is changed.
   9600       baud rate 4800, 9600, 19200, 38400
   8          7, 8 Databits
   1          0, 1 Stopbits
   0          0=none, 1=odd, 2=even Parity
   *0C        checksum
```

The sentences 101 and 104 can be used to initialize values to be used by the gps. Supplying these values can shorten the initial lock time. If the clock offset is set to 0 then an internal default will be used. Sentence 101 supplies data in the internal ECEF (Earth centered, Earth Fixed) format in meters while sentence 104 supplies the data in the traditional Lat / Lon format.

```
$PSRF101,-2686700,-4304200,3851624,95000,497260,921,12,3*22
$PSRF104,37.3875111,-121.97232,0,95000,237759,922,12,3*3A
```

where

```
   $PSRF104
   37.3875111 Latitude in degrees
   -121.97232 Longitude in degrees
   0          Ellipsoid Altitude in meters
   95000      Clock offset
   237759     GPS Time of Week in seconds
   922        GPS Week Number
   12         Channel count (1 to 12)
   3          Reset config where
              1 = warm start, ephemeris valid
              2 = clear ephemeris, warm start (First Fix)
              3 = initialize with data, clear ephemeris
              4 = cold start, clear all data
              8 = cold start, set factory defaults
   *3A        checksum
```

The sentence 103 is used to control which NMEA sentences are to be sent and how often. Each sentence type is controlled individually. If the query bit is set then the gps responds by sending this message in the next second no matter what the rate is set to. Note that if trickle power is in use (can only be set in binary mode) then the actual update rate will be the selected update rate times the trickle rate which could mean that the data will be sent less frequently than was set here.

```
$PSRF103,05,00,01,01*20
```

```
where
   $PSRF103
   05          00=GGA
               01=GLL
               02=GSA
               03=GSV
               04=RMC
               05=VTG
   00          mode, 0=set rate, 1=query
   01          rate in seconds, 0-255
   01          checksum 0=no, 1=yes
   *20         checksum
```

The 105 sentence controls a debug mode which causes the gps to report any errors it finds with the input data. $PSRF105,1*3E would turn debug on while $PSRF105,0*3F would turn it off.

Magnavox

The old Magnavox system used mostly proprietary sentences. The Magnavox system was acquired by Leica Geosystems in 1994. Information on this system can be found at this site. The NMEA sentences themselves are described here. They all use the MVX prefix and include:

Control Port Input sentences

- $PMVXG,000 Initialization/Mode Control - Part A
- $PMVXG,001 Initialization/Mode Control - Part B
- $PMVXG,007 Control Port Configuration
- $PMVXG,023 Time Recovery Configuration
- $CDGPQ,YYY Query From a Remote Device / Request to Output a Sentence

Control Port Output Sentences

- $PMVXG,000 Receiver Status
- $PMVXG,021 Position, Height, Velocity
- $PMVXG,022 DOPs
- $PMVXG,030 Software Configuration
- $PMVXG,101 Control Sentence Accept/Reject
- $PMVXG,523 Time Recovery Configuration
- $PMVXG,830 Time Recovery Results

Sony

The Sony interface uses a proprietary sentence that looks like:

```
$PSNY,0,00,05,500,06,06,06,06*14
```

```
where
   PSNY
   0          Preamp (external antenna) status
              0 = Normal
              1 = Open
              2 = shorted
   00         Geodesic system (datum) 0-25, 0 = WGS84
   05        Elevation mask in degrees
   500       Speed Limit in Km
   06         PDOP limit with DGPS on
   06        HDOP limit with DGPS on
   06        PDOP limit with DGPS off
   06        HDOP limit with DGPS off
   *14       Checksum
```

Sample Streams

These streams will be modified when a route is active with the inclusion of route specific data.

Garmin

Garmin g12 sentences for version 4.57

```
$GPRMC,183729,A,3907.356,N,12102.482,W,000.0,360.0,080301,015.5,E*6F
$GPRMB,A,,,,,,,,,,,V*71
$GPGGA,183730,3907.356,N,12102.482,W,1,05,1.6,646.4,M,-24.1,M,,*75
$GPGSA,A,3,02,,,07,,09,24,26,,,,,1.6,1.6,1.0*3D
$GPGSV,2,1,08,02,43,088,38,04,42,145,00,05,11,291,00,07,60,043,35*71
$GPGSV,2,2,08,08,02,145,00,09,46,303,47,24,16,178,32,26,18,231,43*77
$PGRME,22.0,M,52.9,M,51.0,M*14
$GPGLL,3907.360,N,12102.481,W,183730,A*33
$PGRMZ,2062,f,3*2D
$PGRMM,WGS 84*06
$GPBOD,,T,,M,,*47
$GPRTE,1,1,c,0*07
$GPRMC,183731,A,3907.482,N,12102.436,W,000.0,360.0,080301,015.5,E*67
$GPRMB,A,,,,,,,,,,,V*71
```

Here are some observations:

- Notice the complete cycle shows an update interval of 2 seconds which is caused by the fact that there is too much data to fit in one second at 4800 baud.
- Upping the baud rate to 9600 will cause an update every second.
- Notice that the samples are in real time for each sentence because the GGA sentence shows an update in the time of 1 second.
- It would be possible to provide update data every second by parsing more sentences since the data is adjusted every second.
- Notice the gaps in the GSA message where the satellites in use are shown in a there slots as compared to the GSV locations. Some tools do not decode this configuration correctly.
- Note the GGA sentence starts the sequence every two seconds.
- This sample is similar for other Garmin receivers designed in the same time frame as the G-12.

Garmin etrex summit outputs

```
$GPRMC,002454,A,3553.5295,N,13938.6570,E,0.0,43.1,180700,7.1,W,A*3F
$GPRMB,A,,,,,,,,,,,A,A*0B
$GPGGA,002454,3553.5295,N,13938.6570,E,1,05,2.2,18.3,M,39.0,M,,*7F
$GPGSA,A,3,01,04,07,16,20,,,,,,,3.6,2.2,2.7*35
$GPGSV,3,1,09,01,38,103,37,02,23,215,00,04,38,297,37,05,00,328,00*70
$GPGSV,3,2,09,07,77,299,47,11,07,087,00,16,74,041,47,20,38,044,43*73
$GPGSV,3,3,09,24,12,282,00*4D
$GPGLL,3553.5295,N,13938.6570,E,002454,A,A*4F
$GPBOD,,T,,M,,*47
$PGRME,8.6,M,9.6,M,12.9,M*15
$PGRMZ,51,f*30
$HCHDG,101.1,,,7.1,W*3C
$GPRTE,1,1,c,*37
$GPRMC,002456,A,3553.5295,N,13938.6570,E,0.0,43.1,180700,7.1,W,A*3D
```

Some observations as compared to the G-12:

- Information is buffered. It is all for the same second.
- Information is only updated every two seconds at 4800 baud.
- Lat/Lon numbers have an extra digit.
- This is NMEA 2.3 data as indicated by the extra A at the end of RMC, RMB and GLL.
- Note that the satellites in use have been shoved to the left of the GSA message instead of the slot location.
- The RMC sentence starts the sequence.
- Note the HCHDG sentence for the built in compass.
- Except for the compass output this sentence list is similar for most Garmin units designed around the time of the Summit receivers, beginning with the emap.

Garmin etrex Vista release 2.42 outputs

```
$GPRMC,023042,A,3907.3837,N,12102.4684,W,0.0,156.1,131102,15.3,E,A*36
$GPRMB,A,,,,,,,,,,,A,A*0B
$GPGGA,023042,3907.3837,N,12102.4684,W,1,04,2.3,507.3,M,-24.1,M,,*75
$GPGSA,A,3,04,05,,,09,,,24,,,,,2.8,2.3,1.0*36
$GPGSV,3,2,11,09,47,229,42,10,04,157,00,14,00,305,00,24,70,154,33*79
$GPGLL,3907.3837,N,12102.4684,W,023042,A,A*5E
$GPBOD,,T,,M,,*47
```

```
$GPVTG,156.1,T,140.9,M,0.0,N,0.0,K*41
$PGRME,8.4,M,23.8,M,25.7,M*2B
$PGRMZ,1735,f*34
$PGRMM,WGS 84*06
$HCHDG,,,,15.3,E*30
$GPRTE,1,1,c,*37
$GPRMC,023044,A,3907.3840,N,12102.4692,W,0.0,156.1,131102,15.3,E,A*37
```

Some observations as compared to the Summit:

- Output still repeats at a rate of once every 2 seconds and is NMEA 2.3 Data
- The satellite status sentences are interleaved. This GSV sentences are only sent one in each two second group. Note the example shows sentence two of three. Thus the complete cycle would take 6 seconds.
- New sentence VTG.
- The altitude in PGRMZ is from the altimeter while the altitude in the GGA is from the gps computation.
- Note the HCHDG sentence for the built in compass and is missing for the Legend.

Garmin basic yellow etrex European version

```
$GPRMC,152926,V,6027.8259,N,02225.6713,E,10.8,0.0,190803,5.9,E,S*22
$GPRMB,V,,,,,,,,,,,A,S*0E
$GPGGA,152926,6027.8259,N,02225.6713,E,8,09,2.0,44.7,M,20.6,M,,*79
$GPGSA,A,3,07,08,09,11,18,23,26,28,29,,,,6.6,2.0,3.0*38
$GPGSV,3,1,09,07,29,138,44,08,22,099,42,09,30,273,44,11,07,057,35*75
$GPGSV,3,2,09,18,28,305,43,23,14,340,39,26,64,222,49,28,60,084,49*7E
$GPGSV,3,3,09,29,52,187,48*4E
$GPGLL,6027.8259,N,02225.6713,E,152926,V,S*48
$GPBOD,,T,,M,,*47
$PGRME,15.0,M,22.5,M,15.0,M*1B
$PGRMZ,147,f,3*19
$GPRTE,1,1,c,*37
$GPRMC,152928,V,6027.8319,N,02225.6713,E,10.8,0.0,190803,5.9,E,S*29
```

Some Ovservations:

- The sentence sequence starts with RMC and repeats every 2 seconds.
- The PGRMM sentence is missing so the datum is not identified.

Magellan

Magellan GPS companion sentences

```
$GPGGA,184050.84,3907.3839,N,12102.4772,W,1,05,1.8,00543,M,,,,*33
$GPRMC,184050.84,A,3907.3839,N,12102.4772,W,00.0,000.0,080301,15,E*54
$GPGSA,A,3,24,07,09,26,05,,,,,,,,03.6,01.8,03.1*05
$PMGNST,02.12,3,T,534,05.0,+03327,00*40
$GPGLL,3907.3839,N,12102.4771,W,184051.812,A*2D
$GPGGA,184051.81,3907.3839,N,12102.4771,W,1,05,1.8,00543,M,,,,*34
$GPRMC,184051.81,A,3907.3839,N,12102.4771,W,00.0,000.0,080301,15,E*53
$GPGSA,A,3,24,07,09,26,05,,,,,,,,03.6,01.8,03.1*05
```

```
$GPGSV,3,1,08,07,57,045,43,09,48,303,48,04,44,144,,02,39,092,*7F
$GPGSV,3,2,08,24,18,178,44,26,17,230,41,05,13,292,43,08,01,147,*75
$GPGSV,3,3,08,,,,,,,,,,,,,,,,*71
$GPGLL,3907.3840,N,12102.4770,W,184052.812,A*21
```

Some observations:

- Complete cycle takes two seconds.
- RMC, GGA, GSA, and GLL are update every second.
- GSV data is swapped with MGNST data every other second.
- Time is shown to .xx and for GLL .xxx precision but the unit output is not that accurate. Data seems asynchronous and not tied to top of any particular second.
- Lat/Lon has an extra digit as compared to the Garmin G-12.
- There is a third GSV sentence that is technically not required.
- Notice that all the satellites used are shoved to the left in the GSA message.
- No geoid corrections are shown in the GGA message. This indicates that altitude is shown with respect to the ellipsoid instead of MSL.

Magellan 315 shown in simulation mode.

```
$GPAPB,A,A,0.0,L,N,,,1.1,M,SIM002,1.1,M,,,*21
$GPGSA,A,3,01,02,03,04,,,,,,,,,2.0,2.0,2.0*34
$GPGSV,3,1,11,01,77,103,,13,53,215,,04,47,300,,20,47,090,*76
$GPGSV,3,2,11,19,24,158,,07,21,237,,25,16,039,,24,11,315,*73
$GPGSV,3,3,11,11,08,149,,27,00,179,,30,00,354,,,,,*46
$GPGLL,5100.2111,N,00500.0006,E,104715.203,A*37
$GPGGA,104715.20,5100.2111,N,00500.0006,E,1,04,2.0,-0047,M,,,,*39
$GPRMB,A,0.00,L,SIM001,SIM002,5102.6069,N,00500.0000,E,002.4,000.,021.7,V*0D
$GPRMC,104715.20,A,5100.2111,N,00500.0006,E,21.7,003.0,140801,01.,W*70
$GPAPB,A,A,0.0,L,N,,,1.1,M,SIM002,1.1,M,,,*21
$GPGSA,A,3,01,02,03,04,,,,,,,,,2.0,2.0,2.0*34
```

Some observations:

- This listing shows navigation sentences simulating a route between two locations, SIM001 and SIM002.
- GLL starts the sequence and time stamp in the GLL message shows more precision.
- Update is every 2 seconds.
- NMEA data is only transmitted in simulation mode or you have an actual fix.

Others

Raytheon RN300 sentences:

```
$GPGGA,171537,3350.975,N,11823.991,W,2,07,1.1,-25.8,M,,M,1.8,,D*17
$GPGLL,3350.975,N,11823.991,W,171537,A,D*50
$GPRMC,171537,A,3350.975,N,11823.991,W,0.0,096.5,060401,013.0,E,D*07
$GPVTG,096.5,T,083.5,M,0.0,N,0.0,K,D*22
$GPGSA,A,2,04,09,07,24,02,05,26,,,,,,1.1,*3C
$GPGSV,2,1,07,04,62,120,47,09,52,292,53,07,42,044,41,24,38,179,45*7B
$GPGSV,2,2,07,02,34,101,43,05,18,304,40,26,09,223,36,,,,*48
$PRAYA,6,1,122,0,0,2,36,1,1,,,,*5A
```

```
$GPDTM,W84,,0.000000,N,0.000000,E,0.0,W84*6F
$GPGGA,171538,3350.974,N,11823.991,W,2,07,1.1,-25.8,M,,M,1.8,,D*19
```

Some observations:

- Complete cycle every second triggered off of GGA.
- Date is NMEA 2.3 with integrity value added.
- The proprietary raytheon sentences seems to be for WAAS SV #122.
- Note the new DTM sentences that permits conversion of NMEA datum being used to WGS84.
- The satellites are listed in an arbitrary order, stacked to the left.

NavMan 3400 (SiRF chipset sentences)

```
$GPGGA,230611.016,3907.3813,N,12102.4635,W,0,04,5.7,507.9,M,,,,0000*11
$GPGLL,3907.3813,N,12102.4635,W,230611.016,V*31
$GPGSA,A,1,27,08,28,13,,,,,,,,,21.7,5.7,20.9*38
$GPGSV,3,1,10,27,68,048,42,08,63,326,43,28,48,239,40,13,39,154,39*7E
$GPGSV,3,2,10,31,38,069,34,10,23,282,,03,12,041,,29,09,319,*7C
$GPGSV,3,3,10,23,07,325,,01,05,145,*7E
$GPRMC,230611.016,V,3907.3813,N,12102.4635,W,0.14,136.40,041002,,*04
$GPVTG,136.40,T,,M,0.14,N,0.3,K*66
$GPGGA,230612.015,3907.3815,N,12102.4634,W,0,04,5.7,508.3,M,,,,0000*13
```

Some observations:

- A cycle is every second triggered off of GGA.
- The GSA, GSV sentences are only sent every 4 seconds or so. The actual sentences and rate is adjustable using proprietary NMEA commands.
- Altitude is based on the ellipsoid model and is not corrected for geoid. Note that no geoid corrections are shown in GGA.
- All headings are stated as true direction. There are no magnetic direction outputs.
- The ,0000 at the end of GGA is non standard.
- Lat/Lon has an extra digit as compared to the Garmin G-12.
- The clock is shown with millisecond precision.
- The Navman sends 10 lines of non-nmea ascii data when it is first turned on. Each line does begin with a $.
- This is a sample sentence sequence. The Navman can be programmed to send less sentences or sentences at a different rate.
- The Navman uses the SiRF chipset, see above for more data on this chipset.
- Sentences are stated to be NMEA 2.2 based on documentation.

Earhmate with SiRF chipset (firmware 2.31)

```
$GPGGA,120557.916,5058.7456,N,00647.0515,E,2,06,1.7,108.5,M,47.6,M,1.5,0000*7A
$GPGSA,A,3,20,11,25,01,14,31,,,,,,,2.6,1.7,1.9*3B
$GPGSV,2,1,08,11,74,137,45,20,58,248,43,07,27,309,00,14,23,044,36*7A
$GPGSV,2,2,08,01,14,187,41,25,13,099,39,31,11,172,37,28,09,265,*71
$GPRMC,120557.916,A,5058.7456,N,00647.0515,E,0.00,82.33,220503,,*39
$GPGGA,120558.916,5058.7457,N,00647.0514,E,2,06,1.7,109.0,M,47.6,M,1.5,0000*71
```

Some observations in comparison with the NavMan.

- This unit show WAAS/EGNOS (WADGPS) in use. The GGA sentence shows a 2 indicating differential gps corrections. The 1.5 at the end shows the age of the dgps correction signal.

- This is a new chipset firmware release and does support Geoid height in the altitude as shown in the GGA sentence.
- The RMC sentences shows that there is no support for Magnetic headings.
- When WAAS/EGNOS was not in use a GLL sentence showed up after the GGA.

Evermore GM-305

```
$GPGGA,001430.003,3907.3885,N,12102.4767,W,1,05,02.1,00545.6,M,-26.0,M,,*5F
$GPGSA,A,3,15,18,14,,,31,,,23,,,,04.5,02.1,04.0*0F
$GPGSV,3,1,10,15,48,123,35,18,36,064,36,14,77,186,39,03,36,239,29*7A
$GPGSV,3,2,10,09,08,059,,31,35,276,35,17,10,125,,11,08,306,*79
$GPGSV,3,3,10,23,41,059,37,25,06,173,*70
$GPRMC,001430.003,A,3907.3885,N,12102.4767,W,000.0,175.3,220403,015.4,E*71
$GPGGA,001431.003,3907.3885,N,12102.4767,W,1,05,02.1,00545.5,M,-26.0,M,,*5D
```

Some observations

- This chipset is used in the Deluo universal mouse gps.
- Update is every second by default.
- Actual sentences are programmable using proprietary interface. GLL and VTG can be added and others removed. The update interval can be modified.
- Altitude is given relative to MSL (Geoid height) in GGA
- Magnetic and True headings are supported.

Sony

```
$GPVTG,139.7,T,,M,010.3,N,019.1,K*67
$GPGGA,050306,4259.8839,N,07130.3922,W,0,00,99.9,0010,M,,M,000,0000*66
$GPGLL,4259.8839,N,07130.3922,W,050306,V*20
$GPRMC,050306,V,4259.8839,N,07130.3922,W,010.3,139.7,291003,,*10
$GPZDA,050306,29,10,2003,,*43
$GPGSA,A,1,,,,,,,,,,,,,99.9,99.9,99.9*09
$PSNY,0,00,05,500,06,06,06,06*14
```

- This is the format of Digittraveler from RadioShack.
- If batteries are removed for 5 minutes on the Digitraveler the data is wrong.
- The Sony proprietary message is described above.
- Altitude is Ellipsoid, not MSL.
- Heading is True only, Magnetic variation is not provided.
- VTG, GGA, GLL, RMC, ZDA output every second. GSA and PSNY are alternated with GSV data.

Credits

Peter's and Joe's web sites were used as primary sources for data in this article as well as personal research. Some data was obtained from the Garmin product manuals and product manuals from other manufacturers. The sample data streams were collected as captured from the appropriate devices directly or supplied to me by someone who captured them.

While I didn't use this page as a source there is some good data on Glenn Baddeley's site. It includes some sentences that are not on this page.

Dale DePriest

ANEXO 4
SEPARADOR DE SINCRONISMOS

April 2001

National Semiconductor

LM1881
Video Sync Separator

General Description

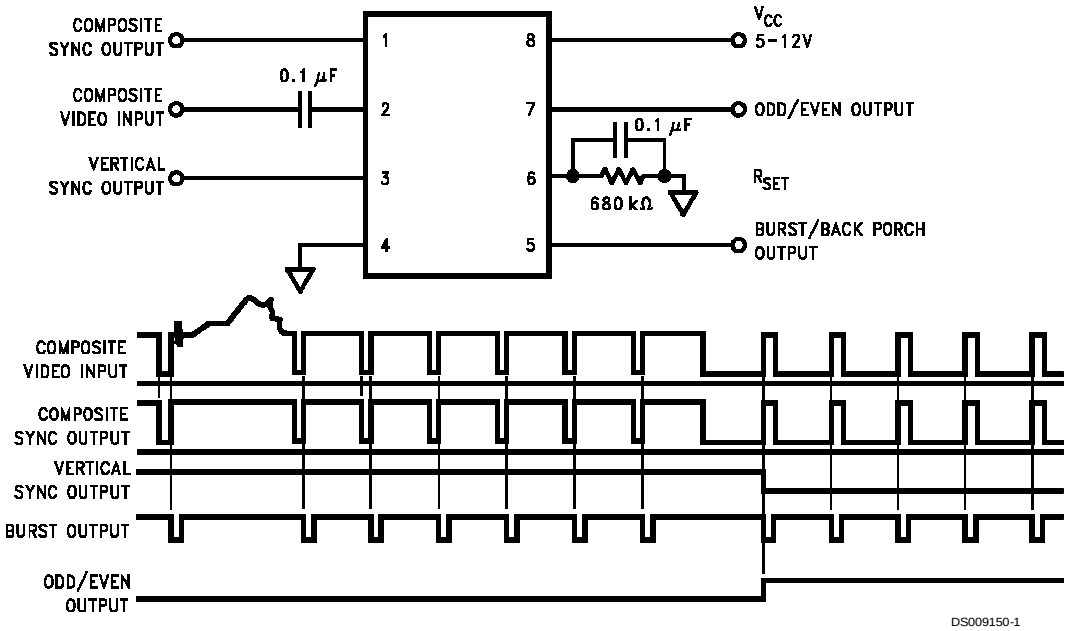
The LM1881 Video sync separator extracts timing information including composite and vertical sync, burst/back porch timing, and odd/even field information from standard negative going sync NTSC, PAL* and SECAM video signals with amplitude from 0.5V to 2V p-p. The integrated circuit is also capable of providing sync separation for non-standard, faster horizontal rate video signals. The vertical output is produced on the rising edge of the first serration in the vertical sync period. A default vertical output is produced after a time delay if the rising edge mentioned above does not occur within the externally set delay period, such as might be the case for a non-standard video signal.

Features

- AC coupled composite input signal
- >10 kΩ input resistance
- <10 mA power supply drain current
- Composite sync and vertical outputs
- Odd/even field output
- Burst gate/back porch output
- Horizontal scan rates to 150 kHz
- Edge triggered vertical output
- Default triggered vertical output for non-standard video signal (video games-home computers)

Connection Diagram

LM1881N



DS009150-1

Order Number LM1881M or LM1881N
See NS Package Number M08A or N08E

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	13.2V
Input Voltage	3 V_{P-P} (V_{CC} = 5V) 6 V_{P-P} ($V_{CC} \geq$ 8V)
Output Sink Currents; Pins, 1, 3, 5	5 mA
Output Sink Current; Pin 7	2 mA
Package Dissipation (Note 2)	1100 mW
Operating Temperature Range	0˚C–70˚C
Storage Temperature Range	−65˚C to +150˚C
ESD Susceptibility (Note 3)	2 kV
Soldering Information	
Dual-In-Line Package (10 sec.)	260˚C
Small Outline Package	
Vapor Phase (60 sec.)	215˚C
Infrared (15 sec.)	220˚C

See AN-450 “Surface Mounting Methods and their Effect on Product Reliability” for other methods of soldering surface mount devices.

Electrical Characteristics

V_{CC} = 5V; R_{SET} = 680 kΩ; T_A = 25˚C; Unless otherwise specified

Parameter	Conditions		Typ	Tested Limit (Note 4)	Design Limit (Note 5)	Units (Limits)
Supply Current	Outputs at Logic 1	V_{CC} = 5V V_{CC} = 12V	5.2 5.5	10 12		mAmax mAmax
DC Input Voltage	Pin 2		1.5	1.3 1.8		Vmin Vmax
Input Threshold Voltage	(Note 6)		70	55 85		mVmin mVmax
Input Discharge Current	Pin 2; V_{IN} = 2V		11	6 16		µAmin µAmax
Input Clamp Charge Current	Pin 2; V_{IN} = 1V		0.8	0.2		mAmin
R_{SET} Pin Reference Voltage	Pin 6; (Note 7)		1.22	1.10 1.35		Vmin Vmax
Composite Sync. & Vertical Outputs	I_{OUT} = 40 µA; Logic 1	V_{CC} = 5V V_{CC} = 12V	4.5	4.0 11.0		Vmin Vmin
	I_{OUT} = 1.6 mA Logic 1	V_{CC} = 5V V_{CC} = 12V	3.6	2.4 10.0		Vmin Vmin
Burst Gate & Odd/Even Outputs	I_{OUT} = 40 µA; Logic 1	V_{CC} = 5V V_{CC} = 12V	4.5	4.0 11.0		Vmin Vmin
Composite Sync. Output	I_{OUT} = −1.6 mA; Logic 0; Pin 1		0.2	0.8		Vmax
Vertical Sync. Output	I_{OUT} = −1.6 mA; Logic 0; Pin 3		0.2	0.8		Vmax
Burst Gate Output	I_{OUT} = −1.6 mA; Logic 0; Pin 5		0.2	0.8		Vmax
Odd/Even Output	I_{OUT} = −1.6 mA; Logic 0; Pin 7		0.2	0.8		Vmax
Vertical Sync Width			230	190 300		µsmin µsmax
Burst Gate Width	2.7 kΩ from Pin 5 to V_{CC}		4	2.5 4.7		µsmin µsmax
Vertical Default Time	(Note 8)		65	32 90		µsmin µsmax

Note 1: Absolute Maximum Ratings indicate limits beyond which damage to the device may occur. Operating Ratings indicate conditions for which the device is functional, but do not guarantee specific performance limits. For guaranteed specifications and test conditions, see the Electrical Characteristics. The guaranteed specifications apply only for the test conditions listed. Some performance characteristics may degrade when the device is not operated under the listed test conditions.

Note 2: For operation in ambient temperatures above 25˚C, the device must be derated based on a 150˚C maximum junction temperature and a package thermal resistance of 110˚C/W, junction to ambient.

Note 3: ESD susceptibility test uses the “human body model, 100 pF discharged through a 1.5 kΩ resistor”.

Note 4: Typicals are at T_J = 25˚C and represent the most likely parametric norm.

Note 5: Tested Limits are guaranteed to National’s AOQL (Average Outgoing Quality Level).

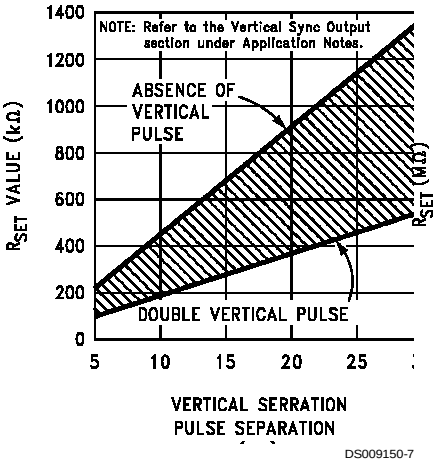
Note 6: Relative difference between the input clamp voltage and the minimum input voltage which produces a horizontal output pulse.

Note 7: Careful attention should be made to prevent parasitic capacitance coupling from any output pin (Pins 1, 3, 5 and 7) to the R_{SET} pin (Pin 6).

Note 8: Delay time between the start of vertical sync (at input) and the vertical output pulse.

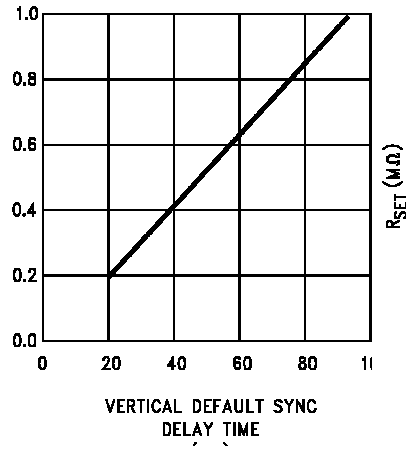
Typical Performance Characteristics

R_{SET} Value Selection vs Vertical Serration Pulse Separation



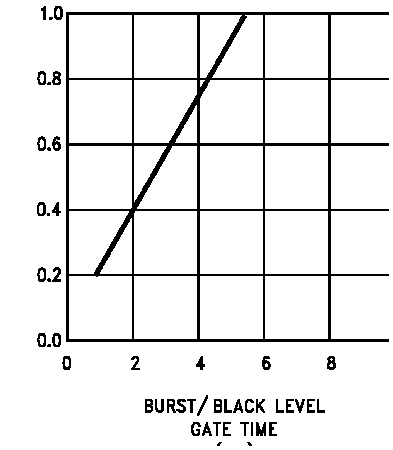
DS009150-7

Vertical Default Sync Delay Time vs R_{SET}



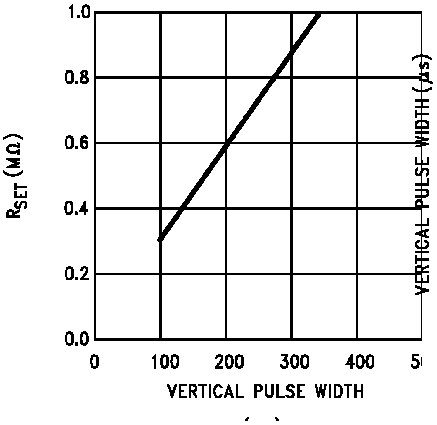
DS009150-8

Burst/Black Level Gate Time vs R_{SET}



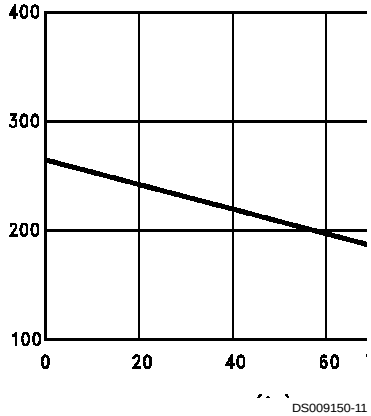
DS009150-9

Vertical Pulse Width vs R_{SET}



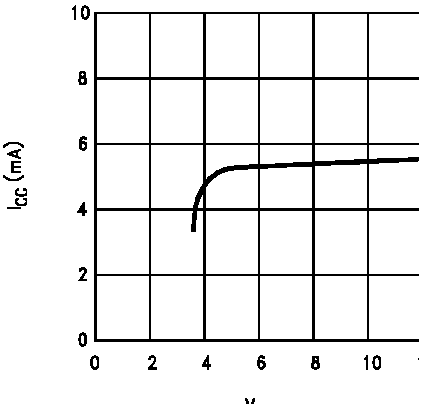
DS009150-10

Vertical Pulse Width vs Temperature



DS009150-11

Supply Current vs Supply Voltage



DS009150-2

Application Notes

The LM1881 is designed to strip the synchronization signals from composite video sources that are in, or similar to, the N.T.S.C. format. Input signals with positive polarity video (increasing signal voltage signifies increasing scene brightness) from 0.5V (p-p) to 2V (p-p) can be accommodated. The LM1881 operates from a single supply voltage between 5V DC and 12V DC. The only required external components besides a power supply decoupling capacitor at pin 8 and a set current decoupling capacitor at pin 6, are the composite input coupling capacitor at pin 2 and one resistor at pin 6 that sets internal current levels. The resistor on pin 6 (i.e. R_{set}) allows the LM1881 to be adjusted for source signals with line scan frequencies differing from 15.734 kHz. Four major sync signals are available from the I/C; composite sync including both horizontal and vertical scan timing information; a vertical sync pulse; a burst gate or back porch clamp pulse; and an odd/even output. The odd/even output level identifies which video field of an interlaced video source is present at the input. The outputs from the LM1881 can be used to gen-lock video camera/VTR signals with graphics sources, provide identification of video fields for memory storage, recover suppressed or contaminated sync signals, and provide timing references for the extraction of coded or uncoded data on specific video scan lines.

To better understand the LM1881 timing information and the type of signals that are used, refer to Figure 1(a-e) which shows a portion of the composite video signal from the end of one field through the beginning of the next field.

COMPOSITE SYNC OUTPUT

The composite sync output, Figure 1(b), is simply a reproduction of the signal waveform below the composite video black level, with the video completely removed. This is obtained by clamping the video signal sync tips to 1.5V DC at Pin 2 and using a comparator threshold set just above this voltage to strip the sync signal, which is then buffered out to Pin 1. The threshold separation from the clamped sync tip is nominally 70 mV which means that for the minimum input level of 0.5V (p-p), the clipping level is close to the halfway point on the sync pulse amplitude (shown by the dashed line

Application Notes (Continued)

on Figure 1(a). This threshold separation is independent of the signal amplitude, therefore, for a 2V (p-p) input the clipping level occurs at 11% of the sync pulse amplitude. The charging current for the input coupling capacitor is 0.8 mA,

Normally the signal source for the LM1881 is assumed to be clean and relatively noise-free, but some sources may have excessive video peaking, causing high frequency video and chroma components to extend below the black level reference. Some video discs keep the chroma burst pulse present throughout the vertical blanking period so that the burst actually appears on the sync tips for three line periods instead of at black level. A clean composite sync signal can be generated from these sources by filtering the input signal. When the source impedance is low, typically 75Ω, a 620Ω resistor in series with the source and a 510 pF capacitor to ground will form a low pass filter with a corner frequency of 500 kHz. This bandwidth is more than sufficient to pass the sync pulse portion of the waveform; however, any subcarrier content in the signal will be attenuated by almost 18 dB, effectively taking it below the comparator threshold. Filtering will also help if the source is contaminated with thermal noise. The output waveforms will become delayed from between 40 ns to as much as 200 ns due to this filter. This much delay will not usually be significant but it does contribute to the sync delay produced by any additional signal processing. Since the original video may also undergo processing, the need for time delay correction will depend on the total system, not just the sync stripper.

VERTICAL SYNC OUTPUT

A vertical sync output is derived by internally integrating the composite sync waveform (Figure 2). To understand the generation of the vertical sync pulse, refer to the lower left hand section Figure 2. Note that there are two comparators in the section. One comparator has an internally generated voltage reference called V_1 going to one of its inputs. The other comparator has an internally generated voltage reference called V_2 going to one of its inputs. Both comparators have a common input at their noninverting input coming from the internal integrator. The internal integrator is used for integrating the composite sync signal. This signal comes from the input side of the composite sync buffer and are **positive** going sync pulses. The capacitor to the integrator is internal to the LM1881. The capacitor charge current is set by the value of the external resistor R_{SET}. The output of the integrator is going to be at a low voltage during the normal horizontal lines because the integrator has a very short time to charge the capacitor, which is during the horizontal sync period. The equalization pulses will keep the output voltage of the integrator at about the same level, below the V_1. During the vertical sync period the narrow going positive pulses shown in Figure 1 is called the serration pulse. The wide negative portion of the vertical sync period is called the vertical sync pulse. At the start of the vertical sync period, before the first Serration pulse occurs, the integrator now charges the capacitor to a much higher voltage. At the first serration pulse the integrator output should be between V_1 and V_2. This would give a high level at the output of the comparator with V_1 as one of its inputs. This high is clocked into the "D" flip-flop by the falling edge of the serration pulse (remember the sync signal is inverted in this section of the LM1881). The "Q" output of the "D" flip-flop goes through the OR gate, and sets the R/S flip-flop. The output of the R/S flip-flop enables the internal oscillator and also clocks the ODD/EVEN "D" flip-flop. The ODD/EVEN field pulse operation is covered in the next section. The output of the oscillator goes to a divide by 8 circuit, thus resetting the R/S flip-flop after 8 cycles of the oscillator. The frequency of the oscillator is established by the internal capacitor going to the oscillator and the external R_{SET}. The "$\overline{Q}$" output of the R/S flip-flop goes to pin 3 and is the actual vertical sync output of the LM1881. By clocking the "D" flip-flop at the start of the first serration pulse means that the vertical sync output pulse starts at this point in time and lasts for eight cycles of the internal oscillator as shown in Figure 1.

How R_{SET} affects the integrator and the internal oscillator is shown under the Typical Performance Characteristics. The first graph is "R_{SET} Value Selection vs Vertical Serration Pulse Separation". For this graph to be valid, the vertical sync pulse should last for at least 85% of the horizontal half line (47% of a full horizontal line). A vertical sync pulse from any standard should meet this requirement; both NTSC and PAL do meet this requirement (the serration pulse is the remainder of the period, 10% to 15% of the horizontal half line). Remember this pulse is a positive pulse at the integrator but negative in Figure 1. This graph shows how long it takes the integrator to charge its internal capacitor above V_1.

With R_{SET} too large the charging current of the integrator will be too small to charge the capacitor above V_1, thus there will be no vertical synch output pulse. As mentioned above, R_{SET} also sets the frequency of the internal oscillator. If the oscillator runs too fast its eight cycles will be shorter than the vertical sync portion of the composite sync. Under this condition another vertical sync pulse can be generated on one of the later serration pulse after the divide by 8 circuit resets the R/S flip-flop. The first graph also shows the minimum R_{SET} necessary to prevent a double vertical pulse, assuming that the serration pulses last for only three full horizontal line periods (six serration pulses for NTSC). The actual pulse width of the vertical sync pulse is shown in the "Vertical Pulse Width vs R_{SET}" graph. Using NTSC as an example, lets see how these two graphs relate to each other. The Horizontal line is 64 µs long, or 32 µs for a horizontal half line. Now round this off to 30 µs. In the "R_{SET} Value Selection vs Vertical Serration Pulse Separation" graph the minimum resistor value for 30 µs serration pulse separation is about 550 kΩ. Going to the "Vertical Pulse Width vs R_{SET}" graph one can see that 550 kΩ gives a vertical pulse width of about 180 µs, the total time for the vertical sync period of NTSC (3 horizontal lines). A 550 kΩ will set the internal oscillator to a frequency such that eight cycles gives a time of 180 µs, just long enough to prevent a double vertical sync pulse at the vertical sync output of the LM1881.

The LM1881 also generates a default vertical sync pulse when the vertical sync period is unusually long and has no serration pulses. With a very long vertical sync time the integrator has time to charge its internal capacitor above the voltage level V_2. Since there is no falling edge at the end of a serration pulse to clock the "D" flip-flop, the only high signal going to the OR gate is from the default comparator when output of the integrator reaches V_2. At this time the R/S flip-flop is toggled by the default comparator, starting the vertical sync pulse at pin 3 of the LM1881. If the default vertical sync period ends before the end of the input vertical sync period, then the falling edge of the vertical sync (positive pulse at the "D" flip-flop) will clock the high output from the comparator with V_1 as a reference input. This will retrigger the oscillator, generating a second vertical sync output pulse. The "Vertical Default Sync Delay Time vs R_{SET}" graph shows the relationship between the R_{SET} value and the delay time from the start of the vertical sync period before the default vertical sync pulse is generated. Using the NTSC

Application Notes (Continued)

example again the smallest resistor for R_{SET} is 500 kΩ. The vertical default time delay is about 50 µs, much longer than the 30 µs serration pulse spacing.

A common question is how can one calculate the required R_{SET} with a video timing standard that has no serration pulses during the vertical blanking. If the default vertical sync is to be used this is a very easy task. Use the "Vertical Default Sync Delay Time vs R_{SET}" graph to select the necessary R_{SET} to give the desired delay time for the vertical sync output signal. If a second pulse is undesirable, then check the "Vertical Pulse Width vs R_{SET}" graph to make sure the vertical output pulse will extend beyond the end of the input vertical sync period. In most systems the end of the vertical sync period may be very accurate. In this case the preferred design may be to start the vertical sync pulse at the end of the vertical sync period, similar to starting the vertical sync pulse after the first serration pulse. A VGA standard is to be used as an example to show how this is done. In this standard a horizontal line is 32 µs long. The vertical sync period is two horizontal lines long, or 64 µs. The vertical default sync delay time **must be longer** than the vertical sync period of 64 µs. In this case R_{SET} must be larger than 680 kΩ. R_{SET} must still be small enough for the output of the integrator to reach V_1 before the end of the vertical period of the input pulse. The first graph can be used to confirm that R_{SET} is small enough for the integrator. Instead of using the vertical serration pulse separation, use the actual pulse width of the vertical sync period, or 64 µs in this example. This graph is linear, meaning that a value as large as 2.7 MΩ can be used for R_{SET} (twice the value as the maximum at 30 µs). Due to leakage currents it is advisable to keep the value of R_{SET} under 2.0 MΩ. In this example a value of 1.0 MΩ is selected, well above the minimum of 680 kΩ. With this value for R_{SET} the pulse width of the vertical sync output pulse of the LM1881 is about 340 µs.

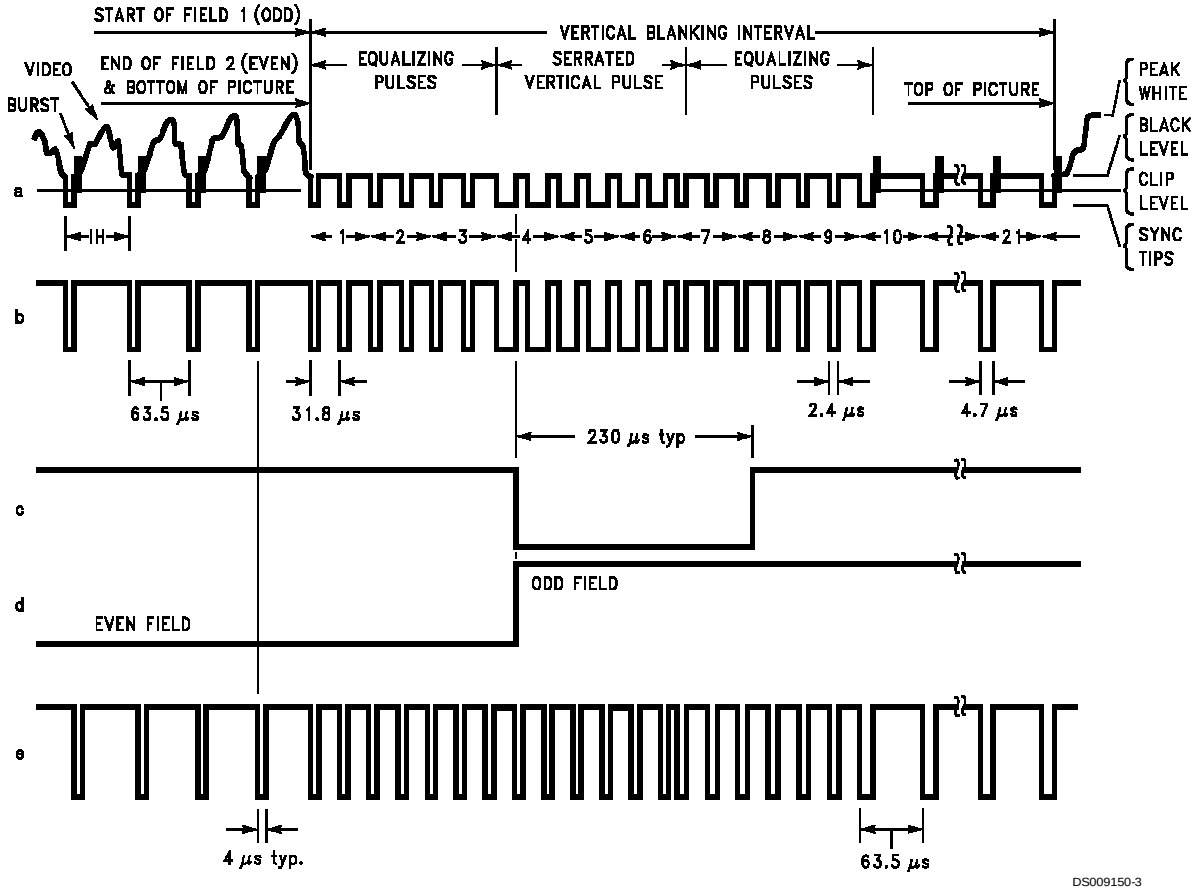
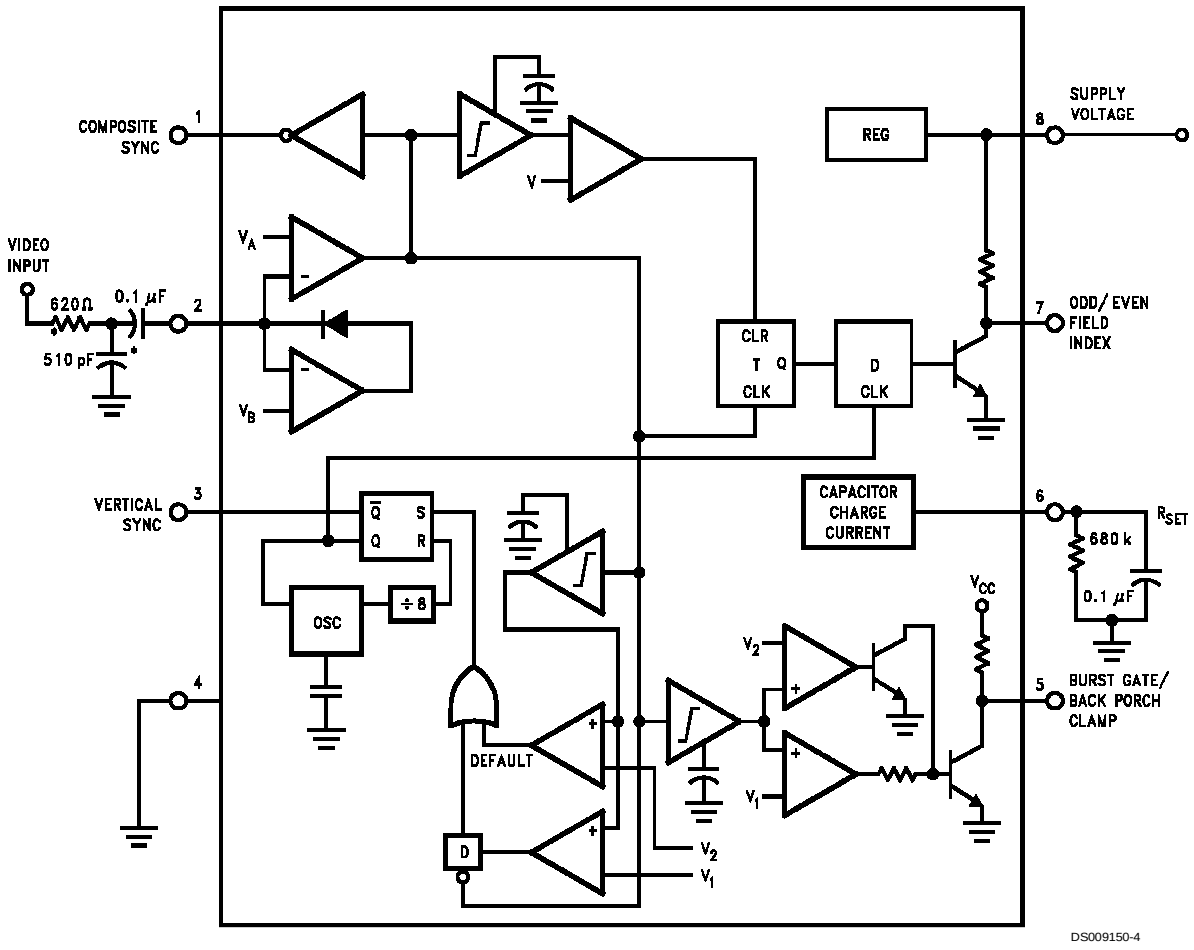


FIGURE 1. (a) Composite Video; (b) Composite Sync; (c) Vertical Output Pulse; (d) Odd/Even Field Index; (e) Burst Gate/Back Porch Clamp

Application Notes (Continued)



*Components Optional, See Text

FIGURE 2.

ODD/EVEN FIELD PULSE

An unusual feature of LM1881 is an output level from Pin 7 that identifies the video field present at the input to the LM1881. This can be useful in frame memory storage applications or in extracting test signals that occur in alternate fields. For a composite video signal that is interlaced, one of the two fields that make up each video frame or picture must have a half horizontal scan line period at the end of the vertical scan — i.e. at the bottom of the picture. This is called the “odd field” or “even field”. The “even field” or “field 2” has a complete horizontal scan line at the end of the field. An odd field starts on the leading edge of the first equalizing pulse, whereas the even field starts on the leading edge of the second equalizing pulse of the vertical retrace interval. Figure 1(a) shows the end of the even field and the start of the odd field.

To detect the odd/even fields the LM1881 again integrates the composite sync waveform (Figure 2). A capacitor is charged when the sync pulse is present. The period between charged during the period between sync pulses and discharged voltage to reach a threshold level of a comparator that normal horizontal sync pulses is enough to allow the capacitor waveform. When the vertical interval is reached, the shorter clears a flip-flop which is also being clocked by the sync integration time between equalizing pulses prevents this threshold from being reached and the Q output of the flip-flop is toggled with each equalizing pulse. Since the half line an equalizing pulse period, the Q output will have a different period at the end of the odd field will have the same effect as polarity with the vertical output pulse, an odd/even field index polarity on successive fields. Thus by comparing the Q high during the odd field is generated. Pin 7 remains low during the even field and

BURST/BACKPORCH OUTPUT PULSE

In a composite video signal, the chroma burst is located on the backporch of the horizontal blanking period. This period, approximately 4.8 µs long, is also the black level reference for the subsequent video scan line. The LM1881 generates a pulse at Pin 5 that can be used either to retrieve the chroma burst from the composite video signal (thus providing a subcarrier synchronizing signal) or as a clamp for the DC restoration of the video waveform. This output is obtained simply by charging an internal capacitor starting on the trailing edge of the horizontal sync pulses. Simultaneously the output of Pin 5 is pulled low and held until the capacitor charge circuit times out — 4 µs later. A shorter output burst gate pulse can be derived by differentiating the burst output using a series C-R network. This may be necessary in applications which require high horizontal scan rates in combination with normal (60 Hz–120 Hz) vertical scan rates.

Application Notes (Continued)

APPLICATIONS

Apart from extracting a composite sync signal free of video information, the LM1881 outputs allow a number of interesting applications to be developed. As mentioned above, the burst gate/backporch clamp pulse allows DC restoration of the original video waveform for display or remodulation on an R.F. carrier, and retrieval of the color burst for color synchronization and decoding into R.G.B. components. For frame memory storage applications, the odd/even field lever allows identification of the appropriate field ensuring the correct read or write sequence. The vertical pulse output is particularly useful since it begins at a precise time — the rising edge of the first vertical serration in the sync waveform. This means that individual lines within the vertical blanking period (or anywhere in the active scan line period) can easily be extracted by counting the required number of transitions in the composite sync waveform following the start of the vertical output pulse.

The vertical blanking interval is proving popular as a means to transmit data which will not appear on a normal T.V. receiver screen. Data can be inserted beginning with line 10 (the first horizontal scan line on which the color burst appears) through to line 21. Usually lines 10 through 13 are not used which leaves lines 14 through 21 for inserting signals, which may be different from field to field. In the U.S., line 19 is normally reserved for a vertical interval reference signal (VIRS) and line 21 is reserved for closed caption data for the hearing impaired. The remaining lines are used in a number of ways. Lines 17 and 18 are frequently used during studio processing to add and delete vertical interval test signals (VITS) while lines 14 through 18 and line 20 can be used for Videotex/Teletext data. Several institutions are proposing to transmit financial data on line 17 and cable systems use the available lines in the vertical interval to send decoding data for descrambler terminals.

Since the vertical output pulse from the LM1881 coincides with the leading edge of the first vertical serration, sixteen positive or negative transitions later will be the start of line 14 in either field. At this point simple counters can be used to select the desired line(s) for insertion or deletion of data.

VIDEO LINE SELECTOR

The circuit in Figure 3 puts out a singe video line according to the binary coded information applied to line select bits b0–b7. A line is selected by adding two to the desired line number, converting to a binary equivalent and applying the result to the line select inputs. The falling edge of the LM1881's vertical pulse is used to load the appropriate number into the counters (MM74C193N) and to set a start count latch using two NAND gates. Composite sync transitions are counted using the borrow out of the desired number of counters. The final borrow out pulse is used to turn on the analog switch (CD4066BC) during the desired line. The falling edge of this signal also resets the start count latch, thereby terminating the counting.

The circuit, as shown, will provide a single line output for each field in an interlaced video system (television) or a single line output in each frame for a non-interlaced video system (computer monitor). When a particular line in only one field of an interlaced video signal is desired, the odd/even field index output must be used instead of the vertical output pulse (invert the field index output to select the odd field). A single counter is needed for selecting lines 3 to 14; two counters are needed for selecting lines 15 to 253; and three counters will work for up to 2046 lines. An output buffer is required to drive low impedance loads.

MULTIPLE CONTIGUOUS VIDEO LINE SELECTOR WITH BLACK LEVEL RESTORATION

The circuit in Figure 4 will select a number of adjoining lines starting with the line selected as in the previous example. Additional counters can be added as described previously for either higher starting line numbers or an increased number of contiguous output lines. The back porch pulse output of the LM1881 is used to gate the video input's black level through a low pass filter (10 kΩ, 10 µF) providing black level restoration at the video output when the output selected line(s) is not being gated through.

Typical Applications

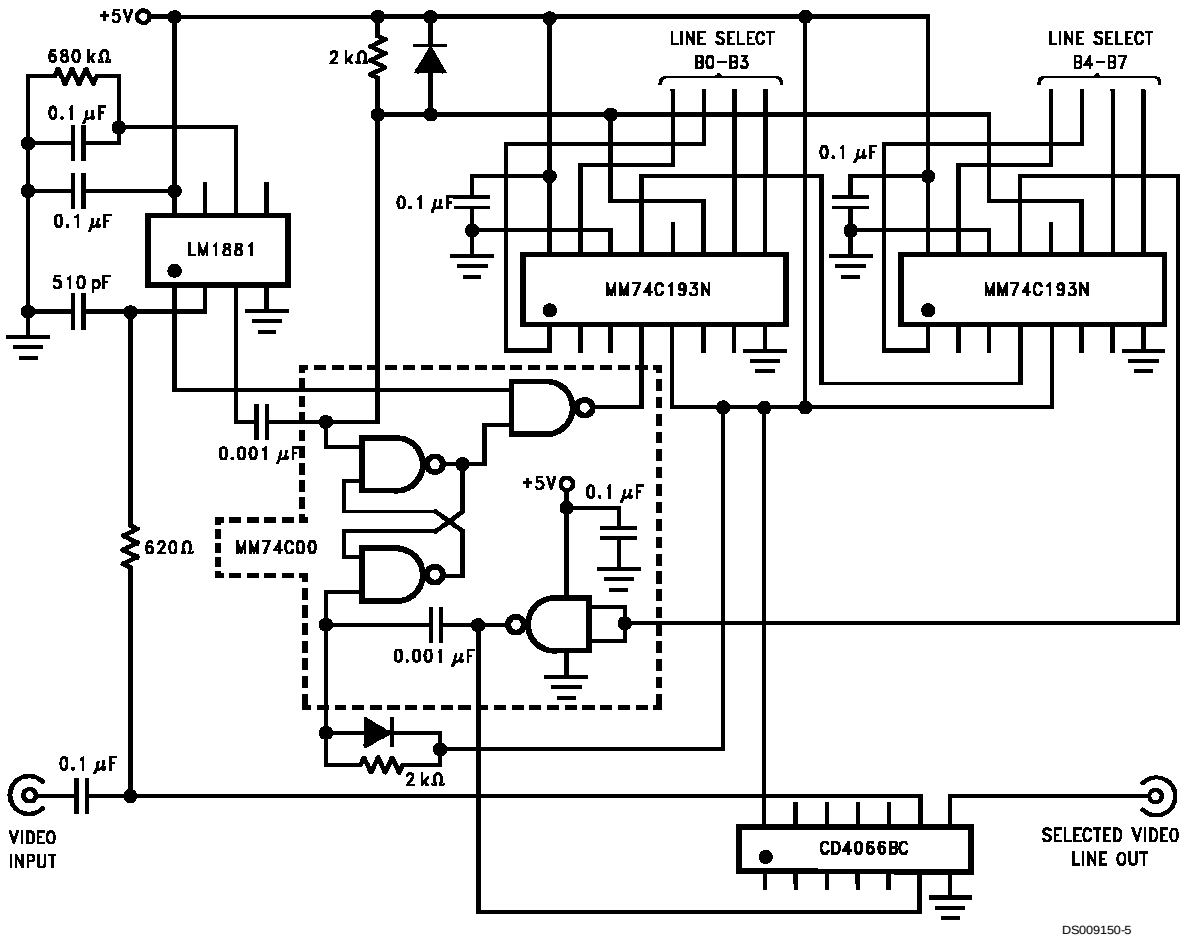


FIGURE 3. Video Line Selector

Typical Applications (Continued)

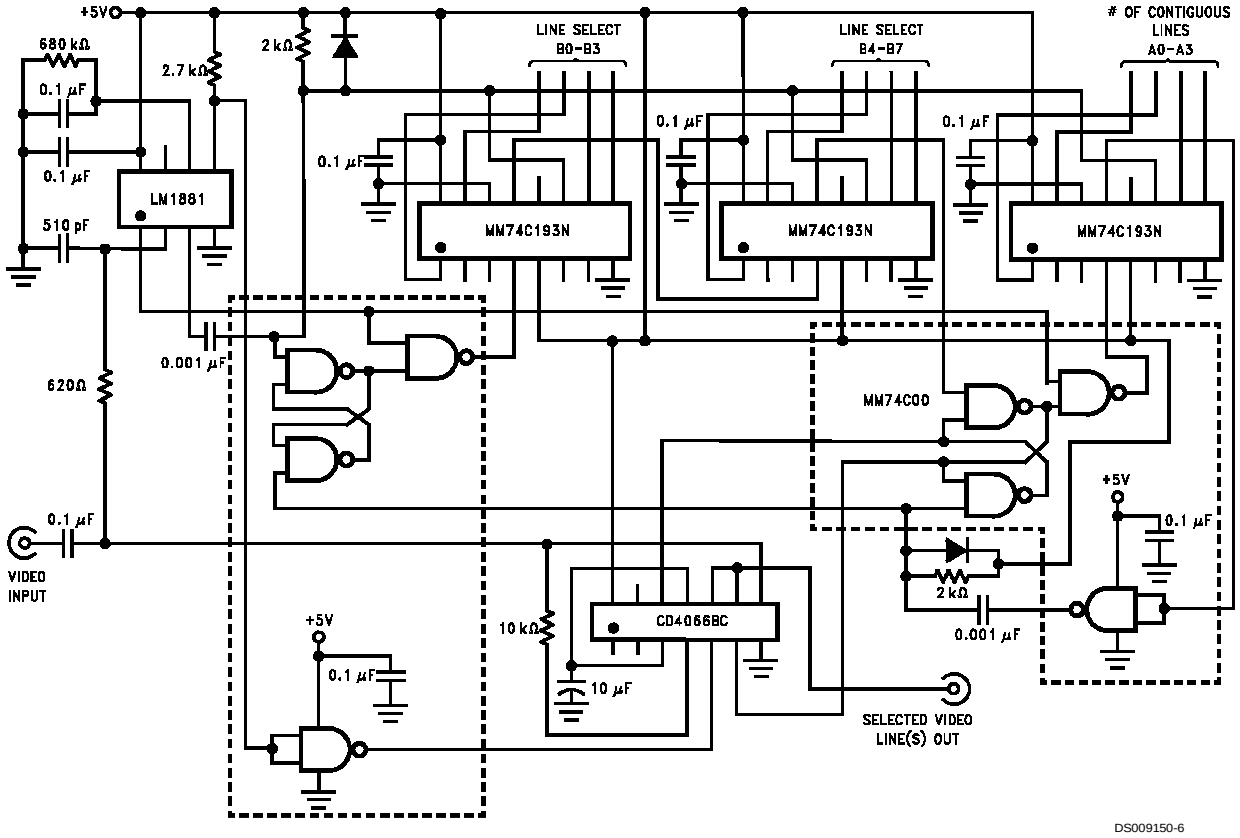
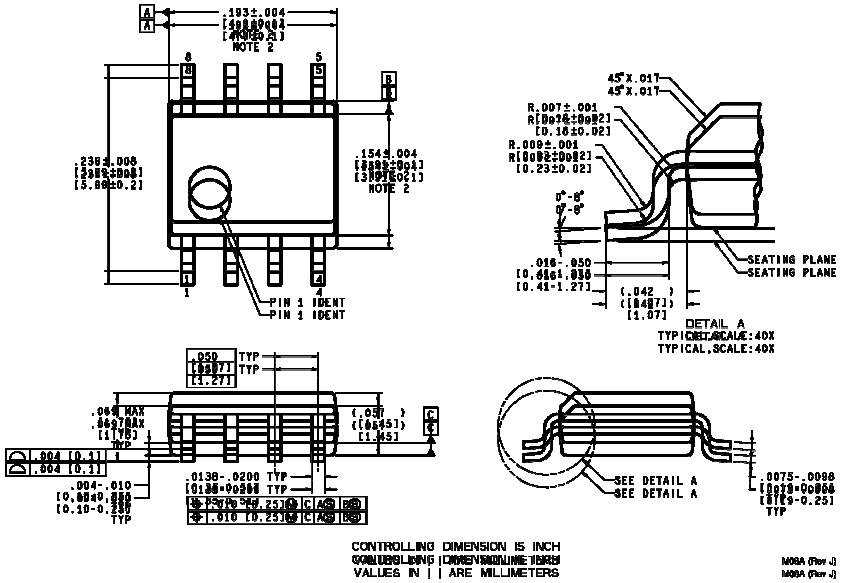
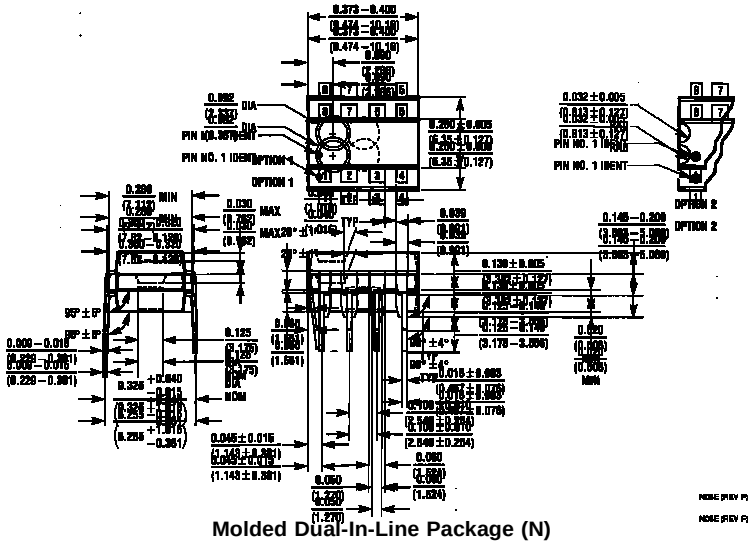


FIGURE 4. Multiple Contiguous Video Line Selector with Black Level Restoration

Physical Dimensions inches (millimeters) unless otherwise noted



Molded Small Outline Package (M)
Order Number LM1881M
NS Package Number M08A



Molded Dual-In-Line Package (N)
Order Number LM1881N
NS Package Number N08E

Notes

LIFE SUPPORT POLICY

NATIONAL'S PRODUCTS ARE NOT AUTHORIZED FOR USE AS CRITICAL COMPONENTS IN LIFE SUPPORT DEVICES OR SYSTEMS WITHOUT THE EXPRESS WRITTEN APPROVAL OF THE PRESIDENT AND GENERAL COUNSEL OF NATIONAL SEMICONDUCTOR CORPORATION. As used herein:

1. Life support devices or systems are devices or systems which, (a) are intended for surgical implant into the body, or (b) support or sustain life, and whose failure to perform when properly used in accordance with instructions for use provided in the labeling, can be reasonably expected to result in a significant injury to the user.
2. A critical component is any component of a life support device or system whose failure to perform can be reasonably expected to cause the failure of the life support device or system, or to affect its safety or effectiveness.

National Semiconductor Corporation
Americas
Tel: 1-800-272-9959
Fax: 1-800-737-7018
Email: support@nsc.com
www.national.com

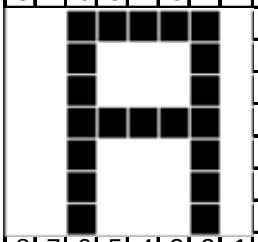
National Semiconductor Europe
Fax: +49 (0) 180-530 85 86
Email: europe.support@nsc.com
Deutsch Tel: +49 (0) 69 9508 6208
English Tel: +44 (0) 870 24 0 2171
Français Tel: +33 (0) 1 41 91 8790

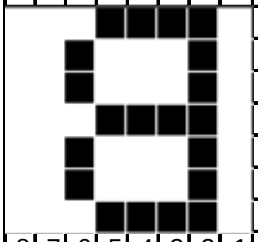
National Semiconductor Asia Pacific Customer Response Group
Tel: 65-2544466
Fax: 65-2504466
Email: ap.support@nsc.com

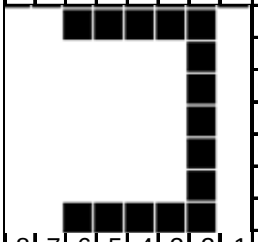
National Semiconductor Japan Ltd.
Tel: 81-3-5639-7560
Fax: 81-3-5639-7507

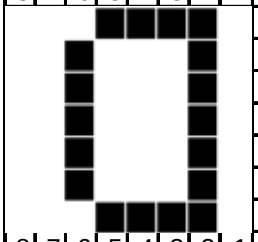
ANEXO 5
TABLA DE CARACTERES CREADOS

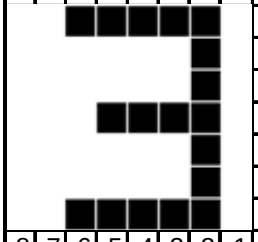
MAPA DE BITS DE LOS CARACTERES USADOS

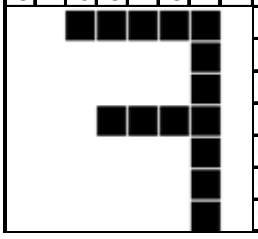
8 7 6 5 4 3 2 1	**A**
	3E
	22
	22
	3E
	22
	22
	22

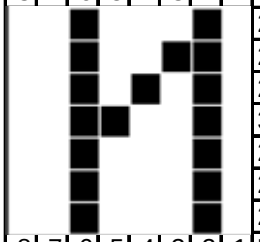
8 7 6 5 4 3 2 1	**B**
	1E
	22
	22
	1E
	22
	22
	1E

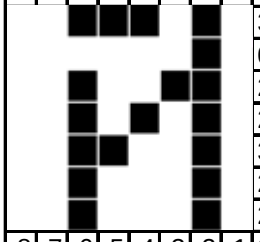
8 7 6 5 4 3 2 1	**C**
	3E
	02
	02
	02
	02
	02
	3E

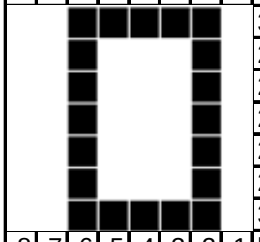
8 7 6 5 4 3 2 1	**D**
	1E
	22
	22
	22
	22
	22
	1E

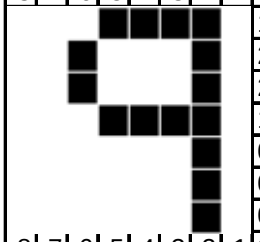
8 7 6 5 4 3 2 1	**E**
	1E
	02
	02
	E
	02
	02
	1E

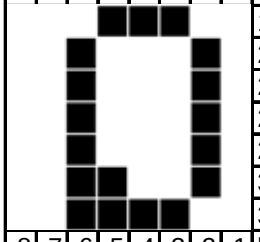
8 7 6 5 4 3 2 1	**F**
	2E
	02
	02
	1E
	02
	02
	02

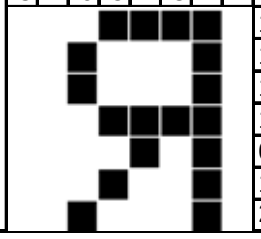
8 7 6 5 4 3 2 1	**N**
	22
	26
	2A
	32
	22
	22
	22

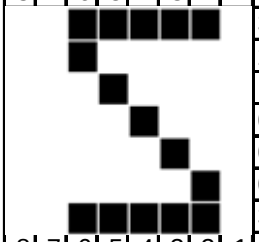
8 7 6 5 4 3 2 1	**Ñ**
	3A
	02
	26
	2A
	32
	22
	22

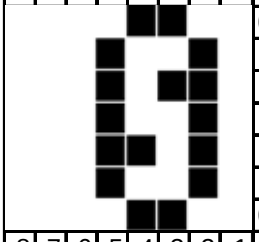
8 7 6 5 4 3 2 1	**O**
	3E
	22
	22
	22
	22
	22
	3E

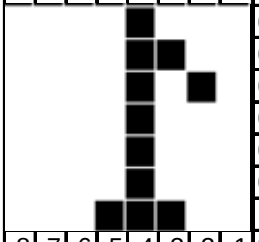
8 7 6 5 4 3 2 1	**P**
	1E
	22
	22
	1E
	02
	02
	02

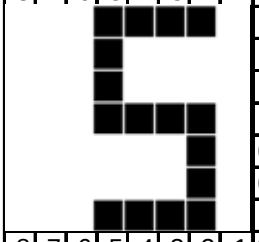
8 7 6 5 4 3 2 1	**Q**
	1C
	22
	22
	22
	22
	32
	3C

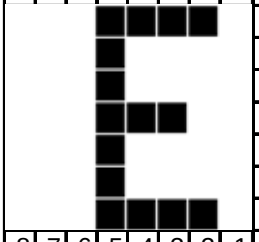
8 7 6 5 4 3 2 1	**R**
	1C
	12
	12
	1C
	0A
	12
	22

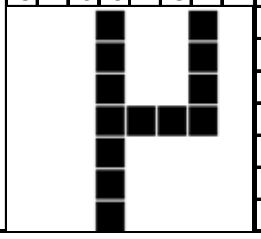
8 7 6 5 4 3 2 1	**Z**
	3E
	20
	10
	08
	04
	02
	3E

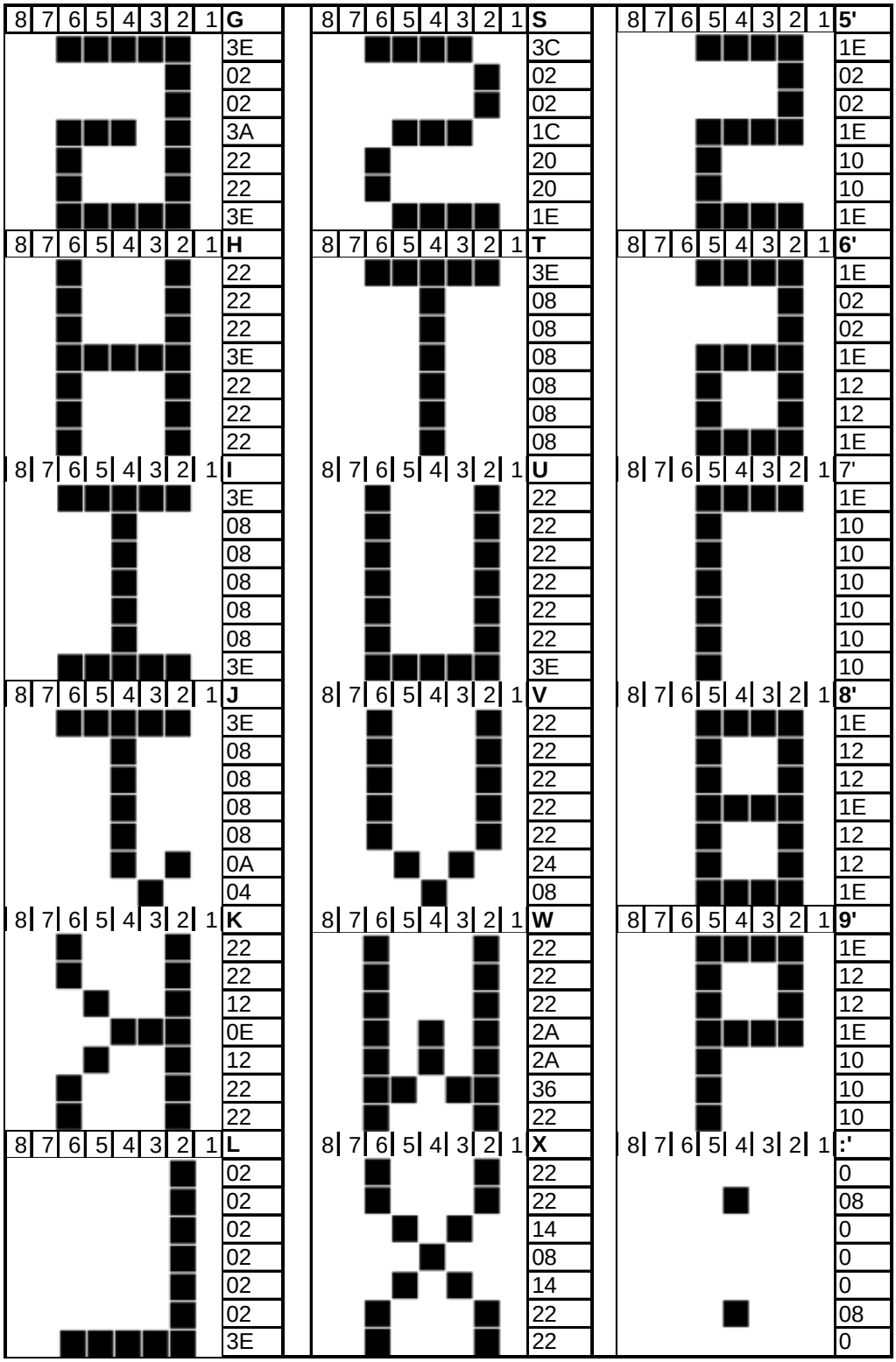
8 7 6 5 4 3 2 1	**0'**
	0C
	12
	16
	12
	1A
	12
	0C

8 7 6 5 4 3 2 1	**1'**
	08
	0C
	0A
	08
	08
	08
	1C

8 7 6 5 4 3 2 1	**2'**
	1E
	10
	10
	1E
	02
	02
	1E

8 7 6 5 4 3 2 1	**3'**
	1E
	10
	10
	1C
	10
	10
	1E

8 7 6 5 4 3 2 1	**4'**
	12
	12
	12
	1E
	10
	10
	10



8 7 6 5 4 3 2 1	G
	3E
	02
	02
	3A
	22
	22
	3E

8 7 6 5 4 3 2 1	S
	3C
	02
	02
	1C
	20
	20
	1E

8 7 6 5 4 3 2 1	5'
	1E
	02
	02
	1E
	10
	10
	1E

8 7 6 5 4 3 2 1	H
	22
	22
	22
	3E
	22
	22
	22

8 7 6 5 4 3 2 1	T
	3E
	08
	08
	08
	08
	08
	08

8 7 6 5 4 3 2 1	6'
	1E
	02
	02
	1E
	12
	12
	1E

8 7 6 5 4 3 2 1	I
	3E
	08
	08
	08
	08
	08
	3E

8 7 6 5 4 3 2 1	U
	22
	22
	22
	22
	22
	22
	3E

8 7 6 5 4 3 2 1	7'
	1E
	10
	10
	10
	10
	10
	10

8 7 6 5 4 3 2 1	J
	3E
	08
	08
	08
	08
	0A
	04

8 7 6 5 4 3 2 1	V
	22
	22
	22
	22
	22
	24
	08

8 7 6 5 4 3 2 1	8'
	1E
	12
	12
	1E
	12
	12
	1E

8 7 6 5 4 3 2 1	K
	22
	22
	12
	0E
	12
	22
	22

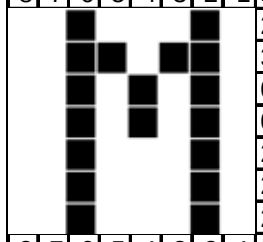
8 7 6 5 4 3 2 1	W
	22
	22
	22
	2A
	2A
	36
	22

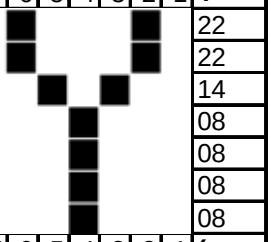
8 7 6 5 4 3 2 1	9'
	1E
	12
	12
	1E
	10
	10
	10

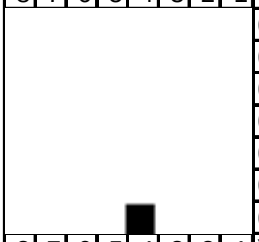
8 7 6 5 4 3 2 1	L
	02
	02
	02
	02
	02
	02
	3E

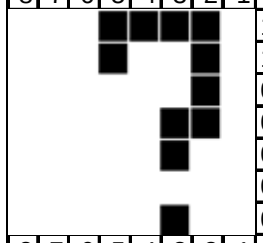
8 7 6 5 4 3 2 1	X
	22
	22
	14
	08
	14
	22
	22

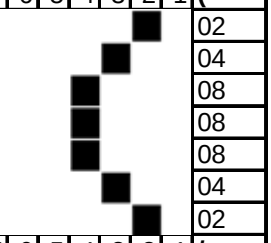
8 7 6 5 4 3 2 1	:'
	0
	08
	0
	0
	0
	08
	0

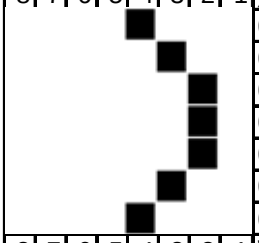
8 7 6 5 4 3 2 1	**M**
	22
	36
	08
	08
	22
	22
	22

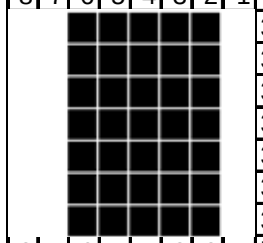
8 7 6 5 4 3 2 1	**Y**
	22
	22
	14
	08
	08
	08
	08

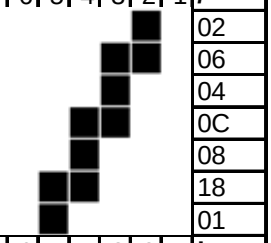
8 7 6 5 4 3 2 1	**.'**
	0
	0
	0
	0
	0
	0
	08

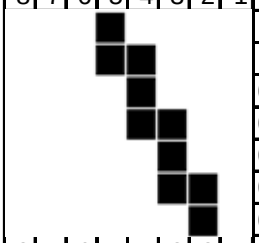
8 7 6 5 4 3 2 1	**?**
	1E
	12
	02
	06
	04
	0
	04

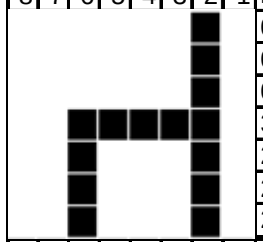
8 7 6 5 4 3 2 1	**(**
	02
	04
	08
	08
	08
	04
	02

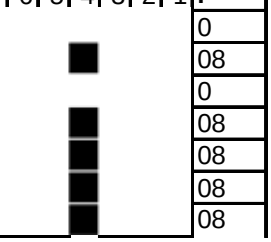
8 7 6 5 4 3 2 1	**)**
	08
	04
	02
	02
	02
	04
	08

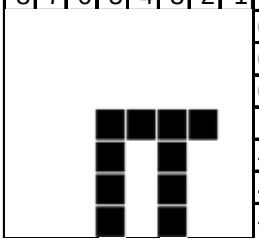
8 7 6 5 4 3 2 1	
	3E
	3E
	3E
	3E
	3E
	3E
	3E

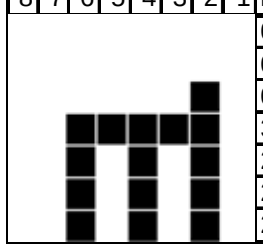
8 7 6 5 4 3 2 1	**/**
	02
	06
	04
	0C
	08
	18
	01

8 7 6 5 4 3 2 1	****
	10
	18
	08
	0C
	04
	06
	02

8 7 6 5 4 3 2 1	**h**
	02
	02
	02
	3E
	22
	22
	22

8 7 6 5 4 3 2 1	**i**
	0
	08
	0
	08
	08
	08
	08

8 7 6 5 4 3 2 1	**n**
	0
	0
	0
	1E
	44
	44
	44

8 7 6 5 4 3 2 1	**m**
	0
	0
	02
	3E
	2A
	2A
	2A