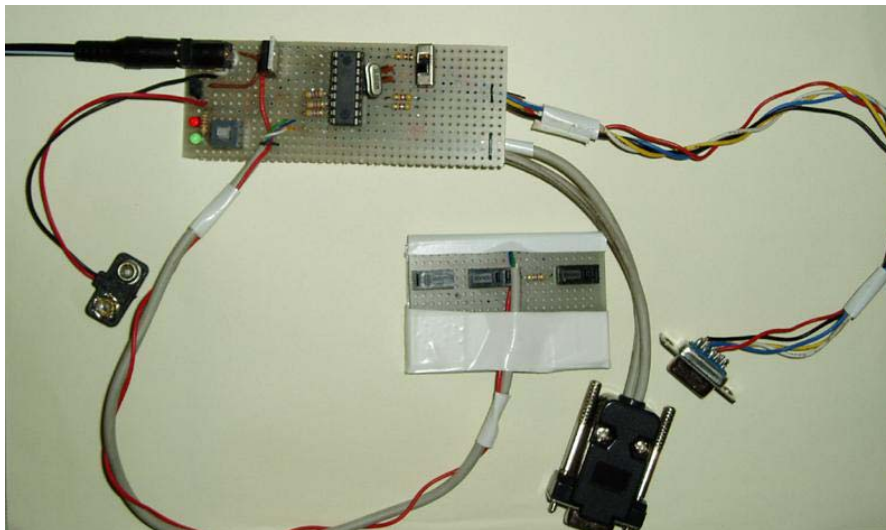


Pic VGA Trainer



Diseño de Sistemas Basados en Microprocesador

Realizado por:

José M^a Viera Rivero
a2945

jose.viera@terra.es

INDICE

1.- OBJETIVOS	2
2.- DESCRIPCIÓN OBJETIVO 1	3
3.- DESCRIPCIÓN OBJETIVO 2	8
4.- DESCRIPCIÓN OBJETIVO 3	10
5.- DESCRIPCIÓN OBJETIVO 4	12
6.- DESCRIPCIÓN DEL CÓDIGO FUENTE	13
6.1- FICHERO F84.ASM	14
6.2- FICHERO TITULO.INC	15
6.3- FICHERO ENEMIGOS.INC	16
6.4- FICHERO DISPAROS.INC	17
6.5- FICHERO NAVE.INC	18
7.- DIARIO DE INCIDENCIAS	18

1.- OBJETIVOS

El objetivo fundamental de la práctica era el de generar una señal de video en formato VGA mediante un PIC 16F84. La práctica comenzó con el estudio del estándar VGA y la generación de una imagen estable. Las limitaciones de potencia del PIC hacen imposible generar una imagen que se adapte fielmente a una resolución estándar. Se eligió una resolución de “640x480” píxeles y 60 Hz. de refresco como patrón a imitar para la generación de la señal aunque la resolución efectiva es muchísimo menor. La posibilidad de los monitores digitales para adaptar la señal recibida al área visible de la pantalla resta dificultad a la práctica por lo que se prefirió el empleo de monitores analógicos que nos forzasen a adaptar la señal emitida para conseguir una imagen nítida y estable. Esto permite experimentar y aprender en un entorno mucho más fiel a la realidad en la que tengamos que adaptarnos a un protocolo establecido. Existen algunas diferencias en el hardware necesario para un tipo u otro de monitores que se explicarán más adelante.

Pronto se hizo patente que para generar una imagen de mediana calidad sería necesario que el PIC corriese bastante rápido y que los 4Mhz de que disponen las placas de entrenamiento del laboratorio de Señales y Sistemas eran insuficientes (ofrecen una resolución horizontal de 24 píxeles). Si a esto añadimos el gran inconveniente que suponía el tener que desplazar un monitor para las pruebas queda más que justificada la necesidad de marcarse un segundo objetivo, el diseño y montaje de un circuito que se ajustase a las necesidades planteadas y en el que el PIC funcionase de manera autónoma.

Una vez logrado este segundo objetivo y tras haber comprobado el enorme incremento que supuso en términos de rendimiento, se optó por optimizar el circuito dotándolo de la capacidad de programar y leer el microprocesador. De esta manera nos

desvinculamos de cualquier otro circuito programador aumentando las capacidades y versatilidad del circuito. La consecución de este tercer objetivo aceleró el proceso de prueba del software programado ya que con solo conmutar un interruptor podíamos elegir entre programar, leer o borrar el PIC o hacer funcionar el programa.

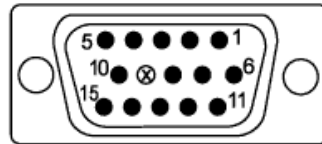
Dado que la generación de la señal de video VGA consistía en generar 5 señales de salida se optó por completar el aprendizaje del funcionamiento del PIC modificando el circuito con señales de entrada. La incorporación de dichas señales permitió la generación de un programa interactivo que modificase la señal emitida en función de las entradas. De esta manera nació un cuarto objetivo que consistía en la creación de un software de mencionadas características. Este software evolucionó hasta convertirse en un juego de naves al estilo del antiquísimo “Space Invaders” al que se llamó “F84 Invaders” en honor al chip que lo ejecuta.

2.- DESCRIPCIÓN OBJETIVO 1 – Generar señal VGA

El funcionamiento de las pantallas es similar al de los receptores de televisión: existe un tubo de rayos catódicos (CRT, del inglés Cathode Ray Tube), en el que un haz de electrones recorre la pantalla, que posee un recubrimiento de fósforo. Al incidir el haz de electrones en la capa de fósforo, excita, de forma proporcional a la señal que llega al periférico, cada punto de la pantalla, dando lugar a figuras o letras. El barrido de la pantalla se hace mediante trazas horizontales. Dando un pulso de sincronización vertical el haz de electrones se posiciona en el extremo superior izquierdo del monitor. El haz se irá desplazando hacia la derecha hasta que se produzca un pulso de sincronización horizontal tras el que el haz vuelve a la izquierda bajando a la siguiente línea. La diferencia radica en que en un monitor VGA la información del color se envía de manera separada mediante tres componentes (una señal para el rojo, otra para el

verde y otra para el azul) en lugar de formar parte de una única señal analógica. Modificando la amplitud de cada señal se obtienen los diferentes colores.

El interfaz de conexión es un conector de 15 patillas y en él se observan claramente las diferentes señales necesarias. A continuación se muestra el interfaz usado por IBM en sus monitores VGA y SVGA.



Female

Señal	Pin	Notas
RED video	1	Señal analógica, impedancia de 75 ohmios
GREEN video	2	Señal analógica, impedancia de 75 ohmios
BLUE video	3	Señal analógica, impedancia de 75 ohmios
Monitor ID #2	4	(véase nota 1)
Masa general	5	Masa del circuito general de video
RED ground	6	Masa para la señal RED (independiente)
GREEN ground	7	" " " GREEN
BLUE ground	8	" " " BLUE
KEY	9	(No se usa)
SYNC ground	10	Retorno para las señales de sincronización (véase nota 2)
Monitor ID #0	11	(véase nota 1)
Monitor ID #1	12	(véase nota 1)
Horizontal Sync	13	Niveles digitales (0 - 5 voltios)
Vertical Sync	14	Niveles digitales (0 - 5 voltios)
Not Connected	15	(No usado)

Los niveles de las señales RED, GREEN y BLUE están cargados con una resistencia de 75 ohmios (0,7 voltios) que es lo que hace que el monitor tenga una imagen gris cuando no esta enchufado el conector. El dispositivo que genere las señales debe suplir esa impedancia para que esos 0,7 voltios se queden en 0 y la pantalla quede en negro.

Nota 1: Los bits ID fueron usados por IBM para identificar el modelo de monitor. Con el tiempo este método cayó en desuso. Actualmente los monitores se identifican empleando el bus I2C.

Nota 2: Según cual sea la resolución empleada las señales de sincronismo pueden estar polarizadas a la inversa , es decir, en unas se entenderá un nivel de 0 voltios como una ausencia de señal mientras que en otras serán los 5 voltios quienes marquen una ausencia de señal. En el caso de la resolución de 640x480 se entenderá 5 voltios como nivel de reposo y el paso a 0 voltios como el inicio de un pulso de sincronización.

El circuito a diseñar debe disponer de lo siguiente:

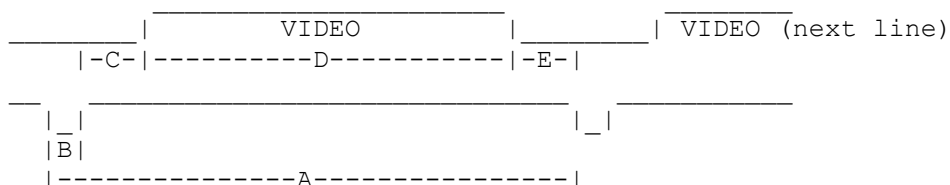
- Señal que mande información referente al color rojo
- Señal que mande información referente al color verde
- Señal que mande información referente al color azul
- Señal que indique un pulso de sincronización Vertical
- Señal que indique un pulso de sincronización Horizontal
- Masa de las señales de color
- Masa general del circuito

Una vez definidas las especificaciones físicas mínimas necesarias pasamos al estudio del protocolo que deben cumplir las señales.

INFORMACIÓN OBTENIDA DEL MANUAL DE UN MONITOR HP

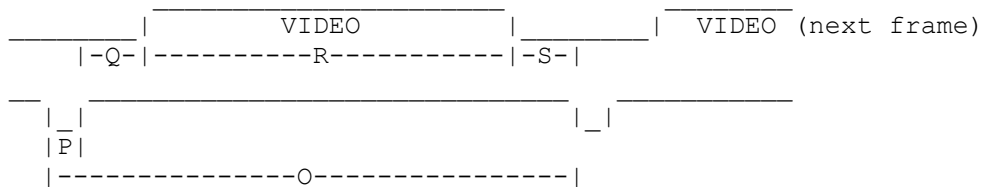
Horizontal Timing

Horizontal Dots	640	640	640	
Vertical Scan Lines	350	400	480	
Horiz. Sync Polarity	POS	NEG	NEG	
A (us)	31.77	31.77	31.77	Scanline time
B (us)	3.77	3.77	3.77	Sync pulse lenght
C (us)	1.89	1.89	1.89	Back porch
D (us)	25.17	25.17	25.17	Active video time
E (us)	0.94	0.94	0.94	Front porch



Vertical Timing

Horizontal Dots	640	640	640	
Vertical Scan Lines	350	400	480	
Vert. Sync Polarity	NEG	POS	NEG	
Vertical Frequency	70Hz	70Hz	60Hz	
O (ms)	14.27	14.27	16.68	Total frame time
P (ms)	0.06	0.06	0.06	Sync length
Q (ms)	1.88	1.08	1.02	Back porch
R (ms)	11.13	12.72	15.25	Active video time
S (ms)	1.2	0.41	0.35	Front porch



La señal horizontal tiene una frecuencia de 31,4 Khz (es decir, tiene una duración de 31.77 us como puede verse en la tabla). A lo largo de eso 31.77 us el haz de electrones avanza desde la izquierda hasta el principio del área visible, pinta cada píxel del área visible, avanza hasta el borde derecho y se produce el pulso horizontal.

Con una frecuencia de 4Mhz el tiempo de ciclo del PIC es de 1 us con lo cual sólo dispondríamos de unas 32 instrucciones por línea, de las cuales sólo podríamos aprovechar 24 para dibujar en la pantalla. Por este motivo se usó un reloj de cuarzo de 16Mhz que hace funcionar internamente el PIC a 4Mhz. El tiempo de ciclo es de tan solo 0.25 us aumentando la cantidad de instrucciones por línea de 32 a 131.

A continuación se muestran los cálculos pertinentes para conocer la duración de las diferentes señales:

Reloj de 16 Mhz , tiempo de ciclo $0,25 \cdot 10e^{-7}$

Datos de sincronización horizontal

Duración total de la línea = 32,700 us → 131 ciclos

Duración del pulso horizontal = 4,480 us → 18 ciclos

Duración del borde izquierdo = 2,360 us → 9 ciclos

Duración del área activa = 25,660 us → 103 ciclos

Duración del borde derecho = 0,200 us → 1 ciclo

Datos de sincronización vertical

Duración total de la pantalla = 16,650 ms → 66600 ciclos

Duración del pulso vertical = 0.065 ms → 260 ciclos

Duración del borde superior = 0.816 ms → 3264 ciclos

Duración del área activa = 15,672 ms → 62688 ciclos

Duración del borde derecho = 0,097 ms → 388 ciclos

Las pruebas prácticas han demostrado que es posible reducir la duración de algunas de estas señales en beneficio de otras. Por ejemplo, la duración del pulso horizontal puede reducirse de 18 a 10 ciclos ganando así 8 ciclos de área activa. Lo mismo ocurre con el borde izquierdo (de 9 a solo 4). Los bordes superior e inferior se consiguen pintando líneas negras.

Finalmente se ha ajustado la imagen a un monitor analógico IBM 1815 disponiendo de un área activa de 117 ciclos por línea y 480 líneas visibles (508 en total).

Cabe destacar que antes de dar un pulso horizontal o vertical es recomendable eliminar cualquier componente de color para evitar que se marque el trazo durante el retroceso del haz de electrones.

3.- DESCRIPCIÓN OBJETIVO 2 – Diseñar y crear un circuito autónomo

En la especificación del fabricante del microcontrolador PIC 16F84 puede observarse que lo único que necesita el PIC para funcionar es alimentación y un oscilador.

Aunque existen modelos capaces de funcionar a bajo voltaje, una tensión de 5 voltios es válida para cualquier microcontrolador de este tipo. Dado que no siempre se tiene una fuente de alimentación regulable al alcance, lo mejor es utilizar un adaptador o una pila de 9 voltios estándar conectada a un regulador de tensión que estabilice la corriente a 5 voltios. En este caso he usado regulador LM7805. Para elegir entre la alimentación con una pila de 9 voltios o un transformador de corriente se dispone de un jumper.

El oscilador está interpuesto entre las patillas OSC1 y OSC2 tal y como indica el fabricante y unido a masa mediante condensadores cerámicos. Tanto los de 33pF como los de 22 pF han dado buen resultado no apreciándose irregularidades con relojes de 4, 8 y 16 Mhz.

Las señales de salida son digitales por lo que es necesario un convertor digital-analógico que las ajuste a la especificación física del monitor. Basta una resistencia de 470 ohmios para realizar esta operación a la vez que se consiguen aproximadamente los 0,7 voltios que indican el máximo nivel de color por canal.

$$V_{in} = \frac{75\Omega}{75\Omega + 470\Omega} * 5V = 0.68V$$

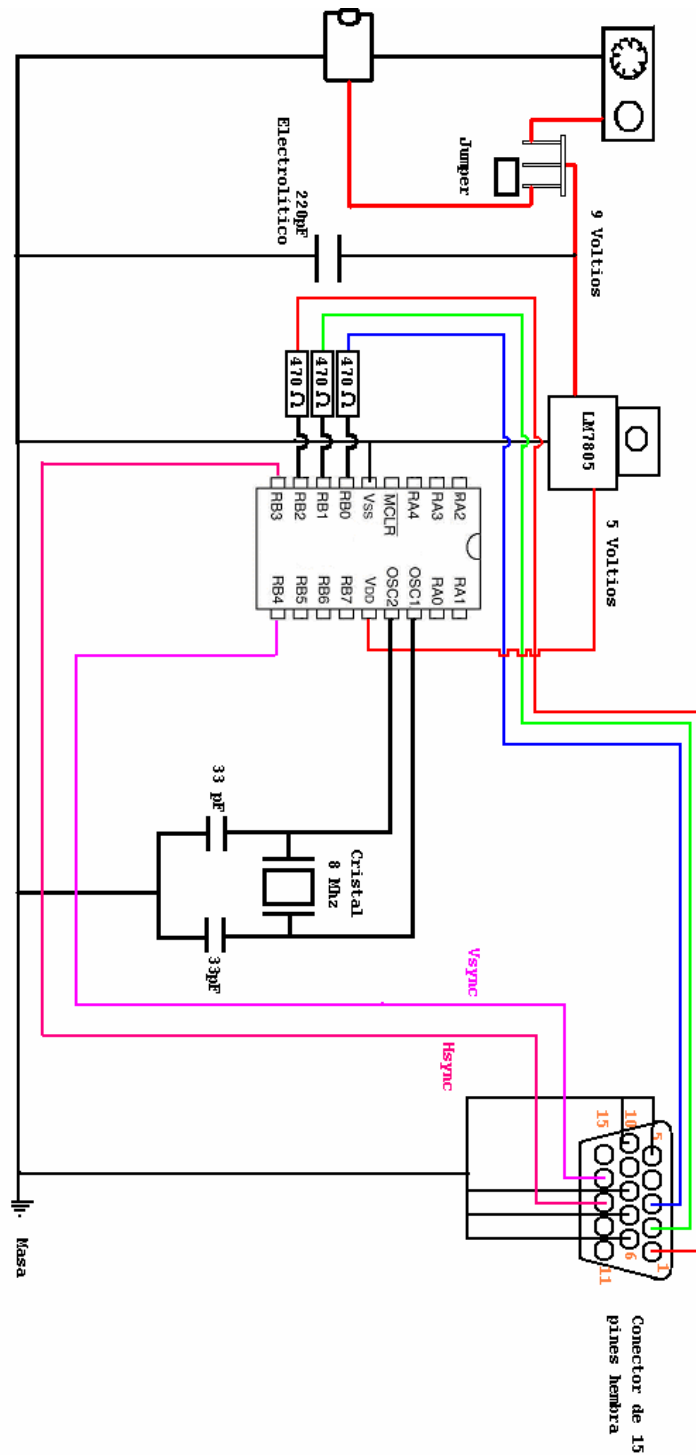
Con esto ya quedan definidos los componentes mínimos:

- 5 Voltios entre Vdd y Vss.
- Reloj de 16 Mhz en modo HS con dos condensadores cerámicos de 33pF
- Tres salidas conectadas cada una a una resistencia de 470 ohmios.

- Dos salidas reservadas para conectar a las entradas HSync y VSync del monitor.

- Conectar cada señal a la patilla adecuada de un conector hembra de 15 pines.

El esquema es el siguiente y ha sido llamado “PIC VGA Trainer”



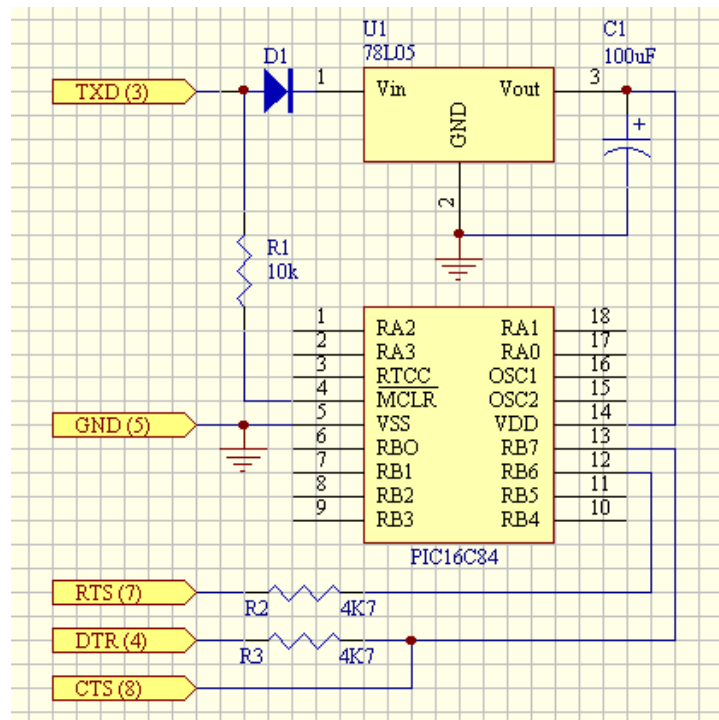
El condensador de 220pF no es necesario si la fuente de voltaje es medianamente estable.

En los monitores analógicos se puede prescindir de conectar el pin 5 a tierra. Para los digitales deberemos de conseguir diferentes voltajes entre dicho pin y la patilla 10 para indicar al monitor si debe estar apagado, en modo hibernación o encendido (testado en un monitor Samtron 55E)

4.- DESCRIPCIÓN OBJETIVO 3 – Añadir la capacidad de programar al circuito

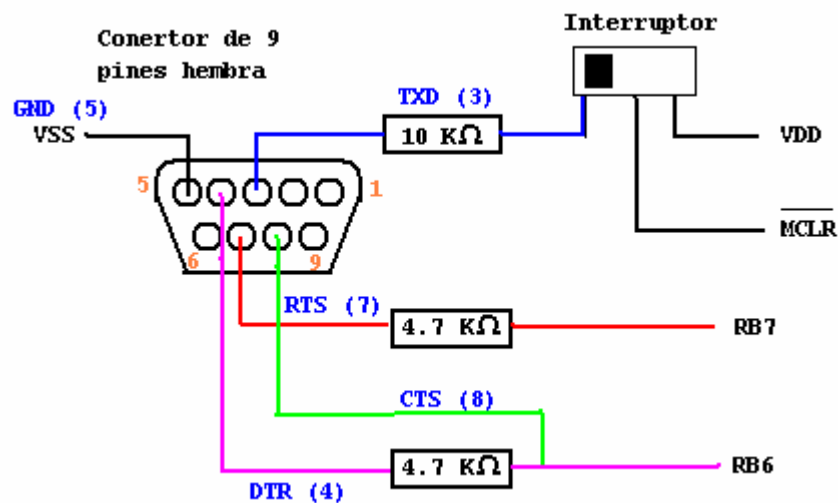
Siguiendo la especificación del fabricante debemos tener una alimentación mínima de 4.5 y máxima de 5.5 para escribir en el PIC. La programación es en serie y se usa el RB6 para indicar el reloj y RB7 para enviar los datos. MCLR debe tener una tensión de unos 8 voltios.

El programador más sencillo y usado para programar los PICs de la familia 16X84 es el JDM (una evolución del Ludipipo). Este programador se alimenta directamente del puerto serie (o paralelo según el modelo) y apenas necesita de algunas resistencias. A continuación se muestra un esquema básico de dicho programador:



(extraído de <http://www.rentron.com/Myke4.htm>)

El problema de este esquema es que la alimentación está muy poco estabilizada por lo que es mucho más que probable que se produzcan numerosos errores de programación en aquellos equipos cuyo puerto serie no pueda ofrecer la suficiente intensidad de corriente (principalmente ordenadores antiguos). Dado que el “Pic VGA Trainer” tiene su propia alimentación se optó por modificar el esquema anterior suprimiendo la parte de la alimentación para utilizar la del circuito (mucho más fiable).



Para utilizar el circuito en modo programador debe cumplirse lo siguiente:

- El circuito debe estar correctamente alimentado y encendido (led verde encendido)
- El interruptor debe estar en posición 1 (alineado con la patilla 1 y 18 del pic)
- El conector de 9 pines debe estar enchufado a un puerto serie del ordenador.

Existen numerosos programas capaces de utilizar este programador. **ICProg**, un programa de libre distribución, es uno de ellos. Funciona bajo plataforma Windows y puede ser descargado desde aquí:

<http://www.ic-prog.net/index1.htm>

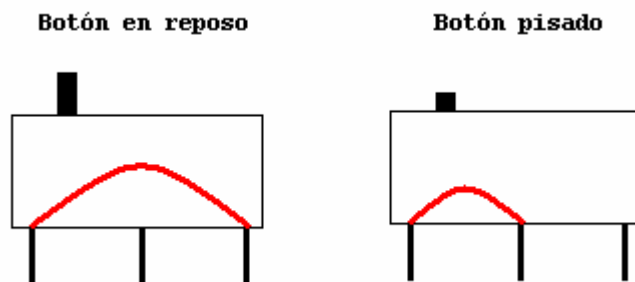
De las numerosísimas veces en que fue programado el chip jamás se produjo un solo error de programación o lectura.

5.- DESCRIPCIÓN OBJETIVO 4 – Programar un software interactivo de prueba.

Algunas patillas del puerto A del microcontrolador fueron configuradas en modo salida para recibir señales de estímulo (RA0, RA1 y RA2). Dado que es mucho más fácil detectar una ausencia de señal que un umbral el PIC está preparado para que las señales de entrada tengan su nivel de reposo en 5 voltios.

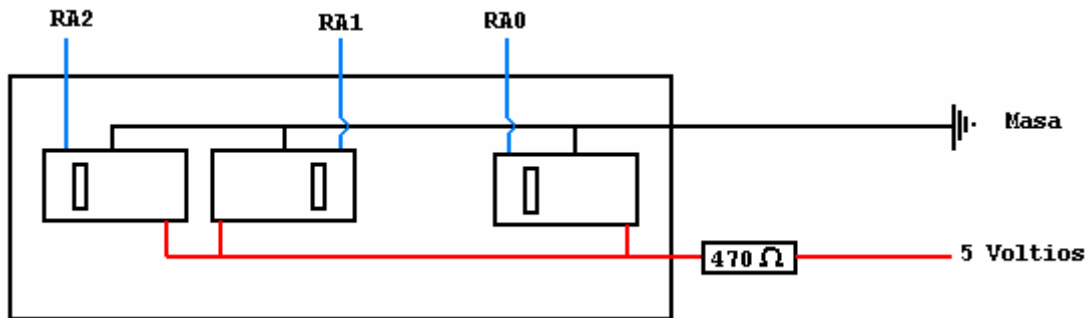
Podría pensarse que la diferencia entre el nivel lógico 0 (5 voltios) y 1 (0 voltios) es suficientemente grande y que se podría trabajar con un puerto de entrada como si de uno de salida se tratase. Es decir, estar normalmente a 0 y dar un pulso de 5 voltios para señalar un 1. Sería sólo cuestión de adaptar el programa sustituyendo instrucciones del tipo “btfss” por “btfsc”. La realidad despeja cualquier duda y obliga a adaptarse a la especificación del fabricante ya que el microcontrolador entra en un estado de incertidumbre si alguno de sus pines de salida no están conectados a 5 voltios. El PIC toma caminos equivocados e incluso se producen modificaciones en registros que, aparentemente no debería tocar. Se entiende que el puerto de entrada es bastante sensible y cualquier pequeña corriente (quizá inducida por la proximidad de las soldaduras y cables) puede ser interpretada como un elemento de señal.

Debido a este requerimiento no es posible utilizar un interruptor o botón común como generador de estímulos en las señales de entrada. Debemos disponer de un botón que conmute entre dos posibles valores de tensión.



Conectamos el cable que va al pin del microcontrolador en la patilla izquierda del botón. Un cable a masa en el centro y otro a 5 voltios en el derecho.

El esquema para un pequeño mando de tres botones sería el siguiente:



Si la tensión de 5 voltios se va a tomar desde el mismo punto que la que alimenta al PIC es importante añadir una resistencia ya que si no la pusiéramos al pisar el botón existiría un enlace directo entre +5v y masa y el microcontrolador podría quedarse sin alimentación. En función de las opciones de programación que hubiésemos elegido el PIC podría resetearse o continuar la ejecución sin ofrecer ninguna garantía de correcto funcionamiento.

6.- DESCRIPCIÓN DEL CÓDIGO FUENTE

El código resulta complejo debido a la densidad y cantidad de instrucciones necesarias (prácticamente se ocupó el total de memoria disponible) pero no destaca por la utilización de elementos complejos de programación.

Una sola instrucción de más o de menos puede provocar que en cada pasada se incremente un desfase que provoque desde pequeños errores de dibujado a grandes interferencias que hagan irreconocible la imagen. Debido a esto resultó imposible utilizar directivas de compilación (es necesario contabilizar todas y cada una de las instrucciones que formen parte del código y no podemos dar margen a que el compilador genere código automático). Por su propia naturaleza se hizo imposible el

utilizar interrupciones. Resultaba muy costoso en términos de memoria mantener un software que supiese cuantos ciclos habían pasado desde el último pulso horizontal hasta que se produjo la interrupción, que continuase ejecutando algo hasta los 117 ciclos, produjera un pulso horizontal y volviese al punto de partida con los mismos ciclos transcurridos a la vez que realizaba el tratamiento de la interrupción.

El uso de tablas estuvo destinado a mostrar por pantalla el número de puntos conseguidos por el jugador pero, finalmente y tras muchos intentos de optimización del código, fue imposible liberar la suficiente memoria de instrucciones.

El código fuente está dividido en cinco ficheros:

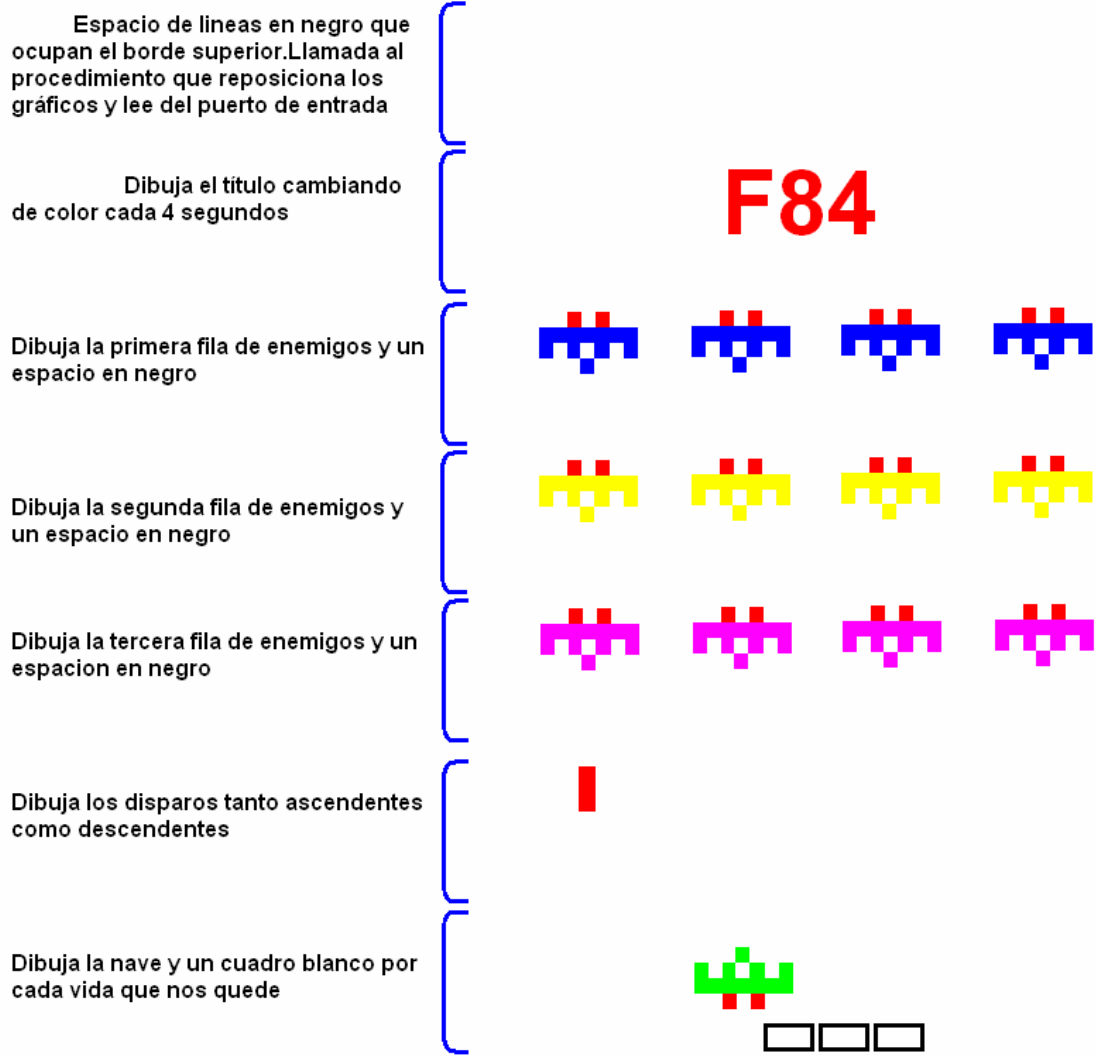
- **F84.asm** (fichero principal)
- **titulo.inc**
- **enemigos.inc**
- **disparos.inc**
- **nave.inc**

6.1 - FICHERO F84.ASM

Es el fichero principal, en él se declaran todas los registros necesarios , algunas funciones de apoyo como las que generan retardos y los pulsos de sincronización horizontal y vertical y se lleva a cabo el dibujado de la pantalla.

El cuerpo del programa consiste en un bucle infinito en el que se produce un pulso Vertical, se dibujan aproximadamente 510 líneas (cada una acabada con un pulso horizontal), se restauran valores de variables para la próxima pasada, se incrementan contadores y se retorna el punto de inicio.

El siguiente gráfico muestra las tareas que realiza el bucle principal en cada pasada:



6.2 FICHERO TITULO.INC

Este fichero lo único que hace es dibujar las letras “F84” que dan título al juego pintándolas de un color u otro en función del contenido de un registro llamado “colortitulo”.

Un pixel en sí es demasiado delgado por lo que cada línea de un gráfico es en realidad el producto de un bucle que pinta la misma línea varias veces. De esa manera se entiende que para pintar las letras “F84” con una fuente de 5x3 se necesitan cinco bucles consecutivos.

Dado que la cantidad de color por canal que tenemos en cada señal de color es constante sólo disponemos de 8 colores (uno por cada posible combinación). Siendo éstos:

	RB2	RB1	RB0	
	0	0	0	Negro
	0	0	1	Azul marino
	0	1	0	Verde
	0	1	1	Azul celeste
	1	0	0	Rojo
	1	0	1	Lila
	1	1	0	Amarillo
	1	1	1	Blanco

6.3 - FICHERO ENEMIGOS.INC

En este fichero se encuentra un procedimiento que dibuja una fila de enemigos.

Se tienen tres registros: filauno, filados y filatres en los que se van a marcar las posiciones de las naves enemigas en cada línea. Los cuatro bits menos significativos de cada registro son quienes van a identificar si debemos pintar una nave (marcado con un 1) o un espacio en negro (un 0).

Lo que hace el programa principal es cargar el contenido del registro filauno en otro llamado registroaux que es con el que trabaja este procedimiento antes de llamarlo. Luego hace lo propio con filados y filatres. Antes de llamar a enemigos debemos cargar en el registro "color" el color con que queremos que se dibujen. (Lo que pongamos en color va a ser cargado en el puerto B por lo que al margen del color que pongamos en

los bits 2,1 y 0, los bits 3 y 4 deben ser un 1 para evitar producir un pulso de sincronización)

6.4 - FICHERO DISPAROS.INC

En este fichero se encuentra el procedimiento “muevedisparos” que es quien realmente realiza las tareas “dinámicas” del juego.

Disponemos de tres registros que van a marcar a lo largo de tres filas (cada fila ocupara un ancho de 45 líneas) la posición de un disparo generado por el jugador y otros tres para los disparos de las naves enemigas.

Cada vez que un contador llegue a un valor establecido los valores de estos registros son rotados entre sí para generar el movimiento de los disparos. La rotación se produce en un sentido para los disparos del jugador (filtres a filados, filados a filauno y limpiar filatres) y en el contrario para los de las naves enemigas.

Si en la ultima fila de los disparos del jugador hay algún 1 y en la misma posición de la ultima fila de las naves enemigas también lo hay entonces hemos dado a la nave y ponemos un 0 en dicha posición para que no vuelva a ser pintada.

Los valores de las filas de las naves son modificados para hacer que las naves de las filas traseras avancen a las posiciones libres (marcadas con un 0) de las filas delanteras. Si la que es alcanzada es la nave del jugador entonces lo que se hace es decrementar el número de vidas disponibles. Si el registro “vidas” tenemos marcados con un 1 los tres bits menos significativos es que tenemos tres vidas, si solo hay unos en los dos menos significativos es que tenemos dos, etc. (con un “rrf” en caso de alcance es suficiente)

Existe un segundo procedimiento llamado “pintadisparos” al que llama el cuerpo principal del juego para hacer pintar los disparos que se indiquen en las filas de los disparos enemigos o en los del jugador. Para llamar a “pintadisparos” debemos hacer un

XOR entre filas de enemigos y jugador.(enemigo1 con jugador3, 2 con 2 y 3 con 1) y el procedimiento pintara un disparo allí donde exista un 1 (también aquí se usan sólo los cuatro bits menos significativos de cada registro).

6.5 - FICHERO NAVE.INC

En este fichero tenemos un procedimiento similar al de enemigos pero que difiere en que el registro que indica donde pintar la nave es “posx” y en que las líneas que conforman la nave están invertidas para generar un gráfico enfrentado a los de “enemigos”.

Tras pintar la nave se recorre el registro “vidas” pintando un cuadro blanco por vida disponible.

7 - DIARIO DE INCIDENCIAS

Las incidencias fueron muy numerosas aunque prácticamente todas debidos a errores humanos. En este apartado cabe destacar algunos detalles principalmente relacionados con la frecuencia de funcionamiento del microcontrolador.

Funcionando a 16Mhz algunas instrucciones producían efectos extraños en la imagen. Estas distorsiones eran producidas porque la señal fluctuaba durante un breve período de tiempo. En general aquellas instrucciones que producían errores eran las que leían un dato y lo modificaban (por ejemplo `addwf color,0`) produciendose riesgos estáticos del tipo WAR (write after read) que generaban los efectos indeseados en la imagen. Este tipo de instrucciones fueron sustituidas por cargas directas al puerto B.

En la especificación se indica que los microcontroladores PIC 16F84 -04P funcionan con una frecuencia máxima de 4 Mhz. El caso es que no estaba del todo claro que esos 4 Mhz fueran la frecuencia máxima de funcionamiento interno o la del externo. La experiencia indica que son capaces de trabajar con un reloj externo de 16Mhz aunque bien es cierto que los errores antes comentados son más frecuentes. Probablemente el proceso de fabricación es el mismo y la catalogación depende de las pruebas que haga el fabricante a cada chip.

El código generado está preparado para funcionar con chips 16F84-04P y 16F84- 20P para evitar la necesidad de adquirir éste último (mucho más caro).