ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA ELECTRÓNICA EN TELECOMUNICACIONES Y REDES

"CONSTRUCCIÓN DE UNA TARJETA DE INTERFÁZ GRÁFICA PARA TELEVISIÓN UTILIZANDO MICROCONTROLADORES"

TESIS DE GRADO

Previa obtención del título de:

INGENIERO EN ELECTRÓNICA Y COMPUTACIÓN

Presentado por:

HENRY ERNESTO VALLEJO VIZHUETE

RIOBAMBA – ECUADOR

2011

NOMBRE	FIRMA	FECHA
Ing. Iván Menes **DECANO DE LAFACULTAD DE INFORMÁTICAY ELECTRÓNICA**	.....	.....
Ing. Pedro Infante **DIRECTOR DE ESCUELA EN TELECOMUNICACIONES Y REDES**	.....	.....
Ing. Paúl Romero **DIRECTOR DE TESIS**	.....	.....
Ing. Wilson Zúñiga **MIEMBRODEL TRIBUNAL**	.....	.....
Lcdo. Carlos Rodríguez **DIRECTOR DPTO. DOCUMENTACIÓN**	.....	.....
NOTA		.....

"Yo Henry Ernesto Vallejo Vizhuete, soy el responsable de las ideas, doctrinas y resultados expuestos en esta tesis; y, el patrimonio intelectual de la misma pertenecen a la **Escuela Superior Politécnica de Chimborazo**".

……
Henry Vallejo Vizhuete

ÍNDICE DE ABREVIATURAS

AM	AmplitudModulada
AVR	Advanced Virtual RISC
CDA	Convertidor Digital Analógico
DIP	Dual in-line package
D0	Dato
EEPROM	Electrically Erasable Program Random Only Memory
FM	Frecuencia Modulada
IG	Interfaz Gráfica
IGTV	Interfaz Gráfica Televisión
LR	Listo para Recibir
NTSC	National Television System Committee
PAL	Phase Alternating Line
PIC	Peripheral Interface Controller
RAM	Random Access Memory
RISC	Reduced Instruction Set Computer
RJ	Reloj
SECAM	SEquentialCouleur Aver Memoire
SYNC	Sincronismo
TRC	CatodeRay Tube
TV	Televisión

ÍNDICEGENERAL

ÍNDICE DE GRÁFICOS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El ojo humano es capaz de percibir imágenes mediante receptores ubicados en la retina. Hay dos tipos de receptores, según su función: los bastones, encargados de percibir imágenes en blanco y negro, y los conos, a cargo de la percepción del color.

Si nos concentramos en el estudio de los conos, veremos qué hay de tres tipos: los que reaccionan frente a la luz roja, los que lo hacen frente a luz verde y finalmente los que son sensibles a la luz azul. Solo se perciben tres colores, sin embargo nosotros “vemos” todos los colores que nos rodean. Aquí se hace evidente una regla básica del color: un objeto basta con tener la proporción de los tres colores básicos: Rojo, Verde y Azul. Por esta razón a dichos colores se los conoce como combinación de los mismos se pueden obtener todos los demás.

Entonces una imagen es en realidad, para el ojo humano, un sinnúmero de pequeñas porciones de luz; si cierta imagen está formada por elementos más pequeños que los del ojo, aquellos no son distinguidos por este y la imagen se percibe distorsionada. Ejemplo: un dibujo de dos líneas paralelas y cercanas entre sí; si un observador se va alejando del dibujo llega un momento en que no puede percibir la separación entre las líneas y son vistas como una sola.

Cuando una imagen es captada por el ojo, el efecto producido por la luz recibida permanece por un tiempo aproximado de 1/16 de segundo. Esto quiere decir que el cerebro recuerda la imagen presentada y la mantiene captada aunque haya sido retirada, de manera tal que si varias imágenes o pedazos de ellas son mostradas al ojo sucesivamente, pero dentro del 1/16 de segundo, el cerebro las asocia y las interpreta como una sola imagen.

Si se presentan al ojo varias imágenes fijas, en forma secuencial, ligeramente diferentes y en el tiempo correcto, el cerebro las interpreta como una imagen continua que va cambiando al paso del tiempo y que posee movimiento.

CAPITULO I
MARCO REFERENCIAL

1.1. JUSTIFICACIÓN

Se plantea la construcción de una tarjeta de interfaz gráfica para televisión que se pueda conectar a cualquier televisor o monitor con entrada de video para obtener así un display de alta resolución con varios tonos de gris. Además pueda procesar señales digitales para facilitar mostrar en la pantalla múltiples controles, imágenes y caracteres, para ello se utilizará invariablemente el uso de microcontroladores que envíe la señal de video a una televisión para su despliegue.

Al realizar este proyecto se puede cubrir la pregunta que algún día nos hemos hecho en la realización de nuestros proyectos cuando hay que mostrar alguna información de un circuito en particular y es, ¿Dónde mostrar dicha información? lo primero que pensaríamos es en un DISPLAY o LCD pero son dispositivos muy limitados, con poca resolución y algunos muy caros. Para lo cual una opción muy favorable es el mostrarlo en una televisión que cumpla con un solo requerimiento, que tenga una entrada de video.

Algunas posibles aplicaciones:

- Señale tica
- Juegos
- Comunicación interna en una institución
- Instrumentos de medición
- Recursos didácticos para enseñanza
- Sistemas de desarrollo completos para microcontroladores

1.2. ANTECEDENTES

El primer transporte a señal de video lo tiene un viejísimo video juego como lo es **Pong**, si ese que hemos visto en un sin fin de oportunidades en donde dos palitos y una pelotita que rebotan eran los protagonistas. Lo cierto es que ese simple primer video juego fue desarrollado en 1972 y hubo un problema con los derechos de autor, los cuales no se establecieron, permitiendo a muchas compañías adaptar el juego en un sin fin de aparatos electrónicos. El juego imitaba una partida de tenis de mesa en donde la pelota rebotaba siguiendo ciertos ángulos. Este juego se llegó a incluir dentro de la circuitería de nuevos televisores para la venta. Para 1974 ya se había consolidado a la compañía ATARI como la vendedora número uno de juegos electrónicos de mesa, en sus primeras versiones.El gran impulsor que tuvo ATARI fue que vendían un producto muy llamativo que conectándolo a la antena del televisor, transformaba aquella caja estática en un centro de diversión.

Hasta hoy en día hay gran variedad de dispositivo que realizan la labor de mostrar información en un televisor, sean estas imágenes o texto, desdeBetamax, (hoy descatalogado), introducido por Sony a principios de 1976, hasta los videojuegos como el nintendo Wii en auge hoy en día. Indispensablemente se necesita de microcontroladores para dicha labor ya que poseen las suficientes herramientas para mostrar dicha señal

1.3. OBJETIVOS

1.3.1. GENERAL

- Construir una tarjeta de interfaz gráfica para televisión utilizando microcontroladores

1.3.2. ESPECÍFICOS

- Construir una tarjeta de interfaz gráfica para televisión utilizando microcontroladores
- Investigar sobre generación de patrones de televisión utilizando microcontroladores
- Implementar un sistema electrónico que permita visualizar información en la televisión
- Diseñar y generar los Algoritmos y programas para control de la Interfaz gráfica
- Generar documentación Técnica referente al Hardware y Software del proyecto

1.4. HIPOTESIS

Mediante el proyecto planteado se pretende demostrar la efectividad del uso de microcontroladores aplicados a la generación de señales de video para televisión.

CAPITULO II
FUNDAMENTO TEÓRICO

2.1. HISTORIA DE LA TELEVISIÓN

Figura II. 2. Televisor

El concepto de televisión (visión a distancia) se puede rastrear hasta Galileo Galilei y su telescopio. Sin embargo, no es hasta 1884, con la invención del Disco de Nipkow de Paul Nipkow cuando se hiciera un avance relevante para crear un medio. El cambio que traería la televisión tal y como hoy la conocemos fue la invención del iconoscopio de Vladimir Zworkyn y Philo Taylor Farnsworth. Esto daría paso a la televisión completamente electrónica, que disponía de una tasa de refresco mucho mejor, mayor definición de imagen e iluminación propia.

2.1.1. Primeros desarrollos

En 1910, el disco de Nipkow fue utilizado en el desarrollo de los sistemas de televisión de los inicios del siglo XX y en 1925 el inventor escocés John Logie Baird efectúa la primera experiencia real utilizando dos discos, uno en el emisor y otro en el receptor, que estaban unidos al mismo eje para que su giro fuera síncrono y separados 2 m.

Las primeras emisiones públicas de televisión las efectuó la BBC en Inglaterra en 1927 y la CBS y NBC en Estados Unidos en 1930. En ambos casos se utilizaron sistemas mecánicos y los programas no se emitían con un horario regular.

Las emisiones con programación se iniciaron en Inglaterra en 1936, y en Estados Unidos el día 30 de abril de 1939, coincidiendo con la inauguración de la Exposición Universal de Nueva York. Las emisiones programadas se interrumpieron durante la Segunda Guerra Mundial, reanudándose cuando terminó.

2.1.2. Televisión en Ecuador

Hablar de la Televisión en Ecuador nos tenemos que remontar 40 años, y preguntarse, quien tuvo el idea de tener la empresa televisora y hasta del primer televisor, esta industria a lo largo de la historia se convierte en la más grande fuente de ingresos, rentable y seguras, empresas del país, ya que las grandes cantidades de dinero que en ella mueven e invierten son inimaginables en cuanto a lo económico se refiere, tienen constante movimiento al marketing mundial, constituyéndose en poder, influyendo en el destino de la Nación.

En el año de 1954 un norteamericano de apellido Hartwell encontró un equipo abandonado en bodegas de General Electric en Syracuse, New York, y fue hasta 1959 que dichos equipos llegaron hasta Quito, asombrando con la nueva tecnología, en ese mismo año fue que la televisión pasa a manos de los protestantes, es ahí cuando la Unión Nacional de Periodistas lleva esos equipos a la HCJB, para realizar una feria celebrada en el Colegio Americano y ver la televisión en blanco y negro.

Fue en 1960 que gracias a la feria de Octubre que la televisión llega al puerto de Guayaquil tras convenio con la Casa de la Cultura, es así que Canal 4, que ahora denominada como Red Telesistema (RTS), obtiene el permiso de laborar y operar así es como nace la televisión en el Ecuador

2.2. TRANSMISIÓN Y RECEPCION

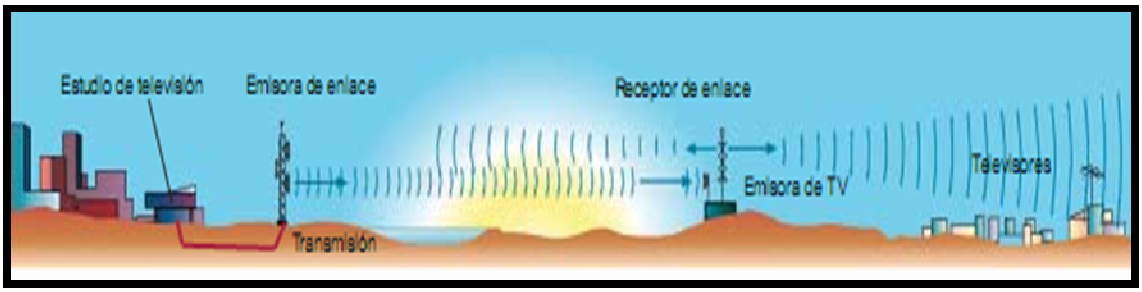


Figura II. 2. Transmisión y recepción desde estación televisoras hasta los televisores

Las antenas transmisoras radian ondas electromagnéticas que son captadas por las antenas receptoras.

El transmisor de televisión tiene dos funciones: transmisión de imagen y de sonido. Ambas señales AM de imagen y FM de sonido son emitidas desde la misma antena transmisora... El radio del área de servicio es de aproximadamente 75 millas (121 Kilómetros) en todas las direcciones desde el transmisor.

En la trasmisión de la imagen, el tubo de cámara convierte la imagen óptica en una señal de video. El tubo de cámara es un tubo de rayos catódicos (TRC) con una placa de imagen fotoeléctrica y un cañón electrónico confinados por una envolvente de vidrio en la cual se ha practicado el vacío. Un tipo comúnmente empleado es el vidicón.

Fundamentalmente el tubo de cámara capta una imagen óptica de la escena en su placa fotoeléctrica, la cual es explorada en líneas horizontales por el haz electrónico. La exploración se hace de izquierda a derecha y de arriba abajo, o sea, desde la parte superior hasta la inferior, tal como es captada por la cámara. La exploración del cuadro completo de imagen dura $^{1}/_{3}$s, comprendiendo un total de525 líneas de exploración.

Por lo tanto la salida del tubo de cámara es una secuencia de variaciones eléctricas – la señal de video – que corresponde a la información de imagen. La señal de video es amplificada y son añadidos los impulsos de sincronización o sincronismo.

La modulación de amplitud de la portadora de imagen da por resultado la señal AM de imagen.

La antena receptora intercepta ambas señales portadoras de imagen y de sonido. Las señales son amplificadas y luego detectadas para recuperar la modulación original. La salida del detector video incluye la señal de video necesaria para reproducir la imagen.

Luego es amplificada la señal video detectada lo suficiente para excitar el circuito rejilla – cátodo del tubo de imagen, el tubo de imagen es muy parecido el TRC utilizado en un osciloscopio. La placa frontal de vidrio tiene un revestimiento fluorescente en su superficie interior. El cuello estrecho contiene el cañón electrónico. Cuando el haz de electrones incide en la pantalla de fósforo, esta emite luz.

Cuando la tensión de la señal video hace que la rejilla de control sea menos negativa, la corriente del haz aumenta, haciendo que el punto luminoso de la pantalla sea más brillante. La salida de luz máxima es un punto blanco en la imagen.

Por el contrario, una tensión más negativa de rejilla reduce el brillo, y si la tensión de la rejilla suficientemente negativa para cortar o suprimir la corriente del haz electrónico en el tubo de imagen, la luz desaparece. Este valor corresponde al negro en la pantalla.

El diagrama de bloques de la (figura*3)* ilustra el sistema monocromático. En la televisión a color se utiliza una cámara y un tubo de imagen en color. La cámara en color provee las señales video para la información de imagen en:rojo, verde y azul. Análogamente, el tubo de imagen reproduce la imagen en rojo verde y azul con todas sus mezclas de color incluyendo el blanco.

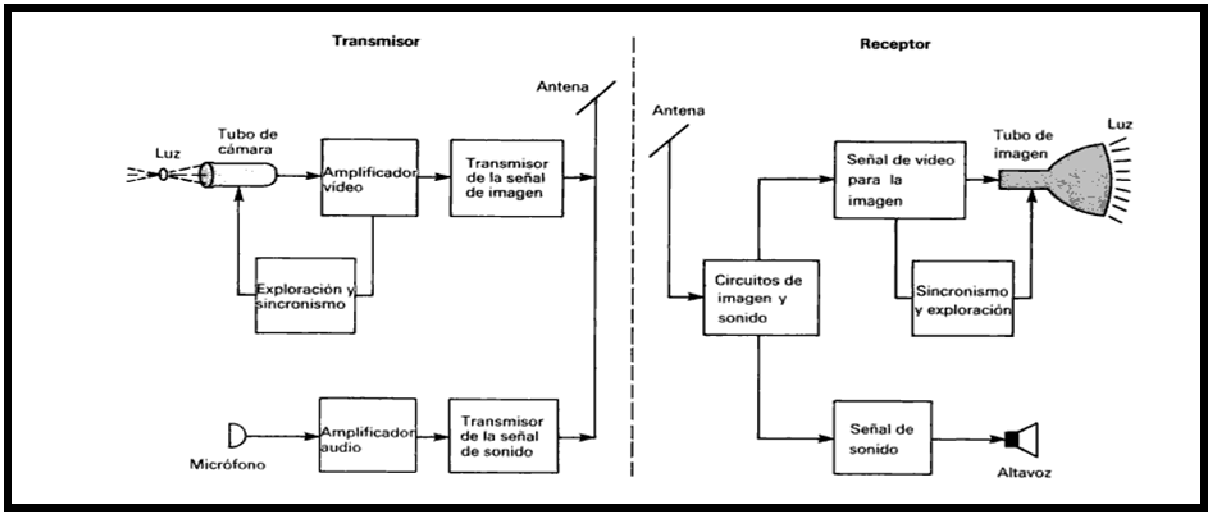


Figura II. 3. Diagrama de bloques del sistema de difusión de televisión

2.3. ELEMENTOS DE LA TELEVISIÓN

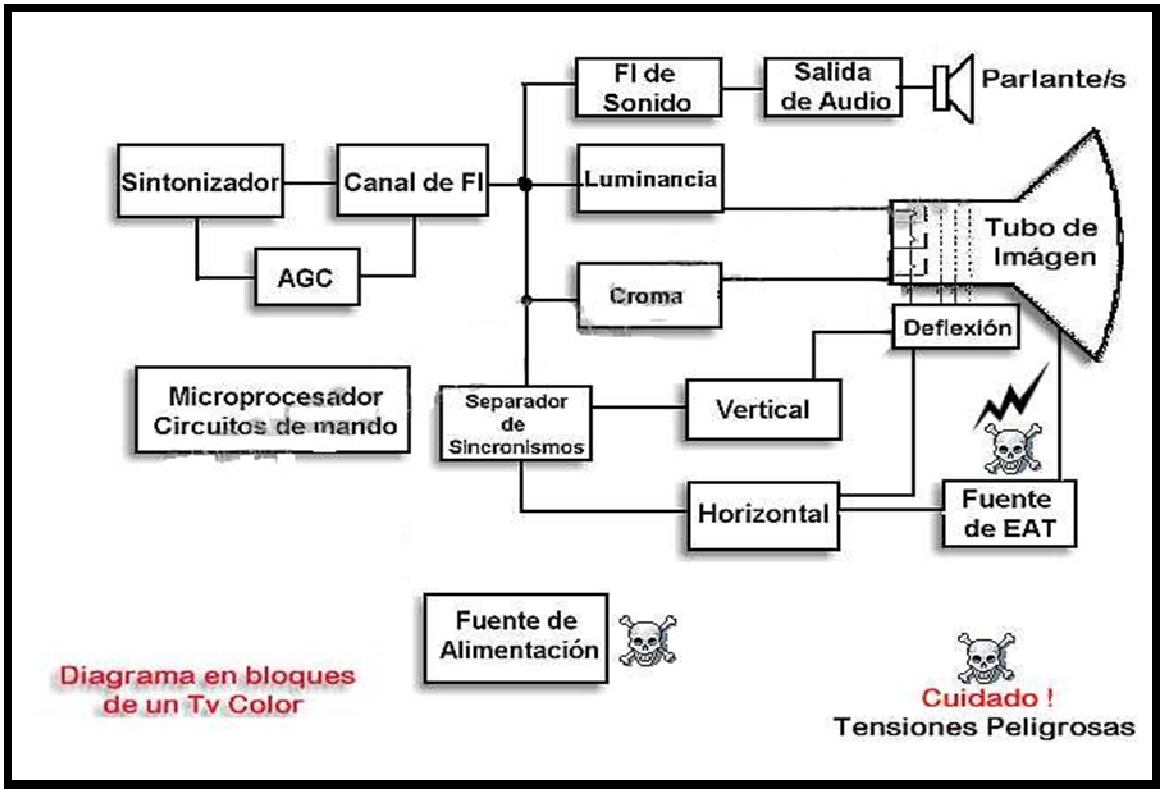


Figura II. 4. Partes fundamentales que componen un televisor

2.3.1. Sintonizador: Recibe y amplifica la señal de TV de cable o antena, y produce una señal de FI conteniendo el canal seleccionado mediante Vc y los terminales de control de los conmutadores de banda.

2.3.2. AGC: Las iniciales AGC significan **A**utomatic **G**ain **C**ontrol lo que en castellano sería Control Automático de Ganancia .La función de esta epapa dentro de un TV , es equilibrar las amplitudes a la salida del amplificador de video del canal de FI , para su posterior tratamiento en los circuitos de Audio , Luminancia y Croma .Es decir , este circuito "mide" constantemente la amplitud de la señal de video compuesto recuperada, y " le informa " de dichas mediciones al sintonizador y al primer amplificador de FI , para , llegado el caso , estos deban aumentar su rendimiento ante señales débiles o deban disminuirlo debido a que la componente de video recuperada está sobrepasando los límites de funcionamiento normal .
O sea que, si este circuito no existiera, tendríamos que en un TV que recibe transmisiones de varios canales, sean por aire o por cable, todos se verían distinto, algunos con mucha lluvia, otros normalmente y cuando las transmisiones son locales, la fuerza de la señal, saturaría de tal manera que sería imposible ver.

2.3.3. Canal de FI: Debido al constante avance de la miniaturización y la integración de múltiples etapas del TV dentro de un sólo integrado, podemos decir que el canal de FI, no debería traernos mayores dolores de cabeza.
Dentro de esta etapa podemos encolumnar los siguientes sub-bloques: Amplificadores de Frecuencia Intermedia, Circuitos detectores de sobrecarga, Demodulador sincrónico, AFT o AFC, Inversor de ruido y Amplificador de Video.

2.3.4. Microprocesadores y circuitos de mando: El microprocesador por un lado recibe órdenes, las procesa, decide en base a una serie de instrucciones llamadas programa y ejecuta en consecuencia. En nuestro caso , el de un TV , podemos decir que recibe una orden desde el receptor del Control Remoto o desde el teclado del panel frontal , procesa ese requerimiento , decide a través del programa cargado por el fabricante , y luego ejecuta en consecuencia : sube o baja el volumen , cambia de canales , etc.

En la gran mayoría de las aplicaciones vienen acompañados de pequeños IC que son Memorias EEPROM (Electrically Erasable Program Random Only Memory). Estas sirven para almacenar todos los datos de preferencia del usuario. Ultimo canal mirado, nivel de volumen, intensidad de brillo, contraste, color, sintonía de canales, etc. El micro graba en ellas toda la información necesaria durante el funcionamiento del TV para que al apagarlo y encenderlo nuevamente, no se inicialice todo, sino que mantenga los registros tal como cuando se apagó. A todo el conjunto formado por el Micro, la Memoria, el Receptor del Remoto, el Teclado y los circuitos que adaptan estos últimos al TV, lo denominaremos **Circuito de Mando**.

2.3.5. Fuente de Alimentación de un TV: La fuente de alimentación en un TV, como en una Video, o cualquier otro equipo electrónico, es una sección muy bien definida que no será muy difícil de identificar físicamente.

2.3.6. Fl de sonido y salida de Audio: Estas etapas serán tratadas en un mismo apartado debido a la simpleza de la última y la conectividad que poseen entre sí. Una vez obtenida la señal de video compuesta del Canal de FI, el primer paso es separar, la imagen del sonido. En el ancho de banda que ocupa un canal, en América 6 Mhz, se reparte para la imagen, desde 0 a 4,2 Mhz y el resto es dedicado al sonido, con una frecuencia de subportadora de audio ubicada en los 4,5 Mhz.
Entonces, nos encontraremos que a la entrada de las etapas de FI de sonido, tenemos un filtro, generalmente cerámico, que dejará pasar sólo la parte superior del espectro de un canal, es decir, donde viene la información de audio. El audio se encuentra dentro de la señal, modulado en frecuencia, por lo que esta componente deberá ser limitada, detectada, controlada en su amplitud y luego será enviada al amplificador final de audio para su reproducción.

2.3.7. Separador de Sincronismos y Oscilador Horizontal: Se conoce al Separador de Sincronismos como la etapa del TV que se encarga de extraer, desde la señal compuesta de video, los impulsos necesarios para enclavar la imagen en la pantalla. Tanto el Oscilador de Vertical, como el de Horizontal, son libres, o sea que, funcionan a una frecuencia muy cercana a la del transmisor, y necesitan de una información enviada por éste último para que la imagen no " flote " en la pantalla de un lado a otro.

En la mayoría de los casos en que tenemos pérdida de sincronización en la imagen, pensamos en este sector, pero la práctica nos demuestra que la falta de sincronización se debe a cualquier otra cosa, menos a una falla en esta sección. Es muy raro que falle esta etapa. Desde aquí se toma una muestra del sincronismo de la señal que estamos recibiendo y se envía una información de ella a: a) el Detector de Coincidencia. Este es un circuito que le informa al Microprocesador de que el canal se ha encontrado. Cuando esto falla, se presenta que la sintonía varía de un lado a otro del canal sin encontrarlo. Nosotros lo vemos que pasa, pero el Micro no. b) luego pasa al circuito del AFC o AFT (Automatic Fine Tuning) el que se encuentra interconectado con el. Aquí se detecta el mejor punto de la sintonía, que no quede desplazada, sino en el punto de máxima amplitud de los sincronismos, que, por lógica será, el de máxima amplitud de video compuesto recuperado. Ambos circuitos informarán al Microprocesador que la amplitud es la máxima, que ahí está bien

2.3.8. Etapa vertical: Para que el haz electrónico emitido por los cátodos del tubo llenen la pantalla con imagen, necesitamos moverlo y hacerlo recorrer apropiadamente, todo el largo y ancho de la misma .El encargado de efectivizar este movimiento será el Yugo, pero a éste debemos indicarle cómo hacerlo. Los osciladores locales de vertical, se encuentran en la mayoría de los TV modernos integrados en el Jungle, y pueden ser libres, controlados por potenciómetros de acceso al usuario en el frente del TV, o bien del tipo "Cont-down", los que se rigen por un generador de reloj a Resonador Cerámico, en frecuencias que varían entre 455 Khz y 503 Khz. Estas frecuencias son sometidas a divisiones fijas y constantes, para obtener las frecuencias de oscilación para el vertical y el horizontal. Una vez recibido el impulso de sincronización vertical desde los separadores de sincronismos, éste se aplica apropiadamente al oscilador que determinará la frecuencia del barrido vertical, para sincronizarlo en fase con el del transmisor que genera la señal que deseamos ver. Luego encontramos un IC dedicado, al que le llega la información del Trigger que proviene del oscilador ya sincronizado, mediante la cual, se controla un "Generador de Rampa", que luego se amplifica para energizar apropiadamente el Yugo

2.3.9. Etapa horizontal: La etapa de Horizontal, podemos decir, se encuentra formada por, Oscilador Horizontal, Transistor Driver, y Transistor de Salida Horizontal. El Oscilador

Horizontal se encuentra habitualmente dentro de lo que se conoce como Jungle. En la mayoría de los diseños, este oscilador recibe desde la Fuente de Alimentación una tensión que está comprendida entre 8 y 12 Volts para inicializar su funcionamiento en el momento de arranque. Cuando esto ocurre, comenzará a oscilar libremente en una frecuencia muy aproximada a la de funcionamiento final. Excitará los circuitos del Driver, estos a su vez harán lo propio con el Transistor de Salida Horizontal y comenzarán a generarse dos situaciones distintas en este momento.

2.3.10. Deflexión: El TRC bombardea desde su cátodo , electrones que llegan hasta la pantalla provocando la luminiscencia .Para que dicha emisión no sea un punto en el centro de la pantalla , se utiliza una unidad en la parte final del cuello del TRC que se la conoce como "Yugo" , o bobinas de deflexión , las que , alimentadas por tensiones específicas , crean campos electromagnéticos en la trayectoria del haz electrónico , provocando su desvío y recorrido , a lo largo y a lo ancho de toda la pantalla .Este movimiento es tan veloz que el ojo humano y la persistencia de luminosidad del fósforo en la pantalla , hacen que parezca que estamos observando una imagen siempre entera y constante , aunque en realidad sea un único punto luminoso que se encarga de recorrer , como dijimos , bajo un cierto orden , toda la pantalla .Ese orden viene dado según la frecuencia del movimiento en forma vertical y en forma horizontal . En Argentina dichas frecuencias son: Vertical 50 Hz. y Horizontal 15625 Hz.

2.3.11. Fuente de Extra Alta Tensión (Fuente de EAT): Más conocida como " La Zona del Fly-Back " posee algunas cuestiones a tener en cuenta, como La pintura negro mate que recubre el TRC en su exterior , es lo que se llama "AQUADAG" y es de características conductivas .Verán que está conectada a potencial cero es decir a GND.Por otro lado, el ánodo del tubo trabaja con una tensión que se encuentra en el orden de los 25000 Volts aproximadamente .Si consideramos que a estas dos tensiones (25000 Volts y 0 Volts) las separa el vidrio de la ampolla, notaremos que estamos en presencia de un capacitor de dimensiones considerables. A pesar de que pueda pasar un tiempo considerable sin que el TV se utilice, este capacitor puede almacenar energía suficiente como para para darnos una fuerte descarga.

2.3.12. Tubo de Imagen y Amplificadores RGB: Lo que denominamos tubo de imagen del TV, se lo conoce como CRT (**C**atode **R**ay **T**ube), cinescopio, pantalla, TRC, etc.El TRC es una válvula como cualquier otra, que posee un Anodo o Placa Gigantesco (comparado a las Válvulas convencionales) (o sea, es una "superválvula") a donde van a dar los electrones expulsados del Cátodo.

Ese Anodo se diferencia de sus congéneres por estar adherido al vidrio y formado por diminutas celdillas de Fósforo que todos conocen como "Píxel". Cuando los electrones chocan contra el Fósforo se produce una luminiscencia, que, ordenada de una forma particular y a una velocidad determinada obtenemos la imagen. Entonces, esto que estás leyendo, lo haces sobre el Anodo de una Válvula. Existen fabricantes que están incorporando tecnología de " Plasma ", logrando dimensiones finales en las pantallas de algunos pocos centímetros. Si bien esta es una historia que cambiará tarde o temprano, hoy, la realidad es vidrio que es el TRC y es uno de los terminales de salida de la información que le llega a la antena.

2.4. FORMACIÓN DE LA IMAGEN EN EL TELEVISOR

Para entender un poco sobre la generación de las señales video en tiempo real, se debe saber cómo trabajan el vídeo-señal detalladamente.

2.4.1. Cómo funciona una TV estándar.

Un TV estándar se construye con un tubo de vacío, que tiene una pantalla de fósforo en la cual un cañón bombardea con electrones. Cuando los electrones del cañón golpean la pantalla, el fósforo emite luz mientras el cañón tire a electrones en él, y también tiene una pos luminiscencia corta. El haz electrónico del electrón se puede doblar usando los imanes así que tira en diversas partes de la pantalla. Si se controla esto así, dibuja líneas horizontales por todas partes en la pantalla en varias pasadas, mientras que la intensidad de la línea es controlada, una imagen se puede dibujar en la pantalla. La pantalla es redibujada 25 veces por segundo en un sistema PAL, pero al reducir oscilación la imagen se entrelaza, mostrando primero todas las líneas impares uniformes, así que la imagen es parcialmente actualizada 50 veces por segundo. Hay que agradecer a la “persistencia

del efecto de la visión" del cerebro humano que la imagen parece ser constante en vez de oscilar a 50Hz. Para conseguir a color cada punto de la pantalla se divide en tres colores: rojo, verde y azul, no obstante aquí discutiremos solamente la televisión en blanco y negro, porque eso es solamente lo que es posible generar tiempo real en software usando un PIC.

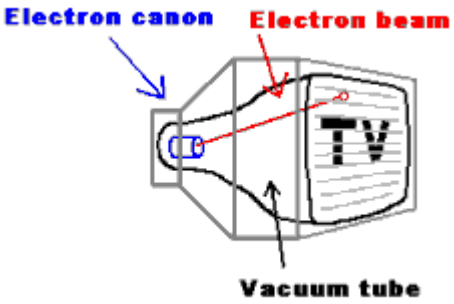


Figura II. 5. El haz electrónico que dibuja la pantalla

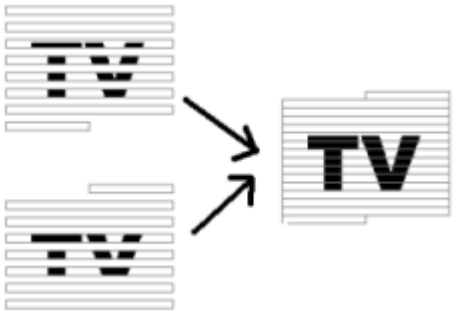


Figura II. 6. Las imágenes bipartitas se convierten en una imagen

2.4.2. La información en la señal video

La imagen considerada en la pantalla tiene diversas intensidades. Como los barridos del haz electrónico sobre la pantalla, la intensidad que debe estar en la posición de la línea, se envía como nivel voltaico en la señal video. No hay información en esta señal de intensidad sobre la que está el haz electrónico en la pantalla. Para solucionar esto, un pulso de sincronización se envía al principio de cada línea para decir que la línea actual de TV está acabada y para bajar el haz electrónico a la línea siguiente. (Como la llave del <Enter> en el teclado, al escribir un texto con una computadora) La TV debe también saber cuándo está viniendo una nueva imagen, esto se realiza haciendo un patrón especial de la sincronización. (Como la función del "nuevo documento" al escribir un texto con una computadora) una imagen que es 25 veces actualizadas por segundo estaría oscilando absolutamente, por consiguiente todas las líneas uniformes se dibuja primero y entonces todos los impares, muestreando de este modo 50 imágenes por segundo, haciendo que el cuadro oscile menos. Las líneas o impares de la información se envían en el patrón de la sincronización vertical, como diversos patrones para las imágenes

impares y uniformes. La señal de video tiene 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 sincronización.

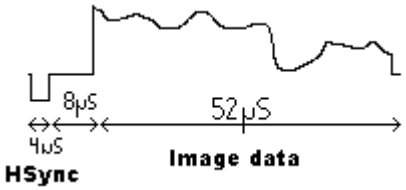


Figura II. 7. Cuadro del "osciloscopio" - de una explorar-línea

2.4.3. Las líneas

La imagen se divide en líneas, él es la parte más importante de la imagen puesto que contiene los datos de la imagen. Las líneas son de 64us de largo. Primero un pulso largo de sincronización de 4us es enviado, fijando la señal llana a 0V, para decir la TV que está viniendo un giro nuevo. Las viejas TV eran más lentas, así que necesitaban 8us después del pulso para conseguir el haz electrónico en la posición. Durante este tiempo la señal se guarda en el nivel negro. El retraso de 8us es seguido por los datos de la imagen para 52us, dibujados en la pantalla de izquierda a derecha con las intensidades obtenidas de la señal video. El negro es representado por 0.3V y como el voltaje aumenta la intensidad, con la intensidad máxima se tiene 1.0v (blanco).

2.4.4. Juntando las líneas como una imagen

Una imagen se construye de 625 líneas, pero una TV no muestra 625 líneas. Algunas de las líneas se utilizan para los pulsos de sincronización, y algunas líneas son invisibles porque las viejas TV necesitaron un cierto tiempo para mover el haz electrónico desde el fondo de la pantalla. (Esas líneas invisibles se utilizan hoy en día para otros propósitos, Texto-TV por ejemplo).

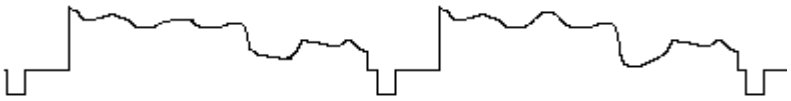


Figura II. 8. Cuadro del "osciloscopio" - de varias líneas en una señal video.

2.4.5. Los pulsos de sincronización vertical.

Para decirle a la TV que esté viniendo una nueva imagen, un patrón especial de pulsos de sincronización es enviado. Puesto que el cuadro se construye a partir de dos medios cuadros, el patrón es diferente para las imágenes impares y uniformes. La sincronización vertical es la siguiente:

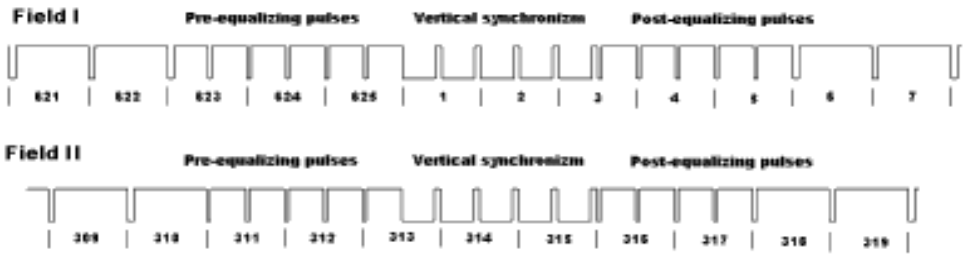


Figura II. 9. Este cuadro muestra los diversos pulsos de la sincronización vertical para las dos medias imágenes. Los niveles son 0v y 0.3v.

2.5. NORMAS Y SISTEMAS DE TELEVISIÓN

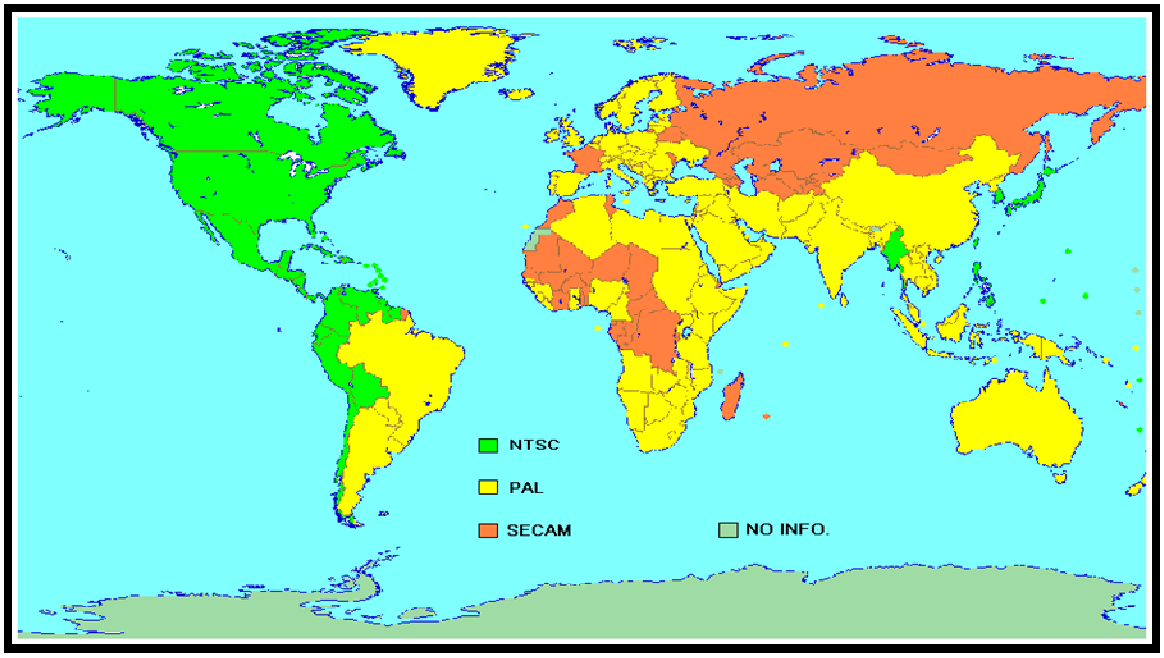


Figura II. 10. Normas y sistemas de televisión en el mundo

Una norma es un conjunto de parámetros adoptados como regla dentro de determinado grupo o región a fin de mantener una relación clara y sin ambigüedades entre las partes. En televisión es exactamente eso. Las normas establecen los parámetros que deben seguir tanto los equipos transmisores de señal como los receptores, a fin de que se establezca una comunicación segura y sin errores entre ambos. Entre los muchos parámetros que se fijan consideraremos solo los que atañen a nuestro proyecto.

Frecuencia Horizontal (H): frecuencia a la que se repiten las líneas de imagen.
Sincronismo Horizontal (H Sync): pulso que indica el comienzo de una línea de imagen.
Frecuencia Vertical (V): frecuencia a la que se repiten las imágenes (campos).
Sincronismo Vertical (V Sync.): pulso que indica el comienzo de una imagen (campo).
Líneas Horizontales: cantidad de líneas que forman una imagen completa(cuadro).
Subportadora de Color (SC): frecuencia a la que se modula para enviar lainformación de color.
“BURST” de Color: ráfaga de la SC que “sincroniza” la demodulación delcolor.
“Front Porch”: intervalo de señal sin información de imagen previo alH. Sync.
“Back Porch”: intervalo de señal posterior al H Sync., donde se colocael BURST.

2.5.1. Sistemas de televisión a color en uso alrededor del mundo

Hay tres TV-estándares importantes en todo el mundo: NTSC, SECAM y PAL*(ver figura)*

2.5.1.2. NTSC

Este sistema, adoptado por los Estados Unidos en 1954, consiste en que dos bandas de la misma frecuencia pero desplazadas 90 grados son moduladas en amplitud por las dos señales roja y azul, es de tipo simultáneo ya que emite al mismo tiempo la información concerniente a los tres colores primarios aditivos RGB, a partir de ellos componen todos los colores en la pantalla al mezclarlos en la proporción correcta. Es un Estándar del Comité Nacional de Sistemas de Televisión. Este comité estandariza el sistema profesional de color NTSC el cual es utilizado activamente en Estados Unidos, Japón, y otros países. Cuando la programación en color llegó a ser una posibilidad comercialmente se desarrolló el sistema NTSC para asegurar que el color pudiera ser aplicado en los aparatos de televisión de blanco y negro en uso en ese tiempo. Se presenta en 525 líneas a 60 ciclos por segundo.

2.5.1.3. PAL

Estatutos para el sistema de Fase de Línea Alternada, desarrollado por la empresa alemana Telefunken realizado por Walter Bruch, en el cual la información de matiz o tono y la saturación son transmitidas por modulación en cuadratura, conmutándose una de las modulaciones 180º de línea a línea de exploración en el transmisor; en el receptor se utiliza una línea de retardo para restaurar la correcta relación de fases de las dos modulaciones de retardo una de éstas en un tiempo igual al de la duración de una línea de exploración. Este sistema evita la distorsión de color que aparece en la recepción por NTSC. Se suele convenir que el sistema PAL es superior a NTSC debido a que es inmune a los problemas de reflexión de la señal en edificios u obstáculos. Los aparatos de televisión que utilizan PAL cuestan un poco más que los que utilizan NTSC debido a la necesidad de una línea de atraso 1H (un periodo de línea). Este sistema opera a 625 líneas a 50 ciclos por segundo y existe una variación PAL-M que opera a 60 ciclos por segundo.

2.5.1.4. SECAM

Abreviación para el sistema de Memoria de Secuencia de Color, el sistema francés creado por Henri De France, muy distinto de los sistemas anteriores, aquí la formación de color es transmitida secuencialmente (rojo menos luminancia R-Y seguida por azul menos luminancia B-Y, etc.) para cada línea converge por un subportador de frecuencia modulada que evita el aumento de distorsión durante la transmisión. Ofrece 819 líneas de resolución a 50 ciclos por segundo. Sus ventajas estriban en la mayor sencillez del aparato receptor y su inmunidad ante los problemas de fase que afectan al NTSC. Sin embargo, el SECAM no es totalmente compatible a los aparatos de blanco y negro y requiere de una línea de atraso 1H como en la recepción del sistema PAL... Existen de hecho dos variaciones del sistema SECAM, el horizontal y el vertical.

En el momento actual prácticamente todos los TV son multinorma (PALB, PALN, PALM, NTSC), o por lo menos trinorma (PALN, PALM, NTSC) y por supuesto con línea de retardo. Toda la sección de color se encuentra dentro del jungla de modo que por afuera sólo se pueden percibir los componentes más importantes como los cristales y la/s línea/s de retardo aunque ya existen líneas de retardo electrónicas con integrados de 8 patitas y por supuesto junglas que incluye la línea de retardo programable en su interior.

2.6. MICROCONTROLADORES

- Microcontrolador: todos los componentes básicos de un computador integrados en un único chip:
  - Unidad central de proceso.
  - Memoria (principal e incluso secundaria).
  - Unidades de entrada/salida.

Figura II. 11. Componentes básicos de un microcontrolador

- ¡Más microcontroladores que personas en el mundo!
  - ¿Cuantos microcontroladores llevas ahora mismo en los bolsillos? Teléfono móvil, llaves del coche, iPod, llave del garaje…
  - ¿Cuántos tienes en casa? Lavadora, microondas, televisor, teclado, ratón, …
  - ¿Y en el coche? Un coche moderno puede tener cerca de 100 microcontroladores

(airbags, tracción, ABS,…). En 2003, Microchip vendió 25 millones de microcontroladores PIC para el control de airbags.

2.6.1. Sistema empotrado/embebido (*Embedded System*): sistema informático cuyo hardware y software están concebidos para un uso concreto. Se componen de dos elementos:

Controlador.

Sistema a controlar.

Son sistemas informáticos embarcados en el propio sistema que van a controlar.

Figura II. 12. Ejemplos de sistemas embebidos

- En muchos sistemas empotrados es más práctico utilizar microcontroladores que microprocesadores:
  - Limitaciones de espacio todo en un único chip. Limitaciones de consumo los sistemas empotrados muchas veces son portátiles.
  - Limitaciones económicas es más barato tener un único chip que todo el hardware de un computador basado en microprocesador.
  - Las unidades de E/S de los microcontroladores están diseñadas para manejar sensores y actuadores típicos de los sistemas empotrados.

• Ejemplos: Lavadora, televisor, TDT, cajero automático, UVI móvil, caja del supermercado, misil, llaves del coche, GPS, boyas,…

2.6.2. CARACTERÍSTICAS DE LOS MICROCONTROLADORES

- Suelen utilizar arquitectura Harvard (memoria de datos e instrucciones separadas) Presenta ventajas en computadores que no hay que reprogramar constantemente.
- Dimensiones muy reducidas los que utilizaremos en las prácticas (encapsulado DIP) habitualmente sólo se usan para diseño de prototipos.
- Gran variedad dentro de una misma familia Adaptado a aplicaciones concretas:

- Control de motores (robots, sistemas mecánicos).
- Comunicaciones (WiFi, USB, Ethernet).
- Multimedia (Entradas A/D y Salidas D/A).
- Aplicaciones críticas (resistencia térmica, robustez).
- Instrumentación (sensores, LCD,…).

- Robustez (pantallazo azul en una lavadora o en un misil??) Watchdog Timer.
- Bajo consumo funcionamiento a pilas, sondas espaciales,… Dormido hasta recibir interrupción.
- Protección anti copia de la memoria de programa evitar ingeniería inversa y espionaje industrial.

2.6.3. ELECCIÓN DE UN MICROCONTROLADOR

- Obviando factores generales, como el precio o el tamaño, las características que nos ayudan a decidir entre la gran variedad de microcontroladores (PIC) son:
  - Número de bits: Arquitecturas de 8, 16 y 32 bits.
  - Velocidad: en MIPS o en MHz. De 4MHz a 80MHz.
  - Memoria de programa:
    • Tamaño: de 300Bs a 512KBs.
    • Tipo: Flash (regrabable) vs OTP (programable una única vez). ¿Auto escribible? ICSP.
  - Memoria principal: de 16Bs a 64KBs.
  - Memoria permanente: EEPROM de 0 a 4KBs.
  - Temperatura de trabajo: De -40ºC a 150ºC.
  - Voltaje: desde 1’8V.
  - Encapsulado: DIP/SPDIP para prototipos. Otros: SOIC, QFN, SSOP, TQFP, DFN,… Pines de E/S y pines totales: 4 a 85 / 6 a 100
  - Unidades de E/S y otros componentes:

- Conversores A/D.
- Puertos de comunicación digital (serie, RF, Ethernet, USB,…).

• Temporizadores, comparadores, PWM, capturadores,…

2.6.4. PRINCIPALES FAMILIAS DE MICROCONTROLADORES

- **Freescale:**División de semiconductores de Motorola.
  - 68HC705: ¿MCU más vendida de la historia? 8 bits, 20 pines, 112B RAM.
  - 68HC11: Familia de MCUs con arquitectura Von Neumann. Instrucciones CISC.
  - Coldfire: arquitectura de 32 bits. Hasta 128KBs datos, 512KBs programa, 500MHz, 120 pines.
- **Intel:**
  - Familias de 8 y 16 bits.
  - 8051: Muy extendido. Sirve como núcleo para MCUs de otros fabricantes (Atmel: mayores fabricantes de tarjetas inteligentes).
- **Texas Instruments, Zilog, Analog Devices…**
- **Microchip:**
  - PIC: MCUs de 8, 16 y 32 bits.
  - dsPIC: DSPs.
  - rfPIC: MCUs con unidad de RF.
  - ¿Por qué lo escogemos? Muy extendido, muchísima información, muy didáctico, alianza académica, muestras gratuitas, fácil programación (HW y SW), …

2.6.5. CAMPO DE APLICACIÓN DE LOS MICROCONTROLADORES

Figura II. 13. Ejemplos de Aplicaciones con microcontroladores

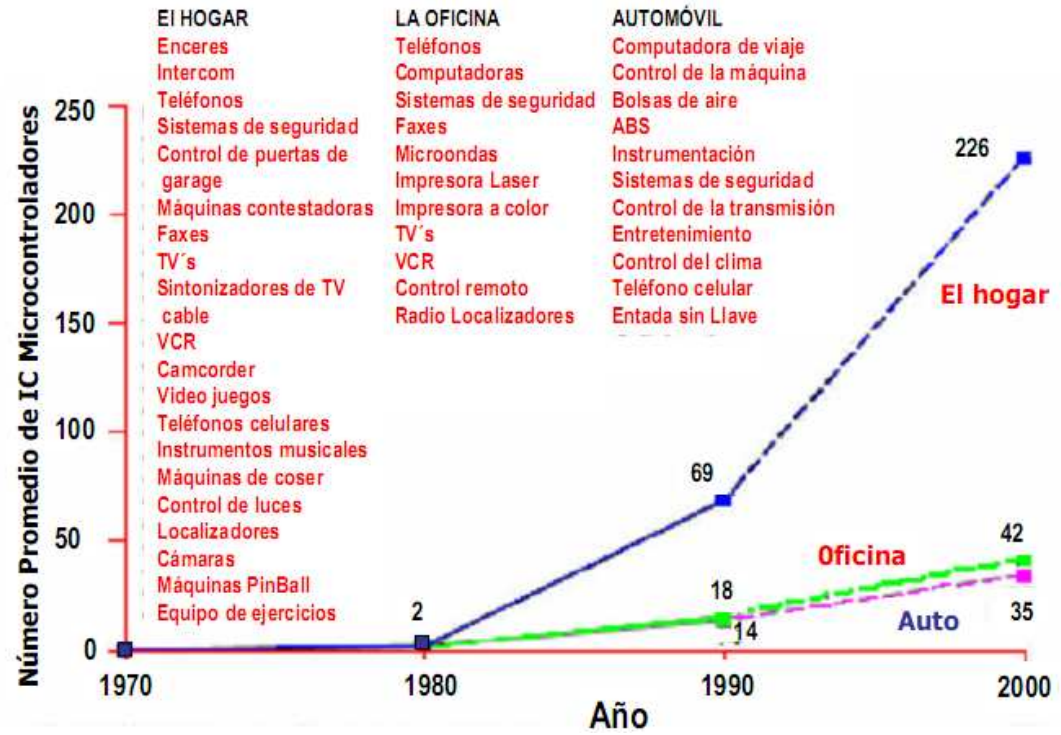


Figura II. 14. Promedio de microcontroladores utilizados en diferentes equipos

CAPITULO III
DISEÑO E IMPLEMENTACIÓN

3.1. PRINCIPIOS DE FUNCIONAMIENTO

Básicamente, la idea es direccionar secuencialmente una memoria y convertir el dato de la localidad en turnoen un nivel de voltaje de video, con el fin de desplegarlo en la pantalla de un televisor.

El inicio del despliegue es en la esquina superior izquierda de la pantalla.

Después de direccionar cierto número de localidades (número de puntos por línea horizontal) se detiene el direccionamiento de la memoria y se genera una señal llamada blanqueo, la cual en la pantalla es invisible.

Después de cierto tiempo de blanqueo se genera un pulso de sincronía horizontal, lo cual hace que se inicie en la televisión una nueva línea. Al cabo de cierto tiempo después del pulso de sincronía, se quita la señal de blanqueo y se continúa con el direccionamiento de la memoria en la siguiente línea.

Este proceso se repite cierto número de veces (líneas horizontales) hasta que se llega a la parte baja en la derecha de la pantalla, entonces se genera otro blanqueo y

posteriormente se genera un pulso de sincronía vertical, lo cual hace que el trazo en la pantalla inicie de nuevo su trayectoria en la parte superior izquierda de la pantalla. Este movimiento de arriba a la izquierda hasta abajo a la derecha en la pantalla se repite constantemente, variando la luminosidad del punto en la pantalla de acuerdo al dato contenido en la memoria.

Después de cada pulso vertical se reinicia el direccionamiento de la memoria de video.

3.1.1. Diagrama a bloques de la interfaz gráfica

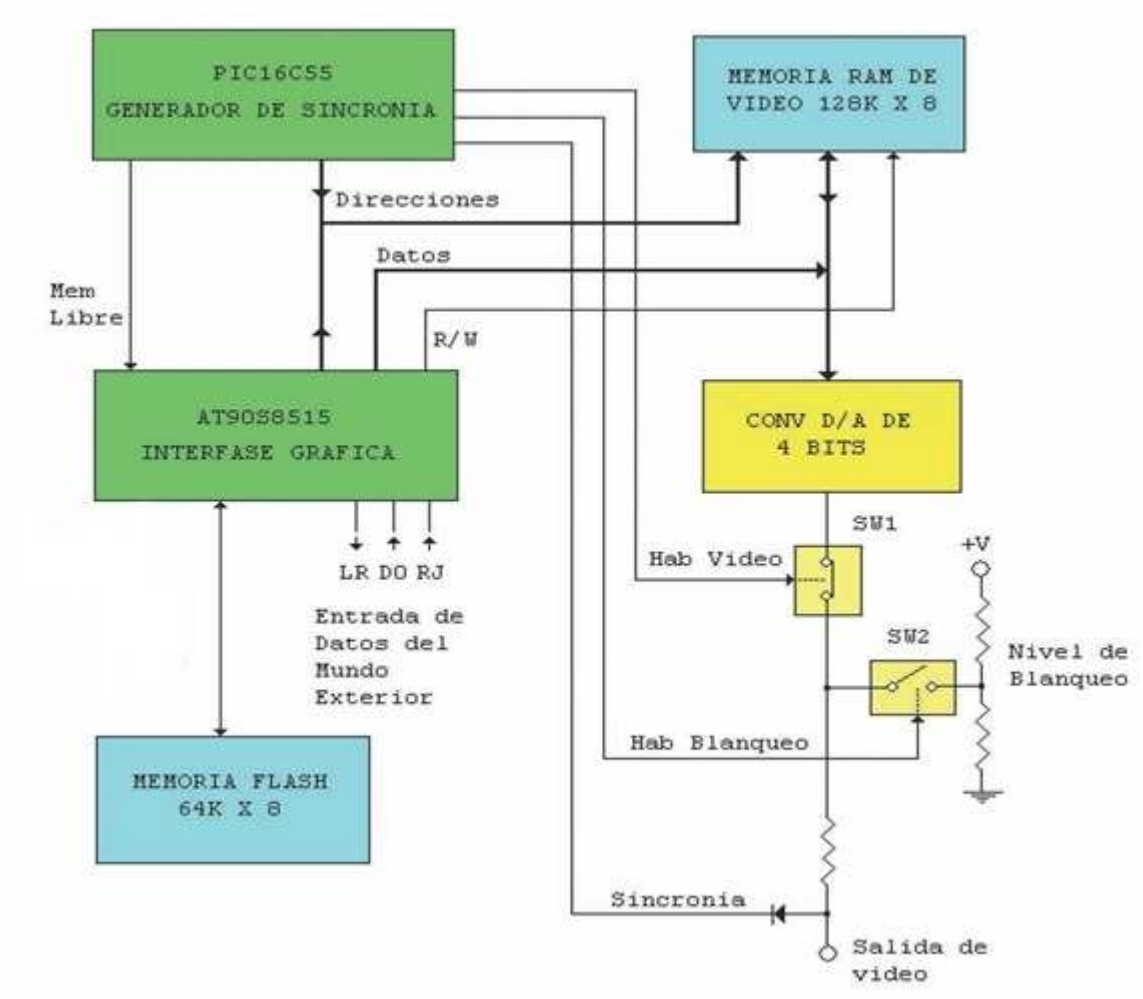


Figura III. 15. Diagrama a bloques de la interfaz gráfica

A continuación se explica el funcionamiento a bloques de la tarjeta:

La tarjeta consta de 2 microcontroladores, un PIC16C55 del conocido fabricante Microchip y otro de la serie Atmel (ATMega8515).

El PIC 16C55 se encarga del video. Cada punto a desplegar es direccionado por el bus “Direcciones” en la memoria RAM en orden creciente (0000-FFFF), cuya salida de datos son los bits de datos de la memoria (D0-D3)que son llevados al convertidor Digital-Analógico de 4 bitscon el fin de convertir a un nivel de voltaje el dato digital almacenado el cual genera, en la pantalla, una gama de grises de 16 tonos. Esta señal de video pasa a través del interruptor analógico SW1, el cual es controlado por el PIC.

Cuando se llega al final de cada línea horizontal o vertical, el SW1 se abre y, al mismo tiempo se cierra el SW2, enviando a la salida el nivel de voltaje de blanqueo, el cual es “más negro que el negro” en lo que a video se refiere, y en consecuencia es invisible en la pantalla.

En cuanto el PIC envía la señal de blanqueo (abriendo SW1 y cerrando SW2), se indica al microcontrolador 8515 a través de la señal “Memoria Libre”, que el bus de datos y direcciones de la memoria RAM están libres debido a que el PIC ha colocado sus salidas en alta impedancia para que puedan ser manipuladas por el procesador gráficoescriba lo que debe desplegarse en la pantalla. En cuanto el 16C55 vuelve a posesionarse de la memoria, se continúa con el barrido secuencial.

No podemos escribir en la memoria cuando el PIC esté presentando la señal en la pantalla, ya que alteraríamos el video al pretender que los 2 microcontroladores escriban en la memoria al mismo tiempo.

Durante la señal de blanqueo, el PIC además de poner sus salidas de direcciones y datos en alta impedancia, genera las señales apropiadas de sincronía horizontal y/o vertical, las cuales consisten en pulsos negativos que a través del diodo llevan la señal de video prácticamente a tierra, obteniéndose así el nivel bajo que se requiere para una sincronía en video.

Como se observa en el diagrama a bloques, el 8515 también controla la entrada a las memorias EEPROM (si se han colocado).

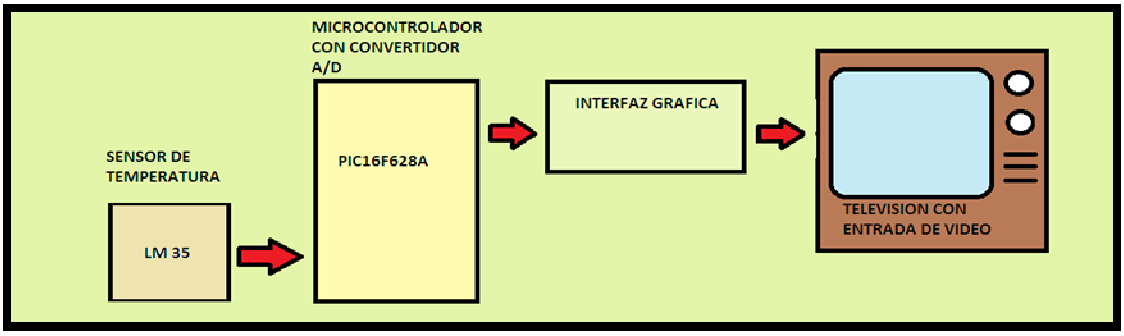


Figura III. 16. APLICACIÓN DE LA INTERFAZ GRÁFICA COMO TERMÓMETRO

La figura 2 muestra un diagrama en bloques con un ejemplo simple de una aplicación, en este caso un termómetro. El microcontrolador envía instrucciones a la tarjeta gráfica utilizando sólo 3 líneas (D0, RJ y LR) y la tarjeta envía la señal de video a una televisión para su despliegue.

3.1.2. Diagrama de flujo de funcionamiento

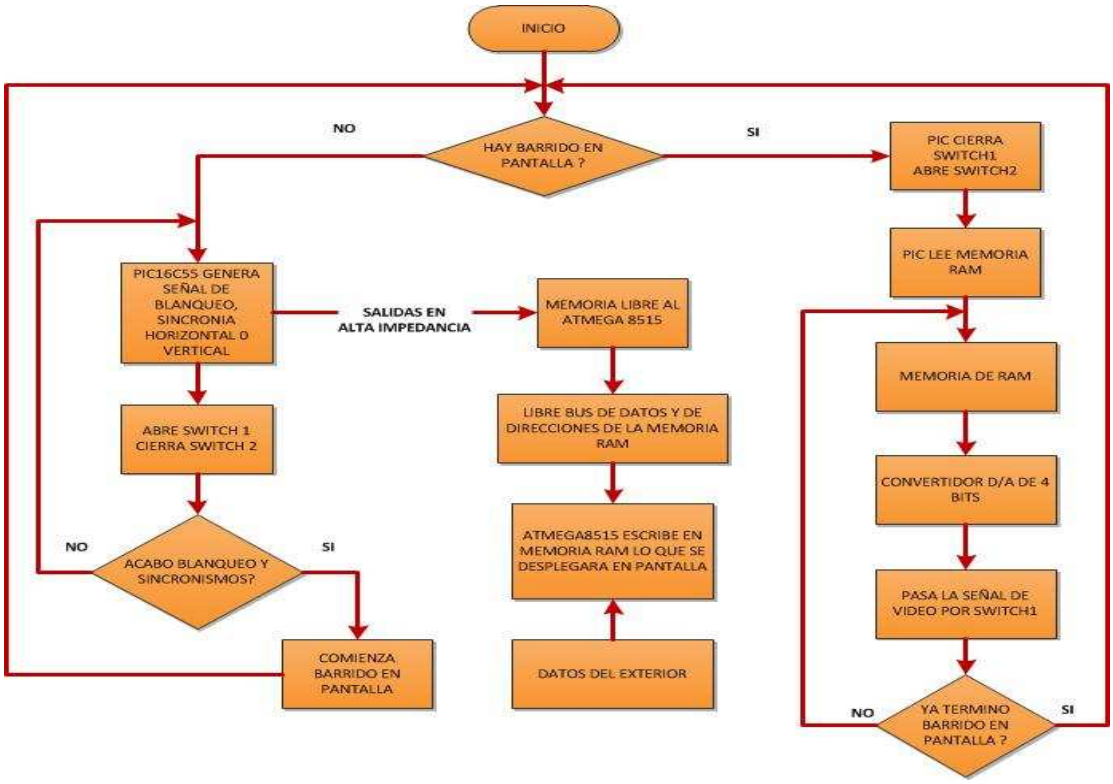


Figura III. 17. Diagrama de flujo de funcionamiento

3.2. ASPECTOS FISICOS DE LA INTERFAZ GRAFICA

La figura muestra una fotografía de la tarjeta IGTV, donde se pueden apreciar los dos microcontroladores utilizados, el ATMEL ATMega8515 y el PIC 16C55, la memoria RAM de video y dos memorias EEPROM de 8 patas cada una

El conector de interfaz, el cual es una base de 16 patas con la cual se comunica la interfaz gráfica con el mundo exterior.

También se puede observar el conector RCA para la salida de video.

El conector para la entrada de alimentación de 12 volts. Un interruptor analógico CMOS 4066.

También se han dispuesto 2 bases para memorias EEPROM del tipo serie, con el objeto de disponer de memoria no volátil en los proyectos que así lo requieran.

3.3. CARACTERISTICAS DE LA INTERFAZ

En realidad, la tarjeta es una verdadera microcomputadora con las siguientes características:

* 32kbytes de memoria RAM (volátil).
* Hasta 512k de memoria EEPROM (no volátil).
* Salida de video estándar para TV.
* Resolución de 228 x 216 pixeles con 16 tonos degris

El procesador gráfico utiliza menos de los 128kbytes de memoria RAM disponible, quedando cerca de 32kbytes libres, los cuales pueden ser usados por el microcontrolador externo en algún proyecto.

No en todos los proyectos se requiere de memoria EEPROM, la interfaz gráfica funciona sin la necesidad de este elemento, el cual puede ser agregados en el momento en que se hagan necesarios. Sin embargo, la memoria RAM de 32kbytes siempre estará disponible, ya que forma parte de la memoria de video la cual está integrada en la tarjeta.

3.4. EL HARDWARE MINIMO

Para poder generar una señal video, es necesario un mínimo hardware para poder generar niveles de señal entre 0 y 1V. Para conseguir un cuadro necesitaremos por lo menos 3 niveles. La TV necesita una sincronización y el nivel negro poder generar la señal video. Si deseas más que una imagen negra necesitarás un cierto nivel gris o blanco. Una cierta clase de convertidor digital a analógico es necesaria, por lo menos de 2 bits para conseguir diferentes niveles. La impedancia de entrada en una TV estándar es de 75 ohmios, y usando dos resistores y un CDA de 2 bit se puede crear (como en las imágenes abajo) gracias a la división del voltaje.

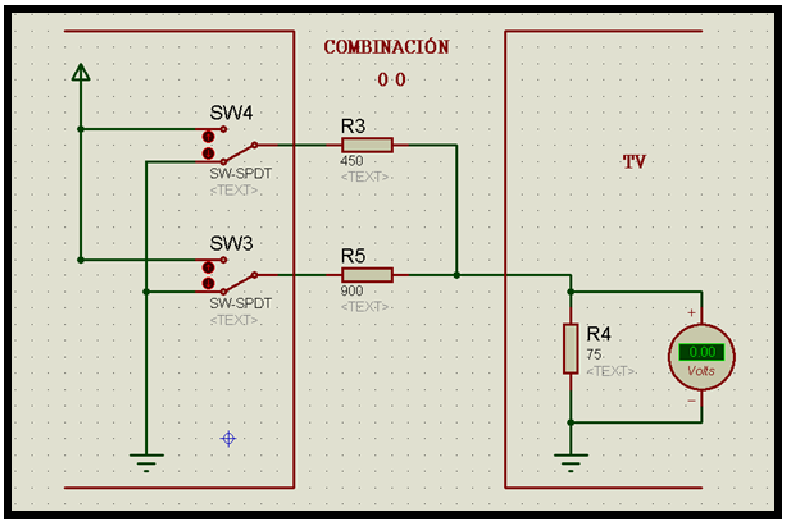


Figura III. 18. Nivel de sincronía

Sync

$$V_{in} = \frac{75\Omega}{75\Omega + \frac{450\Omega * 900\Omega}{450\Omega + 900\Omega}} * 0v = 0v$$

Conectando ambos D0 y D1 con la tierra, el voltaje en la entrada de la TV será 0v (nivel de la sinc.) porque no se conecta nada con VDD. (ver Figura)

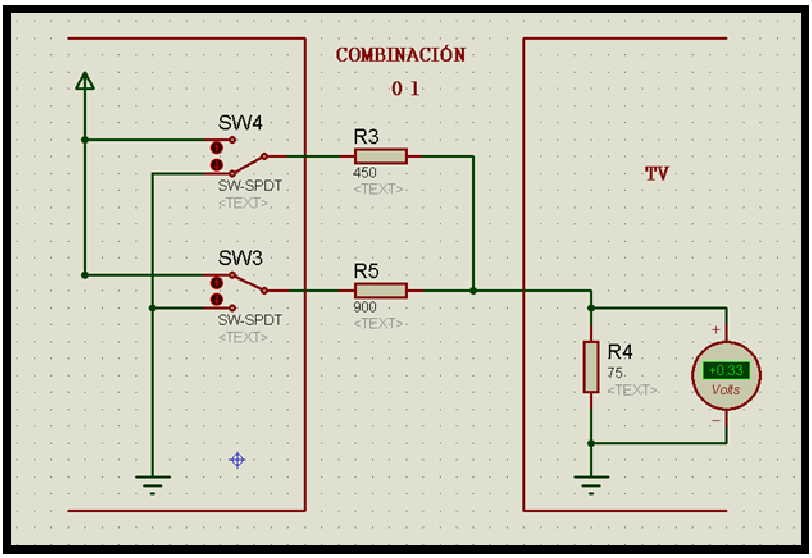


Figura III. 19. Nivel de negro

Negro

$$V_{in} = \frac{\frac{75\Omega * 450\Omega}{75\Omega + 450\Omega}}{900\Omega + \frac{75\Omega * 450\Omega}{75\Omega + 450\Omega}} * 5v = 0.33v$$

D1 que conecta a la tierra y D0 a 5v, pondrá el resistor 450ohm paralelamente a la impedancia de la entrada 75ohm de la TV, y con el resistor 900ohm conectado en serie, y lo agradece a la división del voltaje que éste genera 0.33v en la entrada de la TV, y ése es absolutamente negro. *(ver Figura)* (el nivel negro verdadero es 0.3v*)*

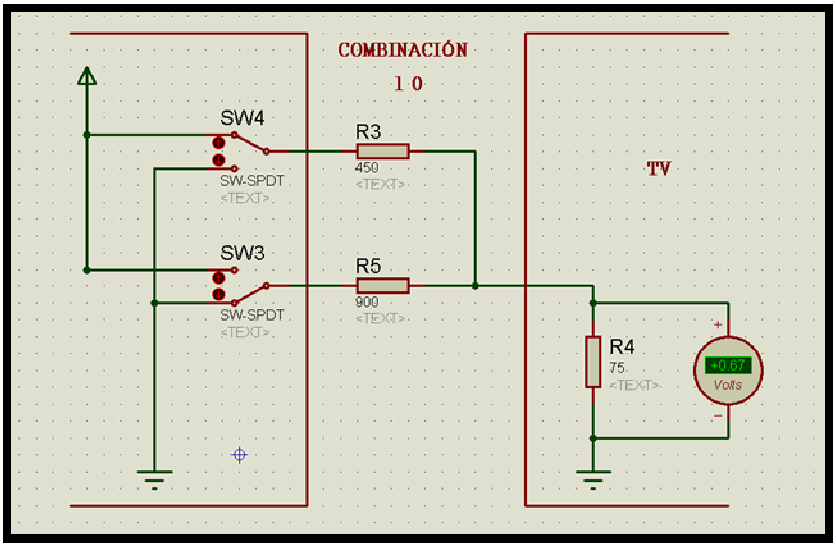


Figura III. 20. Nivel de gris

Gris

$$V_{in} = \frac{\frac{75\Omega * 900\Omega}{75\Omega + 900\Omega}}{450\Omega + \frac{75\Omega * 900\Omega}{75\Omega + 900\Omega}} * 5v = 0.67v$$

D1 que conecta a la tierra y D0 a 5v, pondrá el resistor 900ohm paralelamente a la impedancia de la entrada 75ohm de la TV, y con el resistor 450ohm conectado en serie, y lo agradece a la división del voltaje que éste genera 0.67v en la entrada de la TV, y ése es gris.*(ver Figura)*

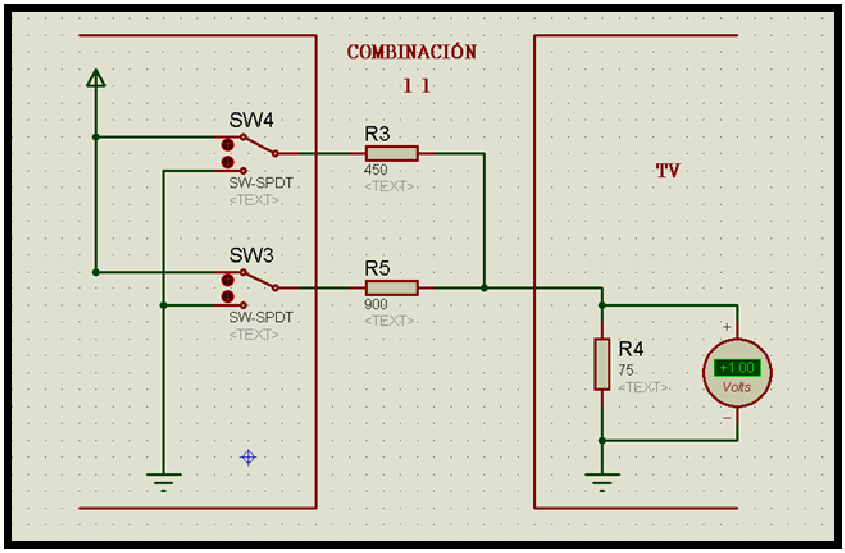


Figura III. 21. Nivel de blanco

Blanco

$$V_{in} = \frac{75\Omega}{75\Omega + \frac{450\Omega * 900\Omega}{450\Omega + 900\Omega}} * 5v = 1v$$

Conectar ambos D1 y D0 con 5v, pondrá el resistor 450ohm paralelamente al resistor de 900 ohmios, con la impedancia de la entrada 75ohm de la TV conectada en serie, y lo agradece a la división del voltaje que éste genera 1.0 en la entrada de la TV, y ése es nivel blanco.

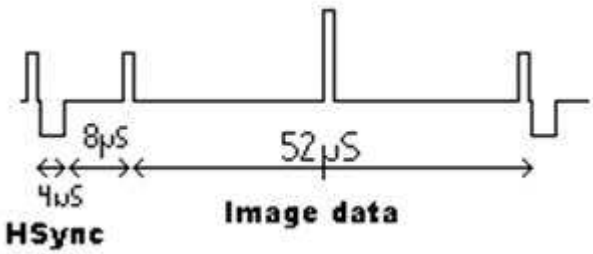
Con este circuito, cuatro niveles pueden ser creados. La imagen arriba muestra los circuitos equivalentes para los cuatro diversos niveles y cómo se crean los voltajes. Los valores del resistor no son críticos, se podrían utilizar los valores más estándares 470 y 1k en vez de 450 y de 900.

3.4.1. Software contra señales video generado por componentes físicos.

En un sistema video estándar como el de la tarjeta de un PC, la información gráfica sobre lo que se va a dibujar en la pantalla se toma de una memoria. Esto se hace automáticamente en hardware. Los pulsos de sincronización también se crean automáticamente en hardware. En un PIC16C55 hay una memoria de 24 bytes, y esa memoria se debe también utilizar para otros propósitos. No es posible almacenar la imagen entera en memoria como en tarjetas gráficas, la imagen tiene que hacerse generado barridos de la señal video a través de la pantalla. La generación de las señales video en software en un procesador tan simple es difícil, sólo las imágenes muy simples pueden ser creadas. La ventaja es que es absolutamente barata.

3.5. UNA LÍNEA, HACIENDO UNA BARRA VERTICAL.

Creando línea con la información del color gris-negro-blanco-negro-gris, y repitiendo la línea indefinidamente, una imagen se podía ver en una TV. La señal contiene un pulso horizontal, seguido por el retraso y de la información del color, así que la TV podría trabar la señal horizontalmente.



FiguraIII. 22. La señal de video generad aparecería esto en un osciloscopio. Los niveles bajos son las barras negras, y el nivel grande en el centro es la barra blanca.

Figura III. 23. La señal video generada aparecería esto sí estuvo entrada en una TV. Dos barras negras y una barra blanca. (La frontera marrón se supone para ser la TV)

3.6. CONECTOR PARA COMUNICACIÓN CON LA INTERFASE

Para la comunicación de la interfaz gráficacon el mundo exterior se cuenta con un conector DIP de 16 patas, el cual proporciona las 3 líneas básicas de comunicación, además de conexiones hacia las memorias EEPROM. Esto con el fin de que se pueda manipular estos elementos con cualquier microcontrolador externo sin intervención de la interfaz gráfica

También se dispone en el conector de 2 tensiones de alimentación: +5 y +12 V, con el fin de que la tarjeta interfaz gráfica pueda alimentar al circuito del microcontrolador externo, en el caso que así se desee.

A continuación se muestra el diagrama de conexiones del conector DIP.

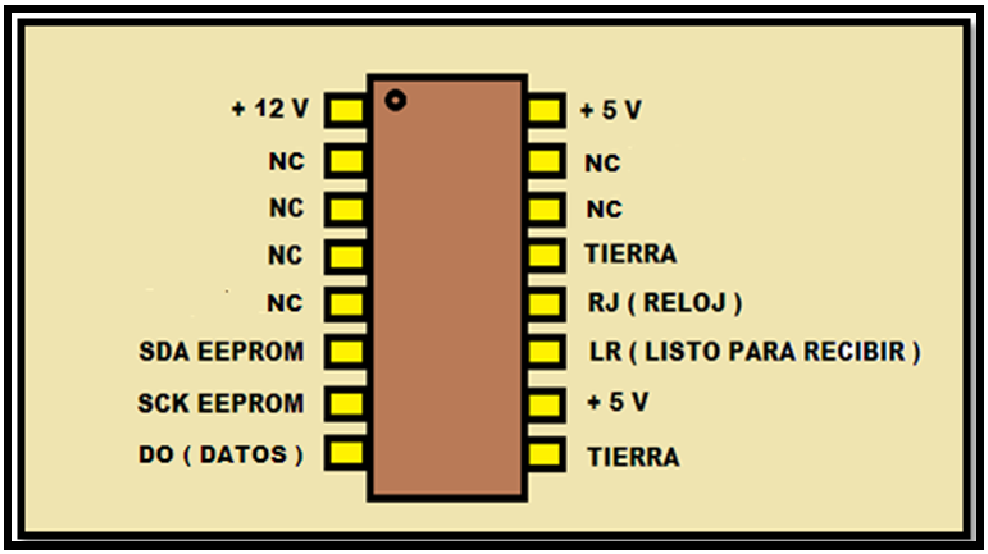


Figura III. 24. Conector de comunicación

3.7. PROTOCOLO DE COMUNICACION

Básicamente se usan solo 3 líneas para la comunicación con la interfaz gráfica**,** las cuales son: LR, D0 y RJ. La señal LR (Listo para Recibir) la envía la interfaz gráfica cuando está lista para recibir instrucciones del mundo exterior. Las instrucciones se envían por D0 en serie de manera síncrona, los pulsos de reloj se envían por RJ, los cuales son proporcionados por el microcontrolador externo junto con D0.

Las instrucciones se componen de uno o más bytes, por ejemplo, para borrar la pantalla se debe enviar un 0, en cuyo caso la instrucción consta de un solo byte.

Para dibujar un punto se debe enviar un 5 seguido de dos bytes indicando las coordenadas del punto (X, Y). Esta instrucción requiere de 3 bytes.

Para ilustrar con un ejemplo, supongamos que deseamos borrar la pantalla y luego dibujar un punto en el centro (coordenadas 114, 108). Los bytes que debemos enviar por D0 serían los siguientes:

Byte 1	0	Instrucción “Borrar Pantalla”.
Byte 2	5	Instrucción “Punto”.
Byte 3	114	Coordenada X del punto.
Byte 4	108	Coordenada Y del punto.

Después de que la interfaz gráficaejecuta éstas instrucciones, lo que vemos en la pantalla es un punto en el centro de la misma. El color del punto se establece con otra instrucción de 2 bytes.

3.8. DIAGRAMA DEL FLUJO DE DATOS

A continuación se muestra el flujo de datos en las 3 líneas al enviar el dato “8” (0001000) a la interface. Se debe enviar primero el bit más significativo. Observe que el pulso de RJ inicia cuando LR está en 1.

Como señala el dibujo, el bit en turno lo lee la interfaz gráfica en cada subida del RJ, el cual es enviado por el microcontrolador externo, junto con el valor del bit.

Mientras la interfaz gráfica procesa cada byte recibido, coloca LR en 0 para avisar que no se deben enviar más bytes. Al terminar de procesar la instrucción recibida, la avisa de nuevo que está lista para recibir más instrucciones colocando a 1 lógico la señal LR.

Es labor del microcontrolador externo revisar el estado de LR antes de enviar cada byte.

La velocidad máxima con la que se pueden enviar datos a la interfaz gráfica es de 1 megahertz, es decir, cada ciclo de RJ puede ser de1 useg, lo cual significa una velocidad de 1 megabit por segundo.

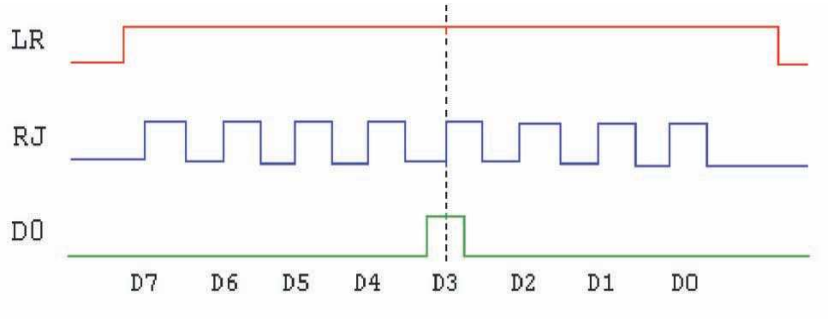


Figura III. 25. Diagrama de flujo de datos

La interfaz gráfica es capaz de dibujar puntos, líneas verticales y horizontales, cuadros, marcos, botones, caracteres, imágenes e inclusive tiene integrada una función para dibujar un display de cuarzo de hasta 8 dígitos simplemente enviando el código en 3 bytes en formato binario. La IG se encarga de hacer la conversión a decimal y desplegar el valor en la pantalla simulando displays de cuarzo.Se han implementado instrucciones para grabar y leer la memoria de video y aprovechar los 32kbytes (32K) libres en algún proyecto que requiera memoria RAM.

Para usar la tarjeta y diseñar equipo con ella, no se requiere gran conocimiento de lasseñales de televisión, ya que la tarjeta sólo espera instrucciones en forma de bytes y los circuitos internos se encargan del despliegue en la pantalla.

3.9. DESCRIPCION DE INTERGRADOS

3.9.1. PIC16C55

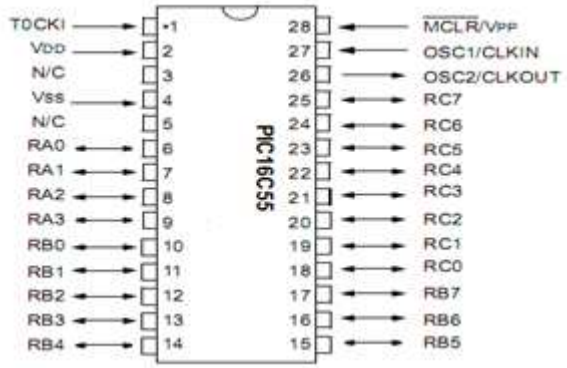


Figura III. 26. PIC16C55 tipo PDIP, SOIC, Windowed CERDIP

PIC16C55 de Microchip Technology pertenece a la familia de bajo costo, alto rendimiento, de 8 bits totalmente estática, EPROM ROM basado en CMOS de Microcontroladores. El PIC16C55 ofrece un rendimiento en un orden de magnitud superior a sus competidores en la misma categoría de precios. Las instrucciones de 12 bits de ancho son muy paralela que ha desembocado en la compresión de código de 2:01 con respecto a otros microcontroladores de 8 bits de su clase. Es fácil de usar y fácil de recordar el conjunto de instrucciones reduce significativamente el tiempo de desarrollo.

Características

- Se emplea una arquitectura RISC con sólo 33 palabras / instrucciones de ciclo único.
- Todas las instrucciones son un solo ciclo, excepto para las de salto del programa que tienen dos ciclos.
- Velocidad de operación: DC – 40 MHz reloj de entrada
  DC – 100 ns ciclo de instrucción
- Pins : 28
- I/O : 20
- EPROM/ROM : 512

- RAM : 24

Características de los periféricos

- Relojen tiempo realde 8bits/contador(TMR0) conprescalerprogramablede 8bits
- Power-onReset(POR)
- El dispositivo tiempo de reinicio(DRT)
- WatchdogTimer(WDT) consupropiochipRCosciladorparaun funcionamiento fiable
- Código de Protecciónprogramable
- Ahorro de energíamodoSLEEP
- Selección deopcionesdel oscilador:
  -RC: RCosciladorde bajo coste

  -XT: Cristalestándar/resonador

  - HS: Cristalde altavelocidad/resonador

  -LP: Ahorro de energía, el cristal de baja frecuencia

3.9.2. ATmega 8515

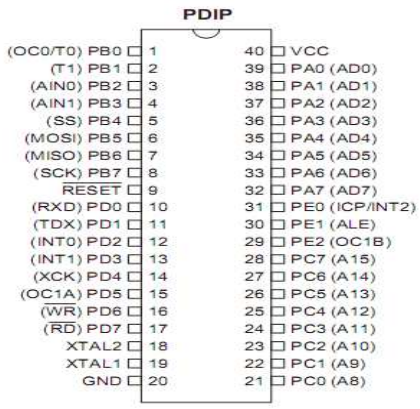


Figura III. 27. Atmega 8515, tipo PDIP

El ATMEGA8515 es un microcontrolador de baja potencia CMOS de 8 bits basado en el AVR mejorado la arquitectura RISC. Mediante la ejecución de instrucciones de gran alcance en un solo ciclo de reloj, el ATMEGA8515 logra tasas de transferencia se acerca 1 MIPS por MHz que permite al diseñador del sistema optimizar el consumo de energía en comparación con la velocidad de procesamiento.

Características

- Arquitectura basada en el 80C51 con CPU de 8 bits.
- Procesador booleano con soporte de operación sobre bits.
- 512 Bytes EEPROM
- Sin memoria de programa interna ROM (*caso 80C32*).
- 512 Bytes Internal SRAM
- Tres contadores temporizadores de 16-bit (*counter/timers*).
- Un canal serie asíncrono Full duplex para las comunicaciones RS-232.
- Capacidad de direccionamiento de memoria 64k ROM y 64k RAM.
- Modos de funcionamiento en bajo consumo (*Idle mode Power-down mode*).
- Compatible con las tecnologías digitales CMOS y TTL.
- Frecuencias de trabajo de reloj comprendida ente 3,5 y 33MHz.
- Oscilador interno.
- 6 fuentes de interrupción con distintos niveles de prioridad.
- 2 interrupciones externas.
- 3 interrupciones para los contadores/temporizadores (*timers*).
- 1 interrupción para el puerto serie.

3.9.3. CD4066

El integrado en sí, es un cuádruple interruptor Bilateral, diseñado para la transmisión o multiplexado de señales digitales o analógicas, puedes considerarlo como un interruptor de conmutación, cada uno de los interruptores dispone de un pin de control y dos pines de entrada/salida, una representación de lo que se vería por dentro sería algo así *(ver figura*III. 28*)*

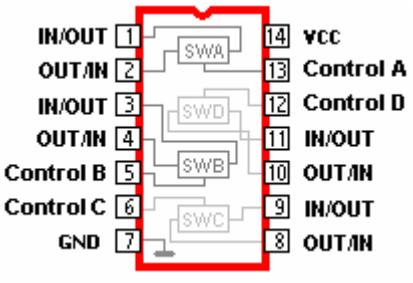


Figura III. 28. CD4066, PDIP

Este integrado puede ser sustituido por un CD4016 y es capaz de realizar las mismas funciones.

Que sea bidireccional significa que cualquiera de los dos pines de cada interruptor exceptuando al pin de control, puede hacer de entrada mientras el otro es de salida. Cada interruptor entra en conducción cuando se presenta un nivel alto (superior al 70% de VCC), y en corte cuando se envíe un nivel bajo (inferior al 30% de VCC) por el mismo pin de control.

En nuestro proyecto es convertido en un interruptor analógico por donde pasa la señal de video y este es controlado por el pic que abre y cierra dichos interruptores para proporcionar los sincronismos necesarios

3.9.4. HM628128D

HM628128D de Hitachi es una memoria RAM de 1 M-bit estática,organizada 131,072-kword · 8-bit de alta densidad, mayor rendimiento y bajo consumo mediante el empleo de la tecnología Hi - CMOS proceso

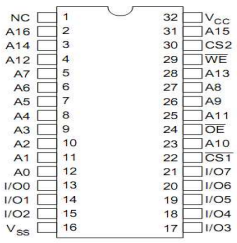


Figura III. 29. Memoria HM628128

Características

• Solo 5 Vde alimentación:5V-10%

•Tiempo de acceso: 55nsns/70(max)

• Potencia dedisipación

Activo: 30mW/MHz(típico) En espera: 10 µW(típico)

•Completamentememoriaestática.

Norelojotiempo requerido

•La igualdad de accesoytiempos de ciclo

•Entraday salidade datoscomún

Tresestado dela salida

•Todaslas entradas directamentecompatible con TTL

•Funcionamiento con bateríasde respaldo

2Selección dechipparacopias de seguridadde la batería

Pin name	Function
A0 to A16	Address input
I/O0 to I/O7	Data input/output
$\overline{CS1}$	Chip select 1
CS2	Chip select 2
$\overline{WE}$	Write enable
$\overline{OE}$	Output enable
V_{CC}	Power supply
V_{SS}	Ground
NC	No connection

Tabla III.1. Descripción de pines HM628128D

CAPITULO IV

CODIGO DE INSTRUCCIONES

==========

4.1. INSTRUCCIONES GRAFICAS BASICAS

4.1.1. BORRAR PANTALLA

CODIGO

0

Llena la pantalla con el color actual del fondo.

4.1.2. PANTALLA DE INICIO

CODIGO

1

Dibuja la pantalla principal, la cual muestra 8 barras de diferentes tonos de gris, desde el negro hasta el blanco. También muestra una indicación en caso de que se encuentren instaladas las memorias EEPROM. Esta pantalla es la que aparece al encender la IG.

4.1.3. FONDO

CODIGO	BYTE 1
2	Color

Fija el color del fondo, el cual viene indicado por el Byte 1. Los valores pueden ser de 0 a 15.

4.1.4. BRILLO

CODIGO	BYTE 1
3	Color

Fija el color del frente (Brillo), el cual viene indicado por el Byte 1. Los valores pueden ser de 0 a 15.

4.1.5. ANCHO

CODIGO	BYTE 1
4	A

Fija el ancho del trazado de las líneas horizontales y verticales. Los valores pueden ser del 0 al 255. Un valor de 0 significa un ancho de 256 pixeles.

4.1.6. CURSOR DE DIBUJO

CODIGO	BYTE 1	BYTE 2
5	X	Y

Fija una nueva posición del cursor de dibujo en los puntos X, Y, los cuales vienen indicados por los Bytes 1 y 2. Las siguientes instrucciones gráficas que hagan uso de la posición del cursor de dibujo (Línea, Caracter etc.) se dibujarán a partir de éste punto (X, Y).

4.1.7. PUNTO

CODIGO	BYTE 1	BYTE 2
6	X1	Y1

Dibuja un punto del color del Brillo actual en las coordenadas de pantalla X1, Y1, indicadas en los Bytes 1 y 2. Las coordenadas actuales X, Y no se afectan con la instrucción.

4.1.8. LINEA HORIZONTAL

CODIGO	BYTE 1
7	X2

Dibuja una línea horizontal del color actual del Brillo, iniciando en el valor actual del cursor X y terminando en el valor indicado por el Byte 1, el cual será X2. Al finalizar la instrucción X tendrá el valor de X2.

4.1.9. LINEA VERTICAL

CODIGO	BYTE 1
8	Y2

Dibuja una línea vertical del color actual del Brillo, iniciando en el valor actual del cursor Y y terminando en el valor indicado por el Byte 1, el cual será Y2. Al finalizar la instrucción Y tendrá el valor de Y2.

4.1.10. CUADRO

CODIGO	BYTE 1	BYTE 2
9	X2	Y2

Dibuja un cuadro vacío del color actual del Brillo cuya esquina superior izquierda es el valor actual del cursor (X, Y) y la esquina inferior derecha viene indicada por los Bytes 1 y 2 en el punto X2, Y2. Al finalizar la instrucción el valor del cursor (X, Y) no se altera.

4.1.11. CUADRO LLENO

CODIGO	BYTE 1	BYTE 2
10	X2	Y2

Dibuja un cuadro lleno del color actual del Brillo cuya esquina superior izquierda es el valor actual del cursor (X, Y) y la esquina inferior derecha viene indicada por los Bytes 1 y 2 en el punto X2, Y2. Al finalizar la instrucción el valor del cursor (X, Y) no se altera.

4.1.12. MARCO ALTO RELIEVE

CODIGO	BYTE 1	BYTE 2
11	X2	Y2

Dibuja un marco en alto relieve con colores predefinidos en los bordes. La esquina superior izquierda corresponde al valor actual del cursor X, Y y la esquina inferior derecha viene indicada por los Bytes 1 y 2 en el punto X2, Y2. Al finalizar la instrucción Y tendrá el valor de Y2. Al finalizar la instrucción el valor del cursor (X, Y) no se altera.

4.1.13. MARCO BAJO RELIEVE

CODIGO	BYTE 1	BYTE 2
12	X2	Y2

Dibuja un marco en bajo relieve con colores predefinidos en los bordes. La esquina superior izquierda corresponde al valor actual del cursor X, Y y la esquina inferior derecha viene indicada por los Bytes 1 y 2 en el punto X2, Y2. Al finalizar la instrucción Y tendrá el valor de Y2. Al finalizar la instrucción el valor del cursor (X, Y) no se altera.

4.1.14. CARACTER

CODIGO	BYTE 1
13	Car

Dibuja un caracter ASCII en el valor actual del cursor X, Y. El caracter viene especificado en código ASCII por el BYTE 1.

4.1.15. TEXTO CON FORMATO

CODIGO	BYTE 1	BYTE 2		BYTE n
14	Car1	Car2	…	255

Coloca texto en la pantalla comenzando en la posición actual del cursor X, Y. El texto se muestra con formato, es decir, se pueden introducir caracteres de control tales como salto de línea, tabuladores etc. El último Byte debe ser 255 para indicar el fin de la cadena. El Byte 1 y los siguientes (con excepción del último) consisten en los caracteres que se desplegarán, incluyendo caracteres de control. En el 4.9...1.1 se explica detalladamente ésta instrucción.

4.1.16. PUNTO INVISIBLE

CODIGO	BYTE 1
15	Inv

Hace invisible el punto decimal en las instrucciones DESPLEGAR DATO EN DISPLAY DE CUARZO y DESPLEGAR DATO EN DECIMAL.
Si Inv=1, el punto decimal no se muestra. Si Inv=0, el punto decimal se mostrará.

==========

4.2. INSTRUCCIONES GRAFICAS DE IMÁGENES

4.2.1. IMAGEN 2 TONOS

CODIGO	BYTE 1	BYTE 2	BYTE 3		BYTE m+2
32	npH	npV	B1	….	Bm

Dibuja una imagen de 2 tonos de gris (Brillo y Fondo). El número de puntos horizontales viene indicado por el BYTE 1 y el número de puntos verticales por el BYTE 2. Después de indicar el número de puntos (npH x npV) se envía la imagen fila por fila, donde cada Byte contiene la información de 8 puntos. Un Bit 1 indicará un punto del color del Brillo y un Bit 0 un punto del color del Fondo. El número de puntos horizontales debe ser un múltiplo de 8, ya que se grafica por bytes. En el capítulo 4.9.2.1 se explica detalladamente ésta instrucción.

4.2.2. IMAGEN 16 TONOS

CODIGO	BYTE 1	BYTE 2	BYTE 3		BYTE m+2
33	npH	npV	B1	….	Bm

Dibuja una imagen de 16 tonos de gris. El número de puntos horizontales viene indicado por el BYTE 1 y el número de puntos verticales por el BYTE 2. Después de indicar el número de puntos (npH x npV) se envía la imagen fila por fila, donde cada Byte contiene la información del color de 1 punto. El color puede ser del 0 al 15. En el capítulo 4.9.2.2.Se explica detalladamente ésta instrucción.

==========

4.3. INSTRUCCIONES GRAFICAS DE CONTROLES

4.3.1. DIBUJAR BOTONES DE CONTROL

CODIGO	BYTE 1
64	N

Dibuja N botones en la parte inferior de la pantalla los cuales proporcionan un aspecto visual uniforme para cada proyecto desarrollado para la **IG**. El máximo valor de N es 5. En el capítulo 4.9.3.1.Se explica detalladamente ésta instrucción.

4.3.2. PRESIONARBOTON DE CONTROL

CODIGO	BYTE 1
65	Bot

Muestra el Botón de Control presionado durante 250 milisegundos, después de los cuales regresa a su estado normal. Bot indica el número de botón y puede ser del 1 al 5.

4.3.3. CUADRO DE DIALOGO

CODIGO	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5
66	X	Y	X2	Y2	Tipo Bot

BYTE 6	BYTE n+5	BYTE k	BYTE m
CarTit1 ….	CarTit n ….	CarCont1 ….	255

Muestra un cuadro de diálogo en el cual se inserta Texto con Formato para mostrar

texto y botones de control.

El tamaño del cuadro de diálogo viene especificado por los bytes del 1 al 4. El byte 5 (Tipo Bot) indica qué tipo de botones se dibujarán, los cuales pueden ser Aceptar, Cancelar Si y No.

A continuación vienen los caracteres de la barra de título y a partir del byte k vienen los caracteres del Texto con Formato. En el capítulo 4.9.3.1 se explica detalladamente ésta instrucción.

4.3.4. PRESIONAR BOTON “ACEPTAR”

CODIGO

67

En un cuadro de Diálogo, el botón marcado con “ACEPTAR” se muestra presionado durante 250 milisegundos, después de los cuales regresa a su estado normal.

4.3.5. PRESIONAR BOTON “CANCELAR”

CODIGO

68

En un cuadro de Diálogo, el botón marcado con “CANCELAR” se muestra presionado durante 250 milisegundos, después de los cuales regresa a su estado normal.

4.3.6. PRESIONAR BOTON “SI”

CODIGO

69

En un cuadro de Diálogo, el botón marcado con “SI” se muestra presionado durante 250 milisegundos, después de los cuales regresa a su estado normal.

4.3.7. PRESIONAR BOTON “NO”

CODIGO
70

En un cuadro de Diálogo, el botón marcado con “NO” se muestra presionado durante 250 milisegundos, después de los cuales regresa a su estado normal.

==========

4.4. INSTRUCCIONES GRAFICAS PARA DESPLIEGUE DE DATOS

4.4.1. DESPLEGAR DATO EN DISPLAY DE CUARZO EN DECIMAL

CODIGO	BYTE 1	BYTE 2	BYTE 3	BYTE 4
96	BF	B2	B1	B0

Despliega el dato binario indicado por los Bytes B2, B1 y B0 en un display en forma de cuarzo.
El display aparecerá en las coordenadas actuales X, Y. La máxima cantidad que puede ser mostrada es 16777215, que corresponde a un dato de 3 bytes. B0 es el byte menos significativo.
BF se usa para indicar el signo, el número de dígitos a mostrar y la posición del punto. En el capítulo 4.9.4.1.Se explica detalladamente ésta instrucción.

4.4.2. DESPLEGAR DATO EN DECIMAL

CODIGO	BYTE 1	BYTE 2	BYTE 3
97	B2	B1	B0

Despliega el dato binario indicado por los Bytes B2, B1 y B0 en formato decimal.
Los números aparecerán en las coordenadas actuales X, Y. La máxima cantidad que puede ser mostrada es 16777215, que corresponde a un dato de 3 Bytes. B0 es el byte menos significativo.

4.4.3. DESPLEGAR DATO EN HEXADECIMAL

CODIGO	BYTE 1	BYTE 2	BYTE 3
98	B2	B1	B0

Despliega el dato binario indicado por los Bytes B2, B1 y B0 en formato hexadecimal.
Los caracteres aparecerán en las coordenadas actuales X, Y. La máxima cantidad que puede ser mostrada es FFFFFF, que corresponde a un dato de 3 Bytes. B0 es el byte menos significativo.

4.4.4. DESPLEGAR DATO EN BINARIO

CODIGO	BYTE 1	BYTE 2	BYTE 3
99	B2	B1	B0

Despliega el dato binario indicado por los Bytes B2, B1 y B0 en formato binario.
Los caracteres aparecerán en las coordenadas actuales X, Y. La máxima cantidad que puede ser mostrada es 11111111 11111111 11111111, que corresponde a un dato de 3 Bytes. B0 es el byte menos significativo.

4.4.5. DESPLEGAR CIFRA NUMERICA EN DECIMAL

CODIGO	BYTE 1	BYTE 2	BYTE 3	BYTE 4
100	BF	B2	B1	B0

Despliega una cifra en formato decimal con punto y signo.
Esta instrucción es semejante al display de cuarzo, solo que los dígitos son caracteres normales de texto. Todas las reglas y formato aplicables al display de cuarzo se aplican a ésta instrucción.

4.4.6. DESPLEGAR SOBRE FLUJO

CODIGO
101

Despliega indicador de “Sobre Flujo” para un display de cuarzo en las coordenadas actuales X, Y.

4.4.7. GRAFICAR PASO X

CODIGO	BYTE 1
102	nBytes

Grafica una secuencia de datos almacenados en memoria, los cuales se transformarán en la pantalla en una curva representativa de tales datos. Con esta instrucción se puede implementar un osciloscopio digital simplemente enviando los datos muestreados a la **IG** y luego ejecutando ésta instrucción.
La **IG** se encargará de graficarlos y mostrarlos en la pantalla a una velocidad suficientemente rápida como para simular un osciloscopio analógico. Se comenzará a

graficar en la posición actual del cursor X. La posición Y viene indicada por el primer punto que se dibujará. Antes de ejecutar ésta instrucción, los datos a graficar deben estar almacenados en la memoria interna de la **IG**. Esto se consigue ejecutando previamente la instrucción que se usa para transferir datos provenientes del exterior a ésta memoria. nBytes es el número de bytes que se graficarán. En el capítulo 4.9.4.2 se explica detalladamente ésta instrucción.

4.4.8. GRAFICAR LINEA BYTE

CODIGO	BYTE 1	BYTE 2	BYTE 3
103	nBits	TamH	TamV

Dibuja líneas horizontales y verticales las cuales representan datos binarios (unos y ceros) en forma gráfica. Con ésta instrucción se puede representar gráficamente en la pantalla una secuencia de datos binarios. Se comenzará a graficar en la posición actual del cursor X, Y. El número de bits a graficar viene indicado por el BYTE 1 en nBits. TamH indica el tamaño de las líneas horizontales y TamV el de las verticales. En el capítulo 4.9.4.3 se explica detalladamente ésta instrucción.

4.4.9. GRAFICAR LINEA BIT

CODIGO	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5
104	nBytes	TamH	TamV	Sep	bits

Dibuja líneas horizontales y verticales las cuales representan datos binarios (unos y ceros) en forma gráfica. La diferencia con la instrucción anterior es que ésta dibuja las gráficas binarias en forma vertical, es decir, para graficar un byte se comenzará dibujando primero el bit menos significativo (b0), luego hacia abajo se dibujará cada bit del mismo byte (b1, b2 … b7), aunque el parámetro bits indica cuantos bits del byte se deben dibujar. Sep indica cuantos pixeles de separación existirán entre cada trazo. Se comenzará a graficar en la posición actual del cursor X, Y. En el capítulo 4.9.4.4.

Seexplica detalladamente ésta instrucción.

==========

4.5. INSTRUCCIONES DE MEMORIA INTERNA

4.5.1. FIJAR APUNTADOR DE MEMORIA INTERNA

CODIGO	BYTE 1
128	Loc

La memoria interna de la **IG** consta de 256 bytes. Para accesarla se debe fijar un apuntador el cual se incrementa automáticamente cada vez que se lee o escribe un byte.
Loc indica la localidad que se fijará como apuntador y puede tomar valores de 0 a 255.

4.5.2. ESCRIBIR BYTE EN MEMORIA INTERNA

CODIGO	BYTE 1
129	B

La **IG** Escribe el byte B en la localidad de la memoria RAM interna indicada por el apuntador.

4.5.3. ESCRIBIR BLOQUE EN MEMORIA INTERNA

CODIGO	BYTE 1	BYTE 2		BYTE n+1
130	nBytes	B1	….	Bn

La **IG** Escribe la cantidad de bytes indicada por nBytes en la localidad de la memoria RAM interna indicada por el apuntador. B1 a Bn son los bytes que se escribirán.

4.5.4. LEER BYTE DE MEMORIA INTERNA

CODIGO
131

RESPUESTA DE LA IG

BYTE 1
B

La **IG** lee de la memoria RAM interna el byte indicado por el apuntador y lo envía al exterior.

4.5.5. LEER BLOQUE DE MEMORIA INTERNA

CODIGO	BYTE 1
132	nBytes

RESPUESTA DE LA IG

BYTE 1		BYTE n
B1	….	Bn

La **IG** lee de la memoria RAM interna la cantidad de bytes indicada por nBbytes (a partir de la localidad señalada por el apuntador) y los envía al exterior.

==========

4.6. INSTRUCCIONES DE MEMORIA DE PÁGINA

4.6.1. FIJAR APUNTADOR DE MEMORIA DE PÁGINA

CODIGO	BYTE 1	BYTE 2
133	LocH	LocL

La memoria de página de la **IG** consta de 32768 Bytes (32 K). Para accesarla se debe fijar un apuntador el cual se incrementa automáticamente cada vez que se lee o escribe un byte.
En los bytes LocH y LocL se indica la localidad que se fijará. LocH es el Byte más significativo.

4.6.2. ESCRIBIR BYTE EN MEMORIA DE PÁGINA

CODIGO	BYTE 1
134	B

La **IG** Escribe el byte B en la localidad de la memoria RAM de página indicada por el apuntador.

4.6.3. ESCRIBIR BLOQUE EN MEMORIA DE PÁGINA

CODIGO	BYTE 1	BYTE 2	BYTE 3	…	BYTE n+2
135	nBytesH	nBytesL	B1		Bn

La **IG** Escribe la cantidad de bytes indicada por nBytesH y nBytesL en la localidad de la memoria RAM de página indicada por el apuntador. B1 a Bn son los bytes que se escribirán.

4.6.4. LEER BYTE DE MEMORIA DE PÁGINA

CODIGO
136

RESPUESTA DE LA IG

BYTE 1
B

La **IG** lee de la memoria de página el byte indicado por el apuntador y lo envía al exterior.

4.6.5. LEER BLOQUE DE MEMORIA DE PÁGINA

CODIGO	BYTE 1	BYTE 2
137	nBytesH	nBytesL

RESPUESTA DE LA IG

BYTE 1		BYTE n
B1	….	Bn

La **IG** lee de la memoria RAM de página la cantidad de bytes indicada por nBytesH y nBytesL (a partir de la localidad señalada por el apuntador) y los envía al exterior.

==========

4.7. INSTRUCCIONES DE MEMORIA EEPROM

4.7.1. FIJAR APUNTADOR DE MEMORIA EEPROM

CODIGO	BYTE 1	BYTE 2
138	LocH	LocL

La memoria EEPROM de la **IG** es opcional y consta de hasta 2 integrados de 16384 (16 K) de memoria EEPROM cada uno, los cuales pueden proporcionar hasta 32768 bytes (32 K) de capacidad. Para accesar la memoria se debe fijar un apuntador el cual se incrementa automáticamente cada vez que se lee o escribe un byte. En los bytes LocH y LocL se indica la localidad del apuntador. LocH es el byte más significativo

---

4.7.2. ESCRIBIR BYTE EN MEMORIA EEPROM

CODIGO	BYTE 1
139	B

La **IG** Escribe el byte B de la instrucción en la localidad de la memoria EEPROM indicada por el apuntador.

4.7.3. ESCRIBIR BLOQUE EN MEMORIA EEPROM

CODIGO	BYTE 1	BYTE 2	BYTE 3	…	BYTE n+2
140	nBytesH	nBytesL	B1		Bn

La **IG** Escribe la cantidad de bytes indicada por nBytesH y nBytesL en la localidad de la memoria EEPROM indicada por el apuntador. B1 a Bn son los bytes que se escribirán.

4.7.4. LEER BYTE DE MEMORIA EEPROM

CODIGO
141

RESPUESTA DE LA IG

BYTE 1
B

La **IG** lee de la memoria EEPROM el byte indicado por el apuntador y lo envía al exterior.

4.7.5. LEER BLOQUE DE MEMORIA EEPROM

CODIGO	BYTE 1	BYTE 2
142	nBytesH	nBytesL

RESPUESTA DE LA IG

BYTE 1		BYTE n
B1	….	Bn

La **IG** lee de la memoria EEPROM la cantidad de bytes indicada por nB_H y nB_L (a partir de la localidad señalada por el apuntador) y los envía al exterior.

==========

4.8. INSTRUCCIONES DE MEMORIA EDICION

4.8.1. LEER MEMORIA DE EDICION

CODIGO	BYTE 1
16	nCar

RESPUESTA DE LA IG

BYTE 1		BYTE n
C1	….	Cn

La **IG** envía al exterior desde la memoria de edición el número de caracteres indicado por nCar.
Se envía primero el carácter de la izquierda. nCar puede tomar valores de 1 a 38.

4.8.2. ESCRIBIR EN MEMORIA DE EDICION

CODIGO	BYTE 1	BYTE 2		BYTE n+1
165	nCar	C1	….	Cn

Se escribe en la memoria de edición el número de caracteres indicados en nCar el cual puede tomar valores de 1 a 38, cualquier otro valor puede causar que aparezcan caracteres en posiciones no controladas. Los caracteres que se escribirán son C1 a Cn. Estos caracteres aparecerán la próxima vez que se efectúe una instrucción LINEA DE EDICION.

4.8.3. BORRAR MEMORIA DE EDICION

CODIGO
166

Siempre que se genera una línea de edición, en dicha línea aparecerán los caracteres que están en ése momento en la memoria de edición, si se desea una línea en blanco se debe llamar ésta instrucción antes de ejecutar LINEA DE EDICION.

4.8.4. HABILITAR CURSOR

CODIGO
167

Muestra el cursor en la pantalla.

4.8.5. DESHABILITAR CURSOR

CODIGO
168

Oculta el cursor de la pantalla.

4.8.6. POSICION CURSOR

CODIGO	BYTE 1	BYTE 2
169	XC	YC

Mueve el cursor hacia la posición indicada por XC y YC.

==========

4.9. USO DE INSTRUCCIONES AVANZADAS

INTRODUCCION

Aunque muchas instrucciones son fáciles de comprender y usar, de igual manera existen muchas que requieren una explicación más profunda. A continuación se detallan las instrucciones avanzadas de cada grupo disponible.

4.9.1. INSTRUCCIONES GRAFICAS BASICAS

4.9.1.1. TEXTO CON FORMATO

CODIGO	BYTE 1	BYTE 2		BYTE m
14	Car1	Car2	…	255

Esta instrucción se usa para escribir en la pantalla no sólo caracteres de texto, sino también caracteres de control los cuales se usan para colocar el cursor en cualquier posición además de cambiar de color el texto. Se incluyen caracteres para desplazamiento por número de puntos, retorno de carro, y posicionamiento predefinido. A continuación se muestra una tabla con el byte de control, parámetros y su función.

CONT	**BYTE 1**	**BYTE 2**	**FUNCION**
1	Brillo		Cambia el Brillo para los caracteres siguientes.
2	NEsp		Escribir nEsp en blanco.
3	Np		Saltar nP Puntos hacia adelante.
4			Retorno de carro de una línea (10 puntos).

5			Retorno de carro de línea y media (15 puntos).
6	X	Y	Saltar a las coordenadas X, Y.
7	Fondo		Cambia el color del Fondo para los caracteres siguientes.
8	nB		Saltar a las coordenadas del Botón de Control n
9			Dibujar el color del Fondo en el Texto.
10			No dibujar el color del Fondo en el Texto.

Tabla IV. 2. Bytes de control para la instrucción TEXTO CON FORMATO.

Los valores para Brillo y Fondo pueden ser de 0 a 15.

Los valores para nB pueden ser de 1 a 5, ya que hay un máximo de 5 botones de control.

Para los demás parámetros el valor puede ser de 0 a 254, sin embargo el ancho de la línea en la pantalla es de 228 puntos máximo y puede ser que el desplazamiento del cursor en algunos casos quede fuera de la pantalla.

El código ASCII comienza en el valor 32 en adelante y cualquier número de carácter menor a éste se interpreta como un caracter de control.

Los caracteres de control 9 y 10 indican la manera en que se escribirán los caracteres de texto. Si se selecciona el modo 9 (Dibujar el color del Fondo en el Texto), cada carácter que se escriba se dibujará sobre el color del fondo actual, es decir, primero se dibujará un cuadro lleno con el color del fondo actual y enseguida se dibujará el carácter de texto con el color del Brillo actual.

De otra manera, si se selecciona el modo 10, se dejará el fondo del carácter del color que está en la pantalla.

EJEMPLO DE USO

A continuación se muestra un ejemplo el cual muestra en la pantalla un texto de 2 líneas, la primera en color blanco y la segunda en color negro. El texto se muestra en la posición X=20, Y=80.

0 Instrucción **“Borrar Pantalla”.**

5 Instrucción **“Cursor de Dibujo”.**

20 X.

80	Y.
14	Instrucción **“Texto con Formato”** Desde aquí, y hasta el Byte con el número 255.
1	Brillo.
15	Blanco.
“FILA 1”	Cadena en código ASCII.
1	Brillo.
0	Negro.
4	Retorno de carro de 1 línea.
“FILA 2”	Cadena en código ASCII.
255	Fin de la Cadena. (Fin de la instrucción).

PRECAUCION

La instrucción no debe contener más de 128 bytes en la cadena de texto, esto se debe a que se utiliza la RAM interna del microcontrolador y sólo se ha reservado ésta cantidad de memoria. Si se sobrepasa éste límite se corre el riesgo de sobreescribir en localidades destinadas a otro uso, lo cual podría hacer que se obtuvieran resultados inesperados en la **IG**. Si necesita escribir una cantidad considerable de texto, repita la misma instrucción cuantas veces sea necesario, cada una con un máximo de 128 Bytes en la cadena de texto (Incluyendo el indicador de fin de cadena 255).

==========

4.9.2. INSTRUCCIONES GRAFICAS DE IMÁGENES

4.9.2.1. IMAGEN 2 TONOS

CODIGO	BYTE 1	BYTE 2	BYTE 3		BYTE m+2
32	npH	npV	B1	….	Bm

Esta instrucción dibuja una imagen en la pantalla cuya esquina superior izquierda está en la posición actual del cursor. npH y npV determinan el tamaño de la imagen en pixeles. Es necesario aclarar que npH debe ser un múltiplo de 8, y si ésta condición no se cumple se perderá parte de la información de la imagen, por lo que se mostrará incompleta. Para npV no existe ésta restricción y puede tomar cualquier valor entre 0 y 255. (0 se interpretará como 256).
Se graficará cada bit de cada byte de la instrucción y se interpreta de la siguiente manera: Si el bit es “1” se dibujará un punto del color del Brillo actual. Si el bit es “0” se dibujará un punto del color del Fondo actual. Se dibuja primero el bit más significativo de cada byte.

CODIGO DE LA IMAGEN

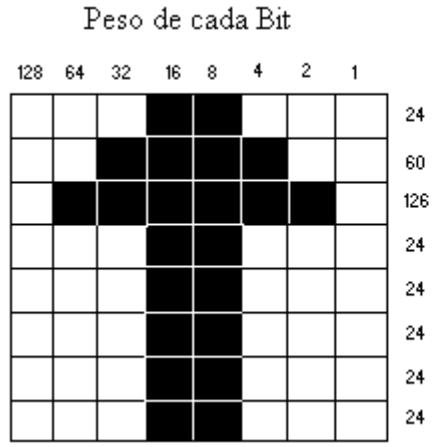
La imagen se debe enviar a la **IG** codificada de cierta manera para que pueda ser interpretada correctamente. Esta se debe dividir en grupos de bytes y se deben enviar en secuencia de izquierda a derecha y de arriba a abajo.

EJEMPLO DE USO

Supongamos que se desea mostrar una imagen la cual consiste en un pequeño cuadro de 8 x 8 pixeles. Para codificar cada byte se debe sumar el peso de cada bit que se deba mostrar. La siguiente figura ilustra éste caso.
La imagen se divide en grupos de bytes y se enviarán a la **IG** con la secuencia indicada en la siguiente tabla.

Byte 1
Byte 2
Byte 3
Byte 4
Byte 5
Byte 6
Byte 7
Byte 8

Tabla IV. 3. Secuencia que se enviara cada byte en una imagen de 2 tonos, 8 x8 pixeles



Para encontrar el valor de cada byte se suma el peso correspondiente a cada bit de acuedro a su color. En el primer byte el peso es de 24, ya que deben dibujarse con color del Brillo los bits de peso 16 y 8.
Esto se hace igual para cada byte y de ésta manera queda codificada la imágen con los valores mostrados a la derecha.

Figura IV. 30. Codificación de una flecha imagen 2 tonos

A continuación se muestra el código de instrucciones para mostrar el inicio de ésta imagen en el centro de la pantalla. (X=114, Y=108).

5	Instrucción **"Cursor de Dibujo"**.
114	X.
108	Y.
32	Instrucción **"Imagen 2 Tonos"**
8	npH Número de pixeles horizontales.
8	npV Número de pixeles verticales.
24 60 126 24 24 24 24 24	Código de la imagen.

Observe que en realidad son muy pocos bytes los que se requieren para una imagen de 2 tonos.

El ejemplo anterior es para una imagen pequeña (8 x 8 pixeles), para una imagen más grande el procedimiento es básicamente el mismo, codificar cada byte y enviarlo a la **IG** con cierta secuencia, la cual se indica en la tabla siguiente para una imagen de 32 x 8 pixeles.

Byte 1	Byte 2	Byte 3	Byte 4
Byte 5			
Byte 29	Byte 30	Byte 31	Byte 32

Tabla IV. 4. Secuencia que se enviara cada byte en una imagen de 2 tonos, 32 x8 pixeles

Como se puede observar, los bytes deben codificarse de izquierda derecha y de arriba abajo.

4.9.2.2. IMAGEN 16 TONOS

CODIGO	BYTE 1	BYTE 2	BYTE 3		BYTE m+2
33	npH	npV	B1	….	Bm

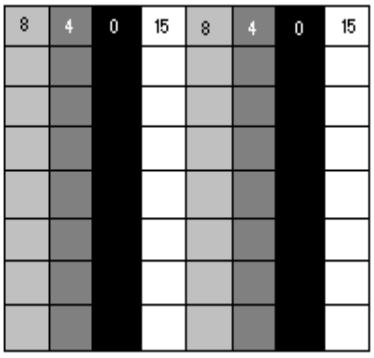
Esta instrucción dibuja una imagen en la pantalla cuya esquina superior izquierda está en la posición actual del cursor. npH y npV determinan el tamaño de la imagen en pixeles. Ambos parámetros pueden tomar cualquier valor entre 0 y 255. (0 se interpretará como 256).

CODIGO DE LA IMAGEN

Cada byte representa un tono de gris entre 0 y 15, los 4 bits más significativos no se usan. Los bytes se envían con la secuencia de izquierda a derecha y de arriba abajo.

EJEMPLO DE USO

Supongamos que se desea mostrar una imagen pequeña de 8 x 8 pixeles. Cada pixel se codifica de acuerdo a su color como se muestra en el dibujo siguiente.



El valor de cada byte corresponde con el valor del color de cada pixel. La codificación de ésta imagen es una secuencia de bytes con los valores 8 4 0 15 8 4 0 15 para la primera fila. Esta secuencia de acuedo a la figura es la misma para cada linea, por lo tanto la codificación se compone de 8 filas con los mismos valores sumando un total de 64 bytes.

Figura IV. 31. Codificación de una flecha imagen 16 tonos

A continuación se muestra el código de instrucciones para mostrar el inicio de ésta imagen en el centro de la pantalla. (X=114, Y=108).

5	Instrucción **“Cursor de Dibujo”**.
114	X.
108	Y.
33	Instrucción “**Imagen 16 Tonos”**
8	npH Número de pixeles horizontales.
8	npV Número de pixeles verticales.
8 4 0 15 8 4 0 15	Inicia el código de la imagen.
8 4 0 15 8 4 0 15	
8 4 0 15 8 4 0 15	
8 4 0 15 8 4 0 15	
8 4 0 15 8 4 0 15	
8 4 0 15 8 4 0 15	
8 4 0 15 8 4 0 15	
8 4 0 15 8 4 0 15	

Observe que se requieren 64 bytes para ésta imagen.

El ejemplo anterior es para una imagen pequeña (8 x 8 pixeles), para una imagen más grande el procedimiento es básicamente el mismo, codificar cada pixel en un byte de color y enviarlo a la **IG** con cierta secuencia, la cual se indica en la tabla siguiente para una imagen de 16 x 4 pixeles.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
17	18														
49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64

Tabla IV.5. Secuencia que se enviara cada byte en una imagen de 16 tonos, 16 x4 pixeles

Como se puede observar, los bytes deben codificarse de izquierda a derecha y de arriba abajo.

Observe que los cuadros 1, 2 y 3 tienen el mismo color en cada pixel, y sólo el número 4 contiene información variable. Por lo tanto los bloques 1, 2 y 3 son candidatos a comprimir.

Dado que la imagen se compone de 4 bloques, se necesitan también 4 bytes de control.

A continuación se muestra la codificación de cada bloque

.

BLOQUE 1

Byte de control 1 0 0 0 0 0 0 1 Compresión, 64 bits del color del Brillo.

BLOQUE 2

Byte de control 1 0 0 0 0 0 0 0 Compresión, 64 bits del color del Fondo.

BLOQUE 3

Byte de control 1 0 0 0 0 0 0 0 Compresión, 64 bits del color del Fondo.

BLOQUE 4

Byte de control 0 0 0 0 0 0 0 0 El bit 7 indica no compresión.Viene bloque completo.

0 66 60 36 36 60 66 0 Bloque completo de 8 bytes.

La instrucción se compone solo de 12 bytes, 4 bytes de control y un solo bloque de 8 bytes completo. Los bytes de control se muestran en binario para una mejor visualización.

A continuación se muestra el código de instrucciones para mostrar ésta imagen en el centro de la pantalla. (X=114, Y=108).

5	Instrucción **“Cursor de Dibujo”**.
114	X.
108	Y.
34	Instrucción “**Imagen 2 Tonos Comprimida”**
16	npH Número de pixeles horizontales.
16	npV Número de pixeles verticales.
129	Byte de control 1. Dibuja un bloque del color del Brillo.
128	Byte de control 2. Dibuja un bloque del color del Fondo.
128	Byte de control 3. Dibuja un bloque del color del Fondo.
0	Byte de control 4. No hay compresión. Viene codigo.
0 66 60 36 36 60 66 0	Código del bloque 4.

Para obtener el código del bloque 4 se procede de la misma forma que en la instrucción IMAGEN 2 TONOS, pesando cada bit de acuerdo a su color y sumándolos.

Si no se comprimiera la imagen, ésta sería de 32 bytes, en lugar de 12, sin embargo efectuando la compresión el código de la imagen se reduce a un 37% del original.

En el caso de imágenes más grandes el procedimiento es el mismo, se divide la imagen en bloques de 8 x 8 pixeles y se codifican de izquierda derecha y de arriba

abajo enviando un byte de control por cada bloque.

==========

4.9.3. INSTRUCCIONES GRAFICAS DE CONTROL

4.9.3.1. BOTONES DE CONTROL

CODIGO	BYTE 1
64	N

Por lo general, los aparatos que tienen una pantalla para despliegue de datos muestran en algún lado botones virtuales (dibujados) que se relacionan con botones reales colocados cerca del dibujo en la pantalla. De ésta manera se sugiere de acuerdo al texto mostrado en la pantalla el botón que debe presionarse para ejecutar alguna función. Por ejemplo, el botón dibujado en la pantalla podría mostrar el texto “CH 1”, y cerca de éste dibujo se encuentra el botón real que al presionarlo se ejecuta dicha función.

La **IG** hace uso de ésta instrucción para dibujar de 1 a 5 botones de control virtuales en la parte inferior de la pantalla, los cuales pueden ser relacionados con botones reales cerca de éstos.

En el byte 1 de la instrucción se especifica cuantos botones se dibujarán (N). El texto se escribe en los botones haciendo uso de la instrucción “Texto con Formato”.

EJEMPLO:

Se dibujarán 2 botones de control, uno con el texto “CH 1” y otro con el texto “CH 2”.

64	Instrucción “**Botones de Control**”
2	Byte 1, número de botones
14	Instrucción **“Texto con Formato**”

8	Saltar a botón de control
1	Número 1
“CH 1”	Texto del botón 1
8	Saltar al botón de control
2	Número 2
“CH 2”	Texto del botón 2
255	Indicador de fin de instrucción “Texto con Formato”

Recuerde que las letras se deben enviar en código ASCII, por lo tanto debe sustituir cada letra por su código equivalente.

4.9.3.2. CUADRO DE DIALOGO

CODIGO	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5
66	X	Y	X2	Y2	Tipo Bot

BYTE 6		BYTE n+5	BYTE k		BYTE m
CarTit1	….	CarTit n	….	CarCont1	…. 255

Esta instrucción dibuja un cuadro de diálogo con botones de varios tipos, los cuales incluyen texto. La **IG** los coloca automáticamente en el cuadro y son fáciles de manipular con instrucciones simples las cuales simulan que se presionan. Después de enviar las coordenadas donde se dibujará el cuadro, se envía el tipo de botones que aparecerán en éste, (Aceptar, Cancelar etc.) El cuadro tiene un título que aparecerá en la parte superior con fondo oscuro. Después del título se puede escribir texto con formato, el cual se colocará automáticamente.
A continuación se detalla el significado de cada parámetro de la instrucción.

X, Y	Coordenadas de la esquina superior izquierda del cuadro de diálogo.
X2, Y2	Coordenadas de la esquina inferior derecha del cuadro de diálogo.

Tipo Bot	Indica el tipo de botones que se dibujará en el cuadro: 1 ACEPTAR 2 CANCELAR-ACEPTAR 3 CANCELAR-SI-NO
CarTit 1-CarTit n-5	Título que aparecerá en la barra superior.
CarCont1…	Texto con Formato.
255	Indicador de fin del Texto con Formato.

EJEMPLO:

A continuación se muestra el código que se enviará a la **IG** para mostrar un cuadro de diálogo típico de manipulación de archivos.

66	Instrucción **“Cuadro de Diálogo”**
10	X Coordenadas de la esquina superior izquierda.
10	Y
134	X2 Coordenadas del a esquina inferior derecha.
110	Y2
3	Tipo de botones = CANCELAR-SI-NO.
“ARCHIVO”	Título.
5	Retorno de carro de línea y media.
7	Cambio del color del fondo.
8	Gris medio.
“N”	Letra N.
1	Cambio de color.
3	Gris oscuro.
“UEVO”	Letras UEVO.
5	Retorno de carro de línea y media.
1	Cambio de color.
12	Blanco pálido.
“A”	Letra A.
1	Cambio de color.
3	Gris oscuro.
“BRIR”	Letras BRIR.

5	Retorno de carro de línea y media.
1	Cambio de color.
12	Blanco pálido.
“G”	Letra G.
1	Cambio de color.
3	Gris oscuro.
“UARDAR”	Letras UARDAR .
1	Cambio de color.
12	Blanco pálido.
“S”	Letra G.
1	Cambio de color.
3	Gris oscuro.
“ALIR”	Letras ALIR.
255	Indicador de fin de Texto con Formato.

Después de ejecutarse el código, se muestra un cuadro de diálogo con 3 botones, los cuales contienen los textos: “CANCELAR”, “SI” y “NO”. El título es “ARCHIVO” y luego se muestran 4 líneas de texto con la primera letra resaltada en blanco pálido, los textos son: “NUEVO”, “ABRIR”, “GUARDAR” y “SALIR”.

==========

4.9.4. INSTRUCCIONES GRAFICAS DE DESPLIEGUE DE DATOS

4.9.4.1. MOSTRAR DATO EN DISPLAY DE CUARZO EN DECIMAL

CODIGO	BYTE 1	BYTE 2	BYTE 3	BYTE 4
96	BF	B2	B1	B0

Con ésta instrucción se pueden desplegar en la pantalla dígitos semejantes a un display de cuarzo. El número que se desplegará viene indicado por los bytes B0

(menos sig.) B1 y B2 (más sig.). BF indica el signo, el número de dígitos visibles y la posición del punto. La siguiente tabla indica el formato de BF.

BIT	SIGNIFICADO	VALOR
b7	Signo	0 Sin signo, 1 signo negativo
b6 b5 b4	Número de Dígitos Visibles	0 a 7 para 1 a 8 dígitos
b3 b2 b1 b0	Posición del punto	0 a 8

Tabla IV. 6. Formato bit BF, MOSTRAR DATO EN DISPLAY DE CUARZO EN DECIMAL

La posición 0 del punto es a la derecha y la posición 8 es a la izquierda. Sin embargo, un valor de 0 no mostrará ningún punto, ya que el punto colocado a la derecha del número lo convierte en un entero.

La cuenta más alta que se puede mostrar es 16777215, la cual corresponde a 3 bytes.

El número de dígitos visibles es útil si se van a mostrar menos de 8 dígitos, por ejemplo, si solo se requiere mostrar números de 1 byte (valor máximo 255), se puede programar en b5, b4 y b3 el número binario 010 (decimal 2), con lo cual se mostrará un máximo de 3 dígitos.

La **IG** automáticamente eliminará los ceros a la izquierda del punto.

EJEMPLO:

El siguiente código muestra la manera en que se usa la instrucción. En éste caso se muestra un display de cuarzo con el número 25.5 en las coordenadas 50, 50.

5	Instrucción **“Cursor de Dibujo”**
50	Coordenadas X, Y
50	
96	Instrucción “**Display de Cuarzo**”
0 0 1 0 **0 0 0 1**	BF =No signo, 3 Dígitos y punto en posición 1
0	B2
0	B1
FFH	B0

En el ejemplo se muestra BF en binario para una mejor visualización de los bits de control.
También se muestra B0 en hexadecimal para acentuar el hecho de que la **IG** codifica el código binario que recibe en código decimal para mostrarlo en la pantalla.

PRECAUCION

Se debe dejar espacio a la izquierda para el signo negativo aunque no se use. De no hacerlo así se mostrarán caracteres aleatorios en la pantalla. Por lo que la coordenada X del display deberá ser mayor a 25, para cumplir con ésta restricción.

4.9.4.2. GRAFICAR PASO X

CODIGO BYTE 1
102 nB

Esta instrucción grafica en la pantalla los datos almacenados en la Memoria Interna, comenzando con la localidad 0. Cada byte representa un valor del eje Y. Al momento de graficar, los puntos se unen automáticamente con una línea, de éste modo se muestra una gráfica continua en la pantalla.
El BYTE 1 contiene el número de puntos que se graficarán (nB). Este puede tomar cualquier valor entre 2 y 255. Sin embargo, recuerde que la pantalla tiene un ancho máximo de 228 puntos.
En ésta instrucción el punto Y=0 se encuentra en la esquina inferior de la pantalla y éste valor crece hacia arriba hasta Y=215. Observe que ésta convención es contraria a las demás instrucciones que hacen uso de coordenadas. Esto se hizo así debido a que el uso principal de ésta instrucción es para graficar datos en una aplicación de osciloscopio digital, donde un valor 0 se representa en la parte baja de la pantalla y crece hacia la parte superior. El eje X no sufre alteraciones, por lo tanto en todas las instrucciones crece hacia la derecha.

EJEMPLO:

El siguiente código muestra un ejemplo de la instrucción con la cual se dibuja en la pantalla dos ciclos de una onda diente de sierra. Primero se debe guardar en la Memoria Interna los datos que se pretende graficar, los cuales en éste ejemplo son 16, esto se consigue con la instrucción “Escribir Bloque en Memoria Interna”. Luego de almacenar los datos en la memoria de la **IG**, enviamos la instrucción “Graficar Paso X”, la cual dibujará la forma de onda en la pantalla.

130	Instrucción **“Escribir Bloque en Memoria Interna”**
16	Número de bytes que se guardarán
10, 11, 12, 13 ,14, 15, 16, 17	Datos del primer ciclo
10, 11, 12, 13, 14, 15, 16, 17	Datos del segundo ciclo
5	Instrucción **“Cursor de Dibujo”**
100	Coordenada X.
0	Coordenada Y. En éste caso el valor no importa.
102	Instrucción **“Graficar PasoX”**
16	Número de bytes que se graficarán

En el siguiente dibujo donde se han ampliado los pixeles (cuadrados) se muestra la forma en que se vería la gráfica en la pantalla.

Recuerde que la coordenada Y=0 se encuentra en la parte inferior de la pantalla y el primer punto tiene el valor de 10 por lo tanto la gráfica se muestra colocada a 10 pixeles de la parte de abajo y al centro.

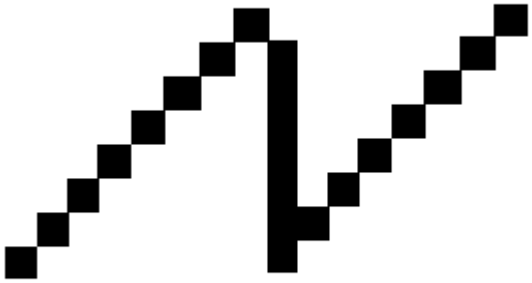


Figura IV. 32. Grafica paso x , diente de sierra

Observará que en la coordenada Y el valor no importa ¿Por qué? El primer valor que tomará Y es el del primer dato que se dibujará, por lo tanto cualquier valor que se indique no se tomará en cuenta, sin embargo la instrucción “Cursor de Dibujo” espera algún valor.

.

PRECAUCION

El valor de cualquier punto debe estar entre 0 y 215 para que pueda ser dibujado en la pantalla, si ésta condición no se cumple, los puntos cuyo valor sea mayor a 215 darán “vuelta” en la pantalla y se tratarán de dibujar en la parte inferior, generándose líneas fuera de control.

4.9.4.3. GRAFICAR LINEA BYTE

CODIGO	BYTE 1	BYTE 2	BYTE 3
103	nBits	TamH	TamV

Con ésta instrucción se puede representar gráficamente en la pantalla una secuencia de datos binarios los cuales se extraen de la Memoria Interna. Se comenzará a graficar en la posición actual del cursor X, Y. El número de bits a graficar viene indicado por

nBits en el BYTE 1. TamH indica el tamaño de las líneas horizontales y TamV el de las verticales, tal como se indica en el siguiente dibujo:

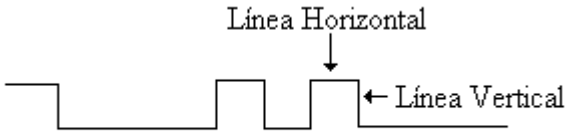


Figura IV. 33. Tamaño de líneas horizontales y verticales

El tamaño de las líneas puede ser desde 1 hasta 255 pixeles (0 se interpretará como 256), sin embargo si se especifica un tamaño muy grande, de modo que la secuencia no quepa en la pantalla, es probable que se obtengan resultados inesperados al dibujarse el trazo.
Los bytes se grafican secuencialmente siendo el bit más significativo el de la izquierda.
Antes de ejecutar ésta instrucción se debe tener almacenados los datos que se pretende graficar en la Memoria Interna, lo cual se consigue con la instrucción “Escribir Bloque en Memoria Interna” o, si son pocos datos se puede utilizar la instrucción “Escribir Byte en memoria Interna”.

EJEMPLO:

A continuación se muestra el código de un ejemplo el cual graficará 24 bits con un tamaño vertical y horizontal de 8 pixeles.

130	Instrucción **“Escribir Bloque en Memoria Interna”**
3	Número de bytes que se guardarán
$AA, 0, $FF	Los 3 bytes (24 bits) que se graficarán
5	Instrucción **“Cursor de Dibujo”**
5	X.
30	Y.
103	Instrucción **“Graficar Línea Byte”**
24	nBits Número de bits que se graficarán

8	Tam H	Tamaño de la línea horizontal
8	Tam V	Tamaño de la línea vertical

Después de que la **IG** ejecuta la instrucción se observa en la pantalla un trazo semejante al que indica el siguiente dibujo:

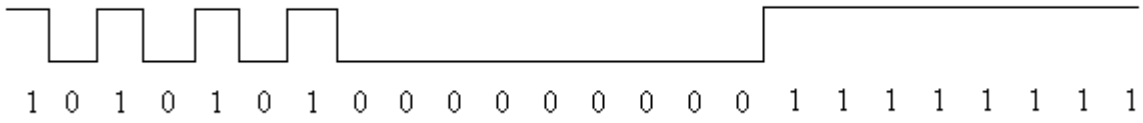


Figura IV.34. Trazo en pantalla de línea de byte

El número hexadecimal $AA es equivalente al 1010 1010 en binario, como se puede apreciar en el primer byte graficado.
Cabe aclarar que no es necesario especificar un número múltiple de 8 bits, es posible graficar por ejemplo 20 bits, 3 bits, 15 bits etc.

4.9.4.4.GRAFICAR LINEA BIT

CODIGO	BYTE 1	BYTE 2	BYTE 3	BYTE 4	BYTE 5
104	nBytes	TamH	TamV	Sep	bits

Esta instrucción es semejante a la anterior, la diferencia radica en que ésta dibuja las gráficas binarias en forma vertical, es decir, para dibujar un byte se comienza dibujando el bit menos significativo (b0), luego hacia abajo se dibujará cada bit del byte (b1, b2… b7).
Aquí nBytes indica el número de bytes que se graficará en forma vetical y el número de trazos (bits) lo especifica bits.
Los parámetros TamH y TamV tienen la misma función que la instrucción anterior y Sep indica la separación entre trazos medida en pixeles.
El primer trazo comenzará en la posición actual del cursor X, Y. Igual que en anteriores instrucciones, los datos que se graficarán deben estar en la Memoria Interna.

EJEMPLO:

A continuación se muestra el código de un ejemplo el cual graficará los 8 bits de 4 bytes con un tamaño vertical y horizontal de 5 pixeles y una separación de 10 pixeles.

130	Instrucción **“Escribir Bloque en Memoria Interna”**	
4	Número de bytes que se guardarán	
$AA, 0, 1,255	Los 4 bytes (24 bits) que se graficarán	
5	Instrucción “Cursor de Dibujo”	
20	X.	
30	Y.	
104	Instrucción **“Graficar Línea Bit”**	
4	nb	Número de bits por trazo que se graficarán
5	Tam H	Tamaño de la línea horizontal
5	Tam V	Tamaño de la línea vertical
10	Sep	Separación entre trazos
8	bits	Número de bits de cada byte que se graficarán

Después de que la **IG** ejecuta la instrucción se observa en la pantalla un trazo semejante al que se muestra en el siguiente dibujo donde también se indica el significado de los parámetros que se enviaron en la instrucción:

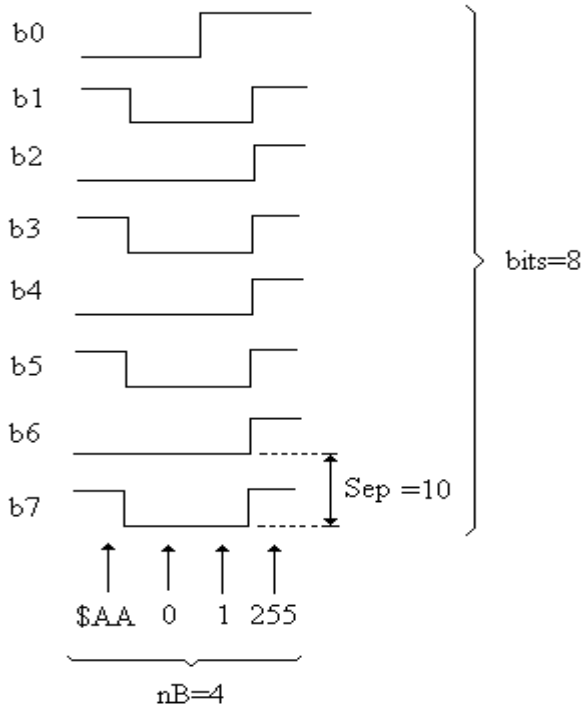


Figura IV.35. Trazo en pantalla grafica línea bit con sus parámetros

Con ésta instrucción se puede implementar fácilmente un analizador de estados lógicos. Esto se consigue colocando en la memoria interna de la **IG** los datos binarios que se desea graficar y posteriormente ejecutando la instrucción. La **IG** se encargará de mostrar en la pantalla la secuencia de bits en forma gráfica.

==========

4.10. COMENTARIOS ACERCA DE LOS TIPOS DE MEMORIA

4.10.1. CONSIDERACIONES PARA USAR LOS 3 TIPOS DE MEMORIA DISPONIBLES

En el desarrollo de algún proyecto puede ser necesario usar memoria, ya sea del tipo volátil (RAM) o no volátil (EEPROM o Flash). La IG cuenta con ambos tipos de memoria disponibles para el usuario. A continuación se exponen los 3 tipos disponibles y sus características.

Nombre	**Tipo**	**Capacidad**	**Vel Lectura**	**Vel Escritura**
Memoria Interna	RAM	256 Bytes	Alta	Alta
Memoria de Página	RAM	32768 Bytes	Media	Media
EEPROM Externo	EEPROM	32768x2	Alta	Muy Baja

La memoria interna está implementada en la memoria RAM del microcontrolador gráfico, por tal motivo es la más rápida en el acceso de escritura y lectura.
El uso normal es para recibir datos del exterior para desplegarlos mediante alguna instrucción diseñada para éste fin.

La memoria de página está implementada en la RAM de video, el acceso a ésta es relativamente lento ya que el microcontrolador sólo puede accesarla durante breves períodos, cuando la interface gráfica no la está ocupando. Aunque es más lenta se cuenta con mayor capacidad.
El uso normal es para almacenar grandes cantidades de datos, como podrían ser los datos de un analizador de estados lógicos.

Le memoria EEPROM es opcional y se cuenta con 2 bases para memorias de 32768 bytes cada una. La lectura es bastante rápida, sin embargo la escritura requiere de varios milisegundos.
El uso normal es para guardar datos que interesa conservar, como por ejemplo los caracteres de un editor de texto. En el conector de 16 patas de interface a la IG están disponibles las patas de acceso a la memoria EEPROM para que ésta pueda ser utilizada en algún proyecto sin la intervención de la **IG**

CAPITULO V

PRUEBAS Y RESULTADOS DE LA IG

En este capítulo se presenta las pruebas que se diseñaron para comprobar el funcionamiento del sistema desarrollado, para lo cual se ha realizado un reloj digital como proyecto.

5.1. EQUIPO NECESARIO

Para el proyecto ejemplo necesitará de una PC con Windows XP o superior y un programador de microcontroladores PIC.

Es necesario aclarar que se puede utilizar cualquier tipo de microcontrolador y no necesariamente PIC (la IG tiene incluso un microcontrolador Atmel), sin embargo, los PIC son populares y existen muchos programadores de bajo costo y fáciles de utilizar por lo que hemos decidido emplear éste microcontrolador en el proyecto ejemplo.

5.2. SUBRUTINAS Y MACROS ESPECIALES

Subrutinas en carpeta “Comunes”:

ComIG.asm	Comunicación con la IG.
MacrosIG.asm	Macros que facilitan enviar instrucciones a la IG.
TablasDatos.asm	Subrutinas que facilitan enviar datos y textos a la IG.

5.3. PLANEANDO PROYECTO RELOJ DIGITAL

Diseñaremos un reloj digital el cual mostrará en la pantalla un display de cuarzo simulado con las horas, minutos y segundos. Tendrá dos botones de control para ajustar las horas y los minutos: “HOR” y “MIN”.
Al presionar el botón “MIN” se incrementarán los minutos y se pondrán en cero los segundos. No tiene caso poner un botón para ajustar los segundos.
Dibujaremos un pequeño icono en forma de reloj en la esquina superior izquierda y trazaremos dos barras superiores con texto. El texto de la primera será “HENRY VALLEJO ESPOCH” y el de la segunda será “RELOJ DIGITAL”.

5.4. PRINCIPIOS DE FUNCIONAMIENTO DE UN RELOJ

En el momento en que empezamos el diseño de un reloj lo primero que se nos viene a la mente es: ¿de dónde vamos a sacar la referencia del tiempo?
Para esto existen básicamente 3 opciones:

a) Un oscilador con RC.
b) Un oscilador con cristal.
c) Una señal de “línea”.

La primera opción francamente ni debemos considerarla, ya que un oscilador simple RC aunque lo implementemos con resistencia y condensador de precisión no proporcionaría la estabilidad de la frecuencia necesaria para un reloj.

La segunda opción parece más viable, podríamos incluso utilizar el cristal del mismo microcontrolador y utilizar el timer interno para generar un pulso cada segundo como referencia, sin embargo el software se complicaría un poco.

Por lo tanto solo nos queda la tercera opción y es la que usaremos, esto es, tomaremos la frecuencia de la línea de AC la cual es de 60 Hz de razonable precisión y la usaremos como nuestra referencia de tiempo.

¿Cómo lo haremos?

Emplearemos el siguiente algoritmo simple:

Introduciremos los pulsos de 60 Hz al timer interno del PIC y generaremos una interrupción cada 60 pulsos, esto es, cada segundo. En la subrutina de interrupción incrementaremos los segundos, si los segundos son 60, pondremos en cero los segundos e incrementaremos los minutos, si los minutos son 60, pondremos en cero los minutos e incrementaremos las horas. Si las horas son 24, pondremos las horas en cero.

Lo anterior es para actualizar la hora cada segundo. Para el funcionamiento de los interruptores simplemente verificaremos en el programa principal si han sido presionados, en caso de ser así se incrementarán los minutos o las horas, dependiendo del interruptor presionado.

5.5. Circuito Reloj Digital

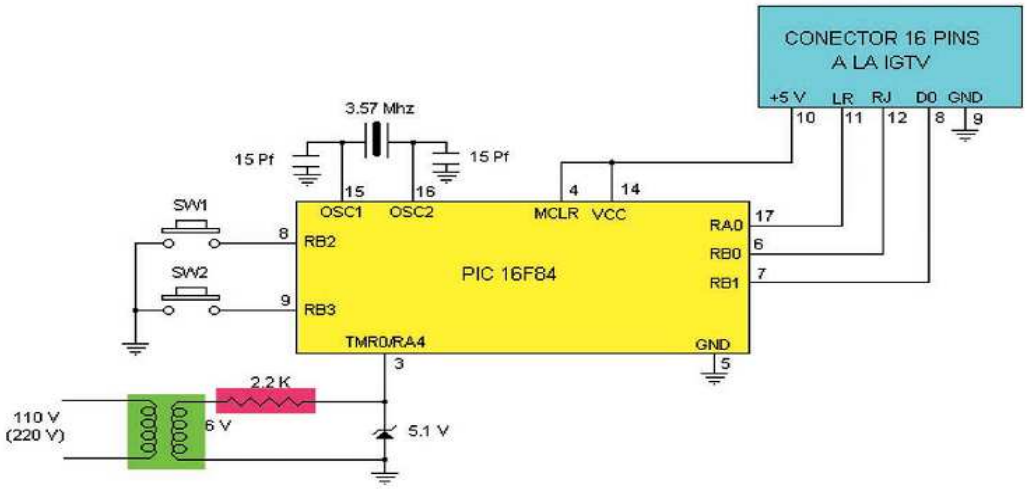


Figura V.36. Reloj Digital

5.6. Instrucciones iniciales

```
;*********************************************************************
;                       PROYECTO RELOJ DIGITAL
;*********************************************************************
#INCLUDE p16f84a.inc
                        __CONFIG _CP_OFF & _WDT_OFF & _PWRTE_ON & _XT_OSC
;---------------------------------------------------------------------------------------------------------
;                       DEFINIR VARIABLES
;---------------------------------------------------------------------------------------------------------
                        CBLOCK 0x0C
                        Dato
                        Reg0
                        Reg1
                        Reg2
                        Segundos
                        Minutos
                        Horas
                        FormPunto
                        Cero
                        ENDC
;---------------------------------------------------------------------------------------------------------
;                       DEFINIR PUERTOS
;---------------------------------------------------------------------------------------------------------
#DEFINE MODO_D0 TRISB,1         ; (IG).
#DEFINE LR PORTA,0                      ; Listo para Recibir (IG).
#DEFINE RJ PORTB,0                      ; Reloj (IG)..
#DEFINE D0 PORTB,1                      ; D0 (IG)..
;-------------------------------------------------
#DEFINE SW1 PORTB,2                     ; Switch 1.
#DEFINE SW2 PORTB,3                     ; Switch 2.
```

La cuarta línea le indica al ensamblador que incluya el archivo “p16f84.inc” el cual usa para encontrar definiciones de registros, puertos y otros datos relacionados con el PIC16F84.

La siguiente línea se usa para configurar el microcontrolador respecto a algunas opciones como protección de código, tipo de cristal que se usará etc.

Enseguida deberemos apartar localidades de memoria para ciertas variables que necesitaremos en nuestro proyecto, para esto utilizaremos la directiva de ensamblador CBLOCK, la cual define una lista de nombres de localidades comenzando en el número de localidad indicada enseguida de CBLOCK.

En el caso del PIC16F84, las localidades RAM que podemos usar comienzan en el número 0x0C (12 en decimal).

La línea #DEFINE MODO_D0 es especial, la usa la subrutina RECIBIR_BYTE (la cual en este proyecto no se usa) para cambiar D0 a entrada o salida y define en qué bit de qué puerto está D0. Observe que en éste caso D0 está en el Puerto B en el Bit 1 (TRISB,1).

Las últimas líneas definen el nombre que se usará en 5 patas de los puertos del PIC. Estas son las líneas de comunicación con la IG y 2 interruptores. Observe en el diagrama la coincidencia de LR, RJ y D0 de la base de 16 patas de la IG con las patas de los puertos del PIC.

5.7. MACROS DISPONIBLES

En el archivo Macros.asm he creado varias macros que simplifican la escritura y mejoran la legibilidad del código. A continuación se indican algunas de éstas macros y su función:

BORRAR_PANTALLA

La pantalla se borra llenándose del color actual del fondo.

CURSOR X,Y

El cursor se mueve a la posición indicada por X,Y. Las siguientes instrucciones gráficas y de texto se dibujarán en la nueva posición.

DISPLAY_CUARZO Formato, Byte2, Byte1, Byte0

Dibuja un display de cuarzo y despliega el número indicado por los 3 bytes.
Formato indica si el número llevará signo, el número de dígitos visibles y
la posición del punto.
Consulte el manual de la IG para una descripción más completa.

BRILLO Color

Cambia el color actual de dibujo al indicado por Color, el cual puede estar entre

0 y 15. (negro y blanco).

BOTONES_CONTROL NumBotones

Dibuja botones de control en la parte inferior de la pantalla. NumBotones indica cuantos botones se dibujarán. Pueden ser hasta 5.

CUADRO_LLENO X2,Y2

Dibuja un cuadro del color del Brillo actual. X2,Y2 indican la esquina inferior derecha y la esquina superior izquierda viene dada por el cursor actual.

MARCO_ALTO_REL X2, Y2

Dibuja un marco resaltado (alto relieve).Las coordenadas se interpretan de la misma manera que en la macro anterior.

TEXTO_FORMATO índice

Envía texto con formato a la pantalla.
Esta macro trabaja combinada con una tabla de texto, la cual debe definirse antes de llamar la macro. Indice es el offset del byte inicial que se enviará hasta que se encuentre el fin de la cadena, el cual es el número 255.

Para una explicación más amplia del texto con formato consulte su manual.

IMAGEN_2TONOS TamHor, TamVert, indice

Envía una imagen a la pantalla.
Al igual que la macro de texto, es necesario que se defina una tabla de datos para poder usar ésta macro.
TamHor y TamVert indica el número de pixeles horizontales y verticales de la imagen. Esta se dibujará en el cursor actual. Indice es el offset en la tabla de datos del byte inicial de los datos de la imagen.

5.8. TABLAS DE DATOS

La manera más fácil de enviar un grupo de bytes a la IG consiste en agruparlos en una tabla y mediante una macro enviarlos uno por uno a la tarjeta.
La directiva DT (Define Table) define una tabla de valores que permite inclusive escribir caracteres en código ASCII, lo cual es apropiado para enviar textos.
Existe una macro para enviar texto y otra para enviar datos (los cuales pueden ser una imagen). Para usar dicha macro se debe proporcionar el offset en la tabla correspondiente al primer byte que se enviará.

5.9. VECTORES DE INICIO E INTERRUPCION, INCLUSION DE ARCHIVOS

```
;-------------------------------------------------------------------------------------------
                        ORG 0x0000
RESET                           GOTO INICIO
                        ORG 0x0004
                        GOTO INT_TMR0
                        ORG 0x0005
# INCLUDE “..\Comunes\ComIG.asm”
# INCLUDE “..\Comunes\MacrosIG.asm”
# INCLUDE “..\Comunes\TablasDatos.com”
```

La directiva ORG 0x0000 indica la dirección donde comenzará la ejecución del programa después de un reset. Aquí se ha escrito una instrucción de salto a la etiqueta INICIO.
Debido a que planeamos usar la interrupción del timer, el vector de inicio de las interrupciones está en la localidad 0x0004, donde se ha escrito un salto a la etiqueta INT_TMR0.

En cuanto se produzca una interrupción del timer, se ejecutarán las instrucciones a partir de la etiqueta INT_TMR0.
A continuación escribimos ORG 0x0005 que es donde iniciaremos nuestro código de subrutinas

Obsérvese las directivas INCLUDE las cuales le indican al ensamblador incluir 3 subrutinas junto con las demás instrucciones que integran el programa.

5.10. CODIGO DE LAS TABLAS DE DATOS Y TEXTOS

```
;-------------------------------------------------------------------------
TABLA_DATOS     ADDWF PCL,F
                        DT 3,192,12,48,16,136,34,164,32,132,72,146,64,130,64,130
                        DT 80,250,64,2,72,18,32,4,34,36,16,136,12,48,3,192
;-----------------------------------------------------------
TABLA_TEXTOS    ADDWF PCL,F
                        DT "HENRY VALLEJO       ESPOCH",255                     ; 0
                        DT "RELOJ DIGITAL",8,1," HOR",8,2," MIN",9,255; 28
```

La tabla de datos agrupa los valores de una imagen de 2 tonos de 16x16 pixeles la cual consiste en un pequeño reloj analógico. Consulte en su manual la instrucción IMAGEN 2 TONOS si requiere de una explicación adicional.

La tabla de textos agrupa dos cadenas: la primera simplemente tiene texto, sin embargo la segunda contiene caracteres de control. La tabla 1 muestra los distintos códigos de control para texto con formato. (Ver tabla IV.2)

Cuando la IG procesa los datos de la TABLA_TEXTOS de la segunda línea escribe “RELOJ DIGITAL” en la posición del cursor actual, luego hay un carácter de control “8”, lo cual significa “saltar a las coordenadas del botón de control”, luego viene un “1”, esto es, el botón 1, Luego la IG escribe “ HOR”, lo cual lo hace dentro del botón de control, ya que anteriormente le ordenamos saltar a éste control.

Algo semejante ocurre con el botón de control 2, dentro del cual se escribe el texto: “ MIN”. Luego viene el código “9”, lo cual hace que de ahora en adelante al escribir texto se redibuje el color del fondo. Si no hacemos esto, los caracteres del texto que dibujemos se muestran empalmados, ya que la IG no borraría el fondo. (esto concierne a los segundos del display del tiempo).

5.11. SUBRUTINA DE INTERRUPCION DEL TMR0

Subrutina que se llamará cuando el TMR0 genere una interrupción, esto es, cada 60 pulsos.

```
;--------------------------------------------------------------------------
INT_TMR0                BCF INTCON,2      ; Borrar interrupción del TMR0.
                        MOVLW 196                 ;  (206 para una frecuencia de CA de 50 Hz).
                        MOVWF TMR0
                        INCF Segundos,1
                        MOVLW 60
                        SUBWF Segundos,0
                        BTFSS STATUS,Z
                        GOTO DesplegarSeg
                        CLRF Segundos
                        INCF Minutos,1
                        MOVLW 60
                        SUBWF Minutos,0
                        BTFSS STATUS,Z
                        GOTO DesplegarMin
                        CLRF Minutos
                        INCF Horas,1
                        MOVLW 24
                        SUBWF Horas,0
                        BTFSS STATUS,Z
                        GOTO DesplegarHoras
                        CLRF Horas
DesplegarHoras      CURSOR 60,95
                        DISPLAY_CUARZO FormPunto,Cero,Cero,Horas
DesplegarMin            CURSOR 120,95
                        DISPLAY_CUARZO FormPunto,Cero,Cero,Minutos
DesplegarSeg            CURSOR 165,117
                        DISPLAY_DECIMAL FormPunto,Cero,Cero,Segundos
                        RETFIE
```

Lo primero que debemos hacer al entrar a la subrutina es borrar el bit de aviso de interrupción del TMR0 (INTCON, 2), enseguida debemos cargar el TMR0 con el valor 196 .El timer interrumpe al PIC cada vez que hay un sobreflujo, la máxima cuenta es 255, en el siguiente pulso se cambia a 0, lo cual genera la interrupción. Si así lo dejamos, la siguiente interrupción sería dentro de 256 pulsos, lo cual no es lo que deseamos, ya que necesitamos que cuente 60 pulsos y no 256. Si cargamos al TMR0 con 196, entonces con 60 pulsos más se produce el sobreflujo (196+60=256) y se genera la interrupción, que es justo lo que deseamos.

Enseguida está el código del pequeño algoritmo que se expuso anteriormente.

Observe el uso de la macro DISPLAY_CUARZO.

Esta macro espera 3 bytes en binario, lo cual puede mostrar un número mayor a 16 millones, sin embargo solo usamos un número menor que 60 para cada display (Horas, minutos y Segundos) por lo tanto debemos enviar ceros en los bytes de valor superior. Para esto hemos definido un registro de nombre “Cero” en el cual colocamos el valor de 0 y lo enviamos en los dos bytes que no se usan.

El registro FormPunto se usa para definir el formato de los números del display.

Observe que si no se produce cambio en los minutos o las horas no se refresca el display respectivo.

PRECAUCION

DISPLAY_CUARZO espera como argumentos registros, no valores numéricos. Por ejemplo, si tecleamos DISPLAY CUARZO 160, 4, 2, 100, su ejecución puede causar resultados inesperados, ya que la macro enviaría el contenido de la localidad 160, luego el de la localidad 100, luego el de la localidad 2 y por último el de la localidad 4 ¡no los valores numéricos!

Primero debe definir 4 registros, luego colocar en ellos el formato del punto y los 3 bytes que desea desplegar. Para usar la macro coloque los nombres de éstos registros en los argumentos.

5.12. CODIGO PRINCIPAL

Básicamente lo que hacemos en ésta parte del programa es iniciar algunos registros, programar el registro de opciones y configurar los puertos y las interrupciones.

```
;-------------------------------------------------------------------------------------------------
;                               INICIO DE CODIGO PRINCIPAL
;-------------------------------------------------------------------------------------------------
INICIO                  BSF STATUS,RP0          ; Seleccionar página 1.
                        MOVLW 0xFF
                        MOVWF TRISA             ; Puerto A todas entradas.
                        MOVLW 0xFE
                        MOVWF TRISB             ; PORTB,0=RJ Salida.
                        MOVLW 0x28
                        MOVWF OPTION_REG        ; Programar opciones.
                        BCF STATUS,RP0          ; Regresar a página 0.
                        BCF RJ                          ; RJ=0.
                        BCF D0                          ; D0=0.
                        CLRF Segundos           ; Segundos=0.
                        CLRF Minutos                    ; Minutos=0.
                        CLRF Horas                      ; Horas=0.
                        CLRF Cero
                        MOVLW 18
                        MOVWF FormPunto
                        MOVLW 15                        ; Instrucción Punto Invisible.
                        CALL ENVIAR_BYTE
                        MOVLW 1
                        CALL ENVIAR_BYTE
```

5.12.1. CONFIGURACION DE PUERTOS

El puerto A es configurado en modo entrada.

El Puerto B se configura con B,0 como salida, ya que es RJ para la comunicación con la IG. Al inicio los demás bits son entradas (incluyendo D0).

5.12.2. REGISTRO DE OPCIONES (OPTION_REG)

A continuación el programa carga el registro de opciones con el valor 0x28.

BIT	USO	Valor fijado	Motivo
0-2	Pre-escalador	0 0 0	No importa.
3	Asignación del pre-escalador	1	Al watch-dog.
4	Modo Disparo TMR0	0	A la bajada.
5	Señal del TMR0	1	Entrada externa.
6	Modo de interrupción externa.	0	No importa.
7	Resistencias pull-up.	0	Activado.

Valor fijado es lo que se programó en nuestro caso. Dado que el pre-escalador se asignó al watch-dog, no importa el valor que usemos.

El TMR0 cuenta los pulsos a la bajada, de acuerdo al bit 4.

El bit 5 determina que la entrada del TMR0 sea del exterior (pata 3).

El bit 6 se usa para la interrupción externa. En nuestro caso no se usa.

El bit 7 activa o desactiva resistencias en el puerto B las cuales están internamente conectadas al VCC. En nuestro caso las hemos activado para que no se necesite colocar resistencias al VCC en los interruptores (usamos las internas, llamadas pull-up).

El valor fijado es el 00101000 (0x28 en hexadecimal).

5.13. INICIANDO REGISTROS

El siguiente paso es iniciar algunas salidas y registros. RJ y D0 se ponen en “0”. Ponemos a cero los registros horas, minutos y segundos de modo que al encender nuestro reloj comenzará en 00:00 00.

Por último configuramos el registro del formato para nuestros 3 displays.

Aquí se indica la posición del punto y el número de dígitos visibles (2).

La última instrucción hace invisible el punto del display ya que no lo necesitamos y solo nos estorbaría.

5.14. DIBUJANDO EN LA PANTALLA

```
                    BORRAR_PANTALLA          ; Borrar la pantalla.
                    BOTONES_CONTROL 2        ; Dibujar 2 botones de control.
;---------- DIBUJAR BARRAS DE TITULOS ----------
                    BRILLO 0
                    CURSOR 22,1
                    CUADRO_LLENO 226,14
                    BRILLO 10
                    CURSOR 22,15
                    CUADRO_LLENO 226,26
```

Lo primero que se hace es borrar la pantalla, para esto utilizamos la macro BORRAR_PANTALLA. Luego dibujamos dos botones de control los cuales se dibujan automáticamente en la parte inferior de la pantalla.

Enseguida dibujamos dos barras de título en la parte superior de la pantalla. El color de Brillo a “0” (negro). Luego colocamos el cursor en las coordenadas 22, 1 y dibujamos un Cuadro Lleno con las coordenadas de la esquina inferior derecha en 226, 14, lo cual da como resultado una barra de color negro.

De la misma manera dibujamos otra barra un poco más abajo pero de color claro (Brillo=10).

5.15. ESCRIBIENDO TEXTO

Escribiendo texto dentro de las barras de títulos

```
;---------- TEXTOS DE LAS BARRAS DE TITULOS ----
                                    BRILLO 10
                                    CURSOR 30,4
                                    TEXTO_FORMATO 0
                                    BRILLO 3
                                    CURSOR 75,17
                                    TEXTO_FORMATO 28
```

El texto en la barra superior será de color claro (Brillo=10), ya que la barra es negra. Después de colocar el cursor en la parte izquierda de la barra, ejecutamos la macro TEXTO_FORMATO, la cual espera el offset de inicio en la tabla, el cual es 0, ya que iniciamos con el texto “HENRY …”.

A continuación cambiamos el color a uno más oscuro y escribimos en la barra inferior. Note que el offset de TEXTO_FORMATO es 28, ya que comenzaremos a escribir a partir del carácter número 28 de la tabla de textos.

5.16. DIBUJANDO EL ICONO

Dibujando el icono de la aplicación en la parte superior izquierda de la pantalla.

```
;---------- DIBUJAR ICONO --------------------
                                    CURSOR 0,0
                                    MARCO_ALTO_REL 20,25
                                    BRILLO 3
                                    CURSOR 2,5
                                    IMAGEN_2TONOS 16,16,0
```

Colocar el cursor en la parte superior izquierda (coordenadas 0, 0), luego enmarcamos nuestro icono con un pequeño marco realzado. Enseguida cambiamos el brillo a un color oscuro (Brillo=3) y por último ejecutamos la macro IMAGEN_2TONOS, donde indicamos que el tamaño de la imagen será de 16x16 puntos con offset de 0.

5.17. PREPARANDO EL DISPLAY

A continuación se dibuja el recuadro donde presentaremos el display de cuarzo

```
;-------------- RECUADRO DEL DISPLAY DE CUARZO -------
                CURSOR 30,80
                MARCO_BAJO_REL 185,140
                CURSOR 105,100
                CUADRO_LLENO 109,105
                CURSOR 105,115
                CUADRO_LLENO 109,120
```

Se enmarca el display con un recuadro en bajo relieve y luego dibujamos dos pequeños cuadros entre las horas y minutos para darle un aspecto más cercano a un reloj real de cuarzo. Con esto queda preparada la pantalla para desplegar la hora.

5.18. DESPLEGANDO LA HORA Y HABILITANDO INTERRUPCIONES

A continuación se despliega la hora actual, la cual será 00:00 00, debido a que ésta parte del programa se efectúa al inicio, sin embargo se habilita la interrupción por TMR0 y a partir de éste instante el reloj comenzará a funcionar

```
;-------------- DESPLEGAR LA HORA --------------------
RefHoras        CURSOR 60,95
                DISPLAY_CUARZO FormPunto,Cero,Cero,Horas
RefMinutos      CURSOR 120,95
                DISPLAY_CUARZO FormPunto,Cero,Cero,Minutos
                CURSOR 165,117
                DISPLAY_DECIMAL FormPunto,Cero,Cero,Segundos
                MOVLW 0xA0      ; Habilitar interrupción por RTCC.
                MOVWF INTCON
                MOVLW 196               ; (206 para frecuencia AC=50 Hz).
                MOVWF TMR0
```

5.19. MANEJANDO LOS INTERRUPTORES

En esta última parte del código se verifica si se ha presionado alguno de los dos interruptores, si esto sucede se actúa en consecuencia incrementando los minutos o las horas.

```
;-------------- VERIFICAR LOS INTERRUPTORES ---------
VerifSW         BTFSS SW1
                GOTO PresHoras
                BTFSC SW2
                GOTO VerifSW
                CLRW
                MOVWF INTCON            ; No interrupciones
                CLRF Segundos
```

```
                INCF Minutos,1
                MOVLW 60
                SUBWF Minutos,0
                BTFSC STATUS,Z
                CLRF Minutos
                PRES_BOT_CONT 2
                GOTO RefMinutos
PresHoras       CLRW
                MOVWF INTCON            ; No interrupciones.
                INCF Horas,1
                MOVLW 24
                SUBWF Horas,0
                BTFSC STATUS,Z
                CLRF Horas
                PRES_BOT_CONT 1
                GOTO RefHoras
                END
```

Si se ha detectado un interruptor presionado, inmediatamente se anulan las interrupciones, con el fin de procesar el código de refresco del display.

Si dentro del programa principal estamos enviando instrucciones a la IG corremos el riesgo de que al mismo tiempo ocurra una interrupción. Si la subrutina de interrupción trata también de enviar datos a la IG, ésta se confundirá ya que esperaba datos del programa principal y de pronto llegan datos de la subrutina de interrupción. Esto ocasionará la ejecución aleatoria de instrucciones y la pantalla se corromperá junto con el programa.

Por este motivo se anulan las interrupciones y solo después de refrescar el display respectivo (horas o minutos) se vuelven a activar.

Proyecto terminado

Figura V.37. Reloj digital funcionando

CONCLUSIONES

1. Para poder generar una señal video, es necesario un mínimo hardware para poder generar niveles de señal entre 0 y 1V, conversores CDA.

2. Para obtener color negro en un televisor se necesita un Voltaje de 0.3V en cambio para el color blanco el voltaje es de 1.0 V y los valor intermedios son los demás colores.

3. No podemos escribir en la memoria RAM cuando el PIC esté presentando la señal en la pantalla, ya que alteraríamos el video al pretender que los 2 microcontroladores escriban en la memoria RAM al mismo tiempo.

4. En un microcontrolador no es posible almacenar la imagen entera en memoria como en tarjetas gráficas, la imagen tiene que hacerse generado barridos de la señal video a través de la pantalla. La generación de las señales video en software en un procesador tan simple es difícil, sólo las imágenes muy simples pueden ser creadas. La ventaja es que es absolutamente barata.

5. Para la interpretación de instrucciones se utilizó un microcontrolador de Atmel debido a que son mucho más veloces que un pic de microchip en un factor de 4 a 1.

6. Es importante tener tres memorias que puedan ser utilizadas desde el exterior pues hay varios proyectos que necesitan de ellas, incluso para guardar datos que luego sean dibujados en pantalla.

RECOMENDACIONES

1. Para enviar o recibir un byte de la IG se tiene que tener presente que el bit LR (listo para recibir se encuentre en uno), de lo contrario alteraríamos la señal de video.

2. Para la graficar en pantalla hay que tener presente que se cuenta con una resolución de 228 x 216 pixeles para no salirse de la misma y obtener resultados no deseados.

3. Si se desea alimentar tanto la IG como el proyecto exterior es recomendable suministrara la IG con una fuente de alimentación de 12 Voltios.

4. Si va a cambiar de proyecto, primero deber apagar la IG y luego conectar en el nuevo proyecto el cable de comunicaciones con la IG, así no se distorsionara la imagen.

5. Cuando se implemente un proyecto exterior con una fuente propia para el mismo, tener presente juntar los GND con la IG.

6. Si se utiliza las macros disponibles para el software generando proyectos en MPLAB, tener encuentra que Default – Radix este en decimal para que MPLAB se encargue de hacer la conversión a hexadecimal.

7. Definir siempre los pines que se comunicaran con la IG, estos son D0, LR, RJ.

8. D0 estará siempre como salida, a menos que reciba datos de la IG, lo cual después de recibirlos se pondrá como salida.

RESUMEN

El desarrollo de mi investigación es la construcción de una tarjeta de interfaz gráfica para televisión utilizando microcontroladores, se diseñóeimplementó para dar una solución a la necesidad que se nos plantea cuando hacemos proyectos con microcontroladores y necesitamos mostrar datos, imágenes, caracteres en algún dispositivo, una buena solución ha sido el televisor, que además de contener una buena resolución hasta 320 por 240 pixeles, puede desplegar colores, caso contrario que no sucede con los lcd`s que utilizamos para el despliegue de información.

Mediante el método científico se ha podido determinar los factores que influyen en la generación de patrones de televisión, los materiales utilizados son invariablemente microcontroladores, el uno un PIC16C55 para generar los patrones de video, las cuales son sincronía vertical, sincronía horizontal y blanqueo, y otro microcontrolador de la serie Atmel (ATMega8515) que es el intérprete de instrucciones a desplegar en el televisor.

Los resultados mostrados fueron de una concepción gráfica muy aceptable, de un 60% de efectividad en despliegue de señal de video, pudiéndose ver en pantalla gráficos de 2 a 16 tonos de gris, líneas, cuadrados, marcos de alto y bajo relieve, texto con formato, procesamiento de datos digitales, botones de control.

En conclusión se puede generar señal de video con microcontroladores, pero teniendo en cuenta que estos dispositivos son limitados en lo que a memoria se refiere y sólo las imágenes muy simples pueden ser creadas.

Se recomienda que en su implementación se utilice microcontroladores Atmel de buenas características de procesamiento, ya que son mucho más veloces que un microcontrolador PIC y su puede generar mejor refresco de pantalla.

SUMMARY

This research proposes a television interface graphic card through the use of microcontrollers. It has been designed and incorporated to solve those problems that are common when we make projects with microcontrollers and we need to display data, images, and characters on a device; a good solution is the television set which besides its excellent 320 – 240 pixel resolution, can display colors. This does not occur with the LCD's that are usually used to display information.

Through the scientific method, it has been possible to determine the factors that intervene in the generation of television standards. The tools generally consist of microcontrollers, especially a PIC16C55 to generate video standards which are vertical synchrony, horizontal synchrony and blanching, another microcontroller from the series Atmel (AT mega 8515) which is the interpreter for the instructions to be displayed on the television screen.

The results were of a good graphic conception with 60% effectiveness at displaying video signal, visible on graphic screen of 2 to 16 gray tones, lines, high and low relief frames, formatted texts, digital data processing, and control keys.

In conclusion, it is possible to generate video signal through microcontrollers, taking into account that the memory of such devices is insufficient and only the most simple images can be created.

It is recommended to use microcontrollers Atmel with good processing features, as they are much faster than a PIC microcontroller and can generated a better screen renewal.

GLOSARIO

BIT	Esundígitodelsistema denumeraciónbinario
BYTE	Es la unidad fundamental de datos en los ordenadores personales, un *byte* son ocho bits contiguos
CUARZO	El cuarzo es un mineral compuesto de dióxido de silicio
COMUNICACIÓN	Es la comunicación que se establece entre dos sujetos
MICROCONTROLADORES	Un microcontrolador es un circuito integrado o chip que incluye en su interior las tres unidades funcionales de una computadora: unidad central de procesamiento, memoria y unidades de E/S.
DISPLAY	Se llama visualizador, display en inglés, a un dispositivo de ciertos aparatos electrónicos que permite mostrar información al usuario,
INTERFAZ	En electrónica, telecomunicaciones y hardware, una interfaz es el puerto (circuito físico) a través del que se envían o reciben señales desde un sistema o subsistemas hacia otros
HARDWARE	Corresponde a todas las partes físicas y tangibles de un dispositivo eléctrico, electrónico, electromecánico
SOFTWARE	Comprende el conjunto de los componentes lógicos necesarios para hacer posible la realización de tareas específica

ANEXOS

ANEXO1

Lista de materiales

Resistencias

R1, R2, R15 - 1kΩ

R4, R7, R8, R11, R12 - 1.8kΩ

R18 - 2.7kΩ

R20, R21, R24, R25 - 3.3kΩ

R3, R5, R6, R9, R10, R17 - 3.6kΩ

R23 - 100Ω

R16 - 220Ω

R26 - 270Ω

R13, R14, R19 - 390Ω

R22 – 470Ω

Condensadores (todos en µF)

C8, C9, C10 - 0.1µF - Cerámico

C3 - 10µF - Electrolítico x 25V

C1, C2, C4, C5 - 15pf - Cerámico

C7 - 1000µF - Electrolítico x 25V

C6 - 100µF - Electrolítico por 25V

Integrados

U1 - PIC16C55 - Microcontrolador 20MHz

U2 - 4066 - Interruptor analógico

U3 - ATMega8515 - Microcontrolador 16MHz

U4, U5 - AT24C256 EEPROM (Opcionales)

U6 - 7805 - Regulador de voltaje

U8 - 628128 - Memoria RAM 128K U4,

Diodos y transistores

D1 - LED de 5mm color rojo (Encendido)

D2 - 1N34 - Diodo de conmutación rápida

Q1, Q2 - 2N2222 - Transistores

Conectores

J1 - Conector programación (Opcional)

J2 - Conector DIP de 16 patas

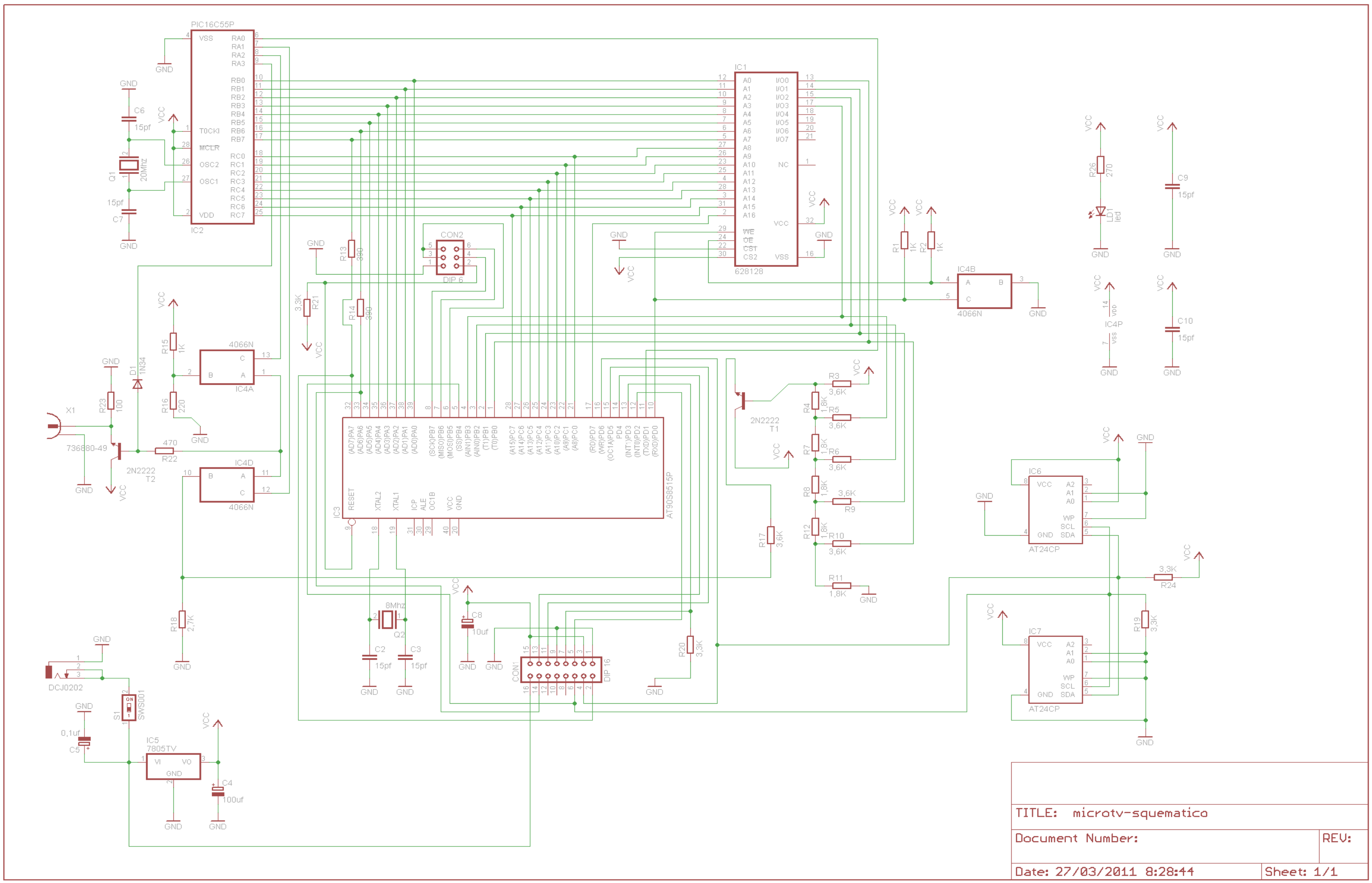
J3 - Conector RCA - Salida Video

U7 - Conector de alimentación - Banana Invertida

Cristales

Y2 - Cristal 16MHz

Y1 - Cristal 20MHz


PIC16C55P
IC1
628128
AT90S8515P
IC3
4066N
IC4A
IC4B
IC4D
IC6
IC7
AT24CP
IC5
7805TV
CON1
CON2
DIP 6
DIP 16
2N2222
T1
T2
X1
736880-49
DCJ0202
S1
SWS001
LD1
8Mhz
20Mhz
VCC
GND
TITLE: microtv-squematico
Document Number:
REV:
Date: 27/03/2011 8:28:44
Sheet: 1/1

BIBLIOGRAFÍA

LIBROS

1. **ANGULO,J.M. ROMERO,S.,** Microcontroladores PIC., 2da.ed., McGrawHill - Madrid., 2003., Pp.191-198

2. **BOYLESTAD,L.,** Electrónica Teoría de Circuitos., 6a. ed., Prentice Hall - MéxicoDF., 1997., Pp.560-577.

3. **REYES,C**., Aprenda a programar Microcontroladores., Gráficas Ayerve – Quito Ecuador., 2004., Pp.201.

BIBLIOGRAFÍAINTERNET

1. COMO FUNCIONA EL TELEVISOR

   http://www.librosmaravillosos.com/comofunciona/comofunciona41.html

   2010 – 11 – 20

2. SEÑALES DE VIDEO

   http://www.electronica2000.net/curso_elec/leccion68.htm

   2010– 11 – 30

3. COMO GENERAR SEÑAL DE VIDEO CON PIC

   http://www.rickard.gunee.com/projects/

   2011 – 01 – 15

4. GENERAR SEÑAL DE VIDEO NTSC

   http://www.acm.uiuc.edu/sigarch/projects/breakout/

   2011 – 01 - 15