

Módulo de visualización μ OLED-32028-P1T

1.- INTRODUCCION

Ingeniería de Microsistemas Programados S.L. tiene el placer de presentar y distribuir el módulo de visualización gráfico en color μ OLED-32028-P1T (SGC) de la firma 4D SYSTEMS (www.4dsystems.com.au) mostrado en la figura 1.

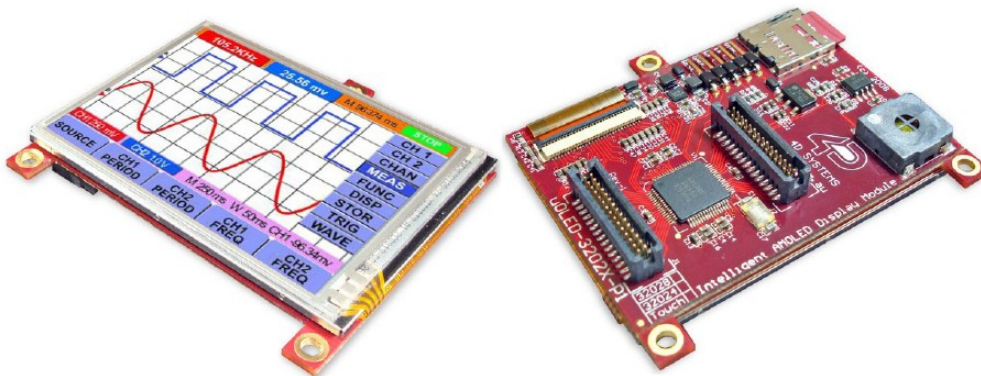


Figura 1. El módulo de visualización μ OLED-3208-P1P

Consiste en un módulo “*todo en uno*” compacto y de coste razonable que combina la más reciente tecnología en pantallas de visualización de Matriz Orgánica Activa OLED (AMOLED) con el controlador gráfico serie PICASO-SGC, para proporcionar una potente herramienta de gran funcionalidad para todo tipo de proyectos y aplicaciones.

Emplea un sencillo interface serie con cualquier tipo de controlador o Host a través del cual se transmiten todo tipo de funciones o comandos de visualización: gestión de gráficos, de texto, de imágenes, de animación, de sonido, gestión de tarjetas de memoria μ SD, entradas/salidas y mucho más.

También puede trabajar de forma totalmente autónoma. En este caso, mediante un PC y potentes herramientas de desarrollo disponibles de forma gratuita, el usuario crea sus propios programas de aplicación o “*scripts*” basados en esa serie de funciones y comandos, y que serán posteriormente ejecutados por el módulo con tan solo conectarle la alimentación.

Finalmente decir que el módulo de visualización μ OLED-3208-P1T incorpora un conector para tarjetas μ SD de memoria de hasta 2Gb en las que podremos almacenar todo tipo de programas o “*scripts*”, imágenes, clips de video, sonido, etc.. y que pueden ser utilizados tanto bajo la conexión con un controlador Host (On line) como cuando se trabaja de forma autónoma (Off line).

2.- CARACTERISTICAS

De entre todas las características que ofrece el módulo de visualización μ OLED-3208-P1T vamos a citar las más relevantes:

- Solución completa de bajo coste para aplicaciones con necesidades de visualización gráfica en color.
- Pantalla AMOLED QVGA de 240 x 320 pixels de resolución con 65K de colores reales.
- Pantalla de 2.83” (49.1 mm x 67.3mm) con una superficie útil de 42.3 mm x 57.6mm
- Integra Touch Screen (pantalla táctil) resistivo de 4 hilos.
- La placa incorpora los soportes necesario para su fijación sobre la aplicación final.
- Angulo de visibilidad de 180º.
- Interface serie mediante 5 conexiones: Vcc, Tx, Rx, GND y RESET

- Comunicación serie asíncrona con niveles TTL (USART) y un rango de velocidad desde los 300 hasta los 256 K baudios.
- Todas las funciones se gestionan con el procesador PICASO-SGC de 4D-Labs.
- Conector para tarjetas de memoria µSD para el almacenamiento de imágenes, música, video clips y programas de aplicación construido a base de sencillas sentencias o comandos (scripts).
- La tarjeta de memoria µSD, con formato FAT16, es compatible con ficheros MS-DOS/Windows
- Salida PWM de audio que soporta ficheros WAV
- Incluye amplificador de audio con un micro altavoz de 8 ohmios para la generación de sonidos y reproducción de ficheros WAV.
- Sencillo conjunto de algoritmos y funciones gráficas de alto nivel que permiten el dibujo de líneas, círculos, texto y mucho más.
- Visualiza imágenes, animaciones, iconos y video clips a pleno color.
- Se pueden importar todos los tipos de caracteres o fonts disponibles en Windows.
- 16 líneas de E/S digitales de propósito general. Ocho de ellas se pueden emplear como un bus paralelo de 8 bits para transferencias rápidas de datos.
- El módulo dispone de dos conectores de 2 x 15 patillas que transportan todas las señales y a través de los cuales se pueden conectar cualquier tipo de tarjeta de expansión y/o de aplicación.
- Tensión de alimentación de 4.5V a 5.5V
- Cumple con la norma RoHS
- En Ingeniería de Microsistemas Programados hemos preparado un CD-ROM donde se incluye abundante información técnica y programas gratuitos para el desarrollo de aplicaciones desde el PC, proporcionados por el fabricante. Igualmente se incluyen una serie de ejemplos realizados por nosotros, basados en PIC y escritos tanto en ensamblador como en lenguaje C

3.- APLICACIONES

Sus aplicaciones y posibilidades son casi infinitas. Detallamos aquí algunas de ellas:

- Mejora de cualquier tipo de aplicación mediante el empleo de gráficos y color gráficas de propósito general
- Sistemas de control de elevadores
- Terminales de punto de venta
- Medidas y calibraciones electrónicas
- Medidas, test e instrumentación de propósito general
- Control industrial y robótica
- Sistemas de visualización en automóviles
- Sistemas de navegación GPS
- Aplicaciones e instrumentación médica
- Aplicaciones y automatizaciones domésticas
- Sistemas de control de seguridad y accesos
- Juegos
- Sistemas de aviación
- Aplicaciones genéricas con interface humano HMI mediante la pantalla y el touch panel

4.- PATILLAJE Y CONEXIONES

Desde un sencillo punto de vista de usuario, el módulo µOLED-3208-P1T se conecta con nuestro controlador Host favorito mediante un interface serie con 5 señales que se resumen en la siguiente tabla.

PIN	NOMBRE	DESCRIPCION
1	VCC	Alimentación de +5Vcc
2	TX	Transmisión de datos serie hacia el Host con niveles TTL
3	RX	Recepción de datos serie desde el Host con niveles TTL
4	GND	Tierra de alimentación
5	RESET	Señal de RESET. Es activa por "0" e inicializa el módulo

Tal y como se muestra en la figura 2 las conexiones con el Host son muy sencillas. Basta con alimentar el módulo y conectar sus señales de transmisión TX y recepción RX con las señales RX y TX respectivamente del controlador. Estas dos líneas son cruzadas. Lo que es transmisión en un extremo es recepción en el opuesto y viceversa.

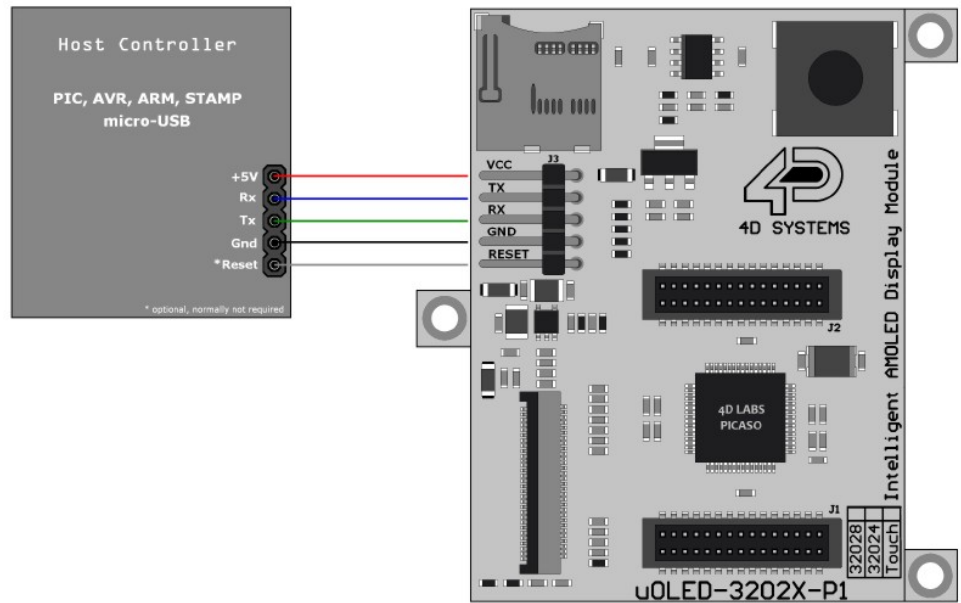


Figura 2.

Interface con el Host

¡¡ Atención !! Estas señales son compatibles con niveles lógicos TTL y por tanto no se pueden conectar directamente con las señales Tx y Rx del canal serie de un PC que son señales con niveles RS232. De no tenerlo en cuenta, se pueden provocar daños irremediables en el módulo.



Figura 3. *El adaptador de USB a serie TTL*

4Dsystems pone en el mercado una serie de adaptadores de USB a serie TTL. En Ingeniería de Microsistemas Programados S.L. hemos trabajado y realizado todas las pruebas con el modelo μ USB-MB5 que se muestra en la figura 3. Los resultados han sido altamente positivos. Por un extremo se conecta directamente con el módulo μ OLED-3208-P1T y por otro con un puerto USB libre del PC mediante un cable (no incluido) con conector mini USB en un extremo.

El empleo de este adaptador no es necesario si TODAS las aplicaciones con el módulo μ OLED-3208-P1T se van a realizar desde un controlador Host (PIC, AVR, BasicSTAMP, ATMET, etc.). Ahora bien, el fabricante proporciona gran cantidad de utilidades y herramientas de desarrollo que se ejecutan desde un PC conectado al módulo μ OLED-3208-P1T. Por ello recomendamos adquirir al menos uno de estos adaptadores.

Toda la información técnica del módulo μ OLED-3208-P1T completa, original y en inglés está disponible en los siguientes documentos PDF (incluidos en el CD-ROM):

- **μ OLED-3202X-P1SGC-DS-rev2**
- **μ OLED-3202X-P1_Users_Manual_Rev1_0**

La versión más actualizada de estos documentos se puede descargar en sus respectivas direcciones del fabricante:

[http://www.4dsystems.com.au/downloads/Serial-Display-Modules/uOLED-3202X-P1\(SGC\)/Docs/uOLED-3202X-P1SGC-DS-rev2.pdf](http://www.4dsystems.com.au/downloads/Serial-Display-Modules/uOLED-3202X-P1(SGC)/Docs/uOLED-3202X-P1SGC-DS-rev2.pdf)

http://www.4dsystems.com.au/downloads/micro-OLED/uOLED-3202X-P1/Docs/uOLED-3202X-P1_Users_Manual_Rev1_0.pdf

5. EL PROCESADOR GRAFICO

El núcleo funcional del módulo de visualización μ OLED-3208-P1T se basa en el procesador gráfico PICASO-SGC diseñado por 4D LABS y cuya imagen se muestra en la figura 4. En el documento **PICASO-SGC-DS-rev2.PDF** (adjunto en el CDROM), el fabricante ofrece toda la información técnica disponible del mismo.



Figura 4. El procesador gráfico PICASO-SGC

La última versión disponible y revisada de este documento se puede descargar desde:

<http://www.4dsystems.com.au/downloads/Semiconductors/PICASO-SGC/Docs/PICASO-SGC-DS-rev2.pdf>

Por un lado, este procesador gráfico conecta con nuestro controlador Host (PIC, AVT, PC, ATmel, STAMP, etc..) mediante un sencillo interface serie. El Host envía una serie de funciones o comandos que el PICASO recibe, interpreta y ejecuta. Estas son las funciones que permiten realizar todo tipo de gráficos, gestión de imágenes, sonido y video, gestión de textos, etc..

Por otro lado el procesador gráfico se conecta con todos los dispositivos y accesorios que se encuentran integrados en el propio módulo de visualización μ OLED-3208-P1T:

- Interface con la pantalla gráfica de matriz activa AMOLED de 2.8"
- Interface con el Touch Panel adherido en la propia pantalla gráfica
- Interface con la tarjeta de memoria μ SD de 2 Gb
- Interface con el amplificador de audio y el altavoz
- Interface con los conectores de expansión y/o aplicación

El firmware interno del procesador gráfico es el encargado de realizar la comunicación con el Host para recibir las funciones o comandos, interpretar los mismos y ejecutarlos. En el documento **PICASO-SGC-COMMANDS-SIS-rev3.PDF** (adjunto en el CDROM) se proporciona una detallada relación y explicación de cada una de estas funciones. La última versión disponible y actualizada de este documento se puede descargar desde:

<http://www.4dsystems.com.au/downloads/Semiconductors/PICASO-SGC/Docs/PICASO-SGC-COMMANDS-SIS-rev3.pdf>

5.1 Actualización del firmware interno

Una de las características más importantes y ventajosas del procesador gráfico PICASO-SGC es la posibilidad de actualizar su firmware interno con la última versión que suministra el fabricante de forma gratuita. Esto supone que, tanto el funcionamiento del procesador como sus prestaciones y los comandos o funciones que integra, pueden ser revisadas, mejoradas y ampliadas.

Esta actualización es recomendable, por no decir necesario, realizarla cuando se adquiere el módulo de visualización y se comienza a trabajar con él por vez primera. Posteriormente se puede realizar tantas veces como sea necesario.

5.1.1 Hardware necesario

Tanto para proceder a la actualización del firmware interno, como para poder hacer uso de las diferentes y potentes herramientas software de diseño que proporciona el fabricante, es necesario disponer del siguiente equipamiento:

- Ordenador PC de sobre mesa o portátil dotado de un puerto USB libre.

- Módulo de visualización μ OLED-3208-P1T que integra el procesador gráfico PICASO-SGC cuyo firmware interno deseamos grabar y/o actualizar con la última versión disponible que facilita el fabricante.
- Adaptador USB a serie TTL para la conexión y comunicación entre el PC y el módulo de visualización. En Ingeniería de Microsistemas sugerimos y distribuimos el modelo μ USB-MB5 con el que hemos obtenido excelentes resultados. Es preciso adquirir por separado el cable USB a mini USB.

5.1.2 Software y ficheros necesarios

Las herramientas software y ficheros necesarios para proceder a la actualización del firmware son los siguientes:

- El PC debe estar dotado de sistema operativo Windows XP o Vista. También debe disponer, si todavía no lo tuviera, el .NET Framework que se puede descargar desde la página oficial de Microsoft.
- Drivers para la instalación del adaptador USB a serie TTL modelos μ USB-MB5. Estos están contenidos en el fichero **CP210x_VCP_Win2K_XP_S2K3.ZIP** que se adjunta en el CDROM y que también se pueden descargar desde: <http://www.4dsystems.com.au/prod.php?id=18>
- Aplicación **PmmC Loader** para Windows. Este programa establece la comunicación entre el PC y el módulo de visualización y permite la grabación del firmware interno sobre el procesador gráfico del módulo. Este software se adjunta en el CDROM y también se puede descargar desde: <http://www.4dsystems.com.au/prod.php?id=46>
- Fichero **uOLED-32028-P1TxSGC-R17.PMMC**. Este fichero contiene el firmware propiamente dicho que se grabará físicamente sobre el procesador gráfico PICASO-SGC. Aunque también se adjunta en el CDROM, recomendamos visitar periódicamente la web del fabricante y descargar la versión más actualizada en: <http://www.4dsystems.com.au/prod.php?id=91>

5.1.3 Pasos a seguir, ¡ manos a la obra !

A continuación vamos a resumir los pasos a seguir para proceder a la grabación/actualización del firmware interno vez primera sobre el procesador gráfico PICASO-SGC.

Esta tarea la realizamos en Ingeniería de Microsistemas Programados S.L. con todos los módulos de visualización. Así, el usuario recibe un módulo grabado, comprobado y listo para conectarlo con su microcontrolador o Host favorito.

1. Descomprimir e instalar el fichero **CP210x_VCP_Win2K_XP_S2K3.ZIP** que contiene los drivers para el adaptador USB a serie TTL. Estos drivers crean sobre el PC un nuevo canal serie virtual COMn.
2. Descomprimir el fichero **PmmCLoader.ZIP** que contiene la aplicación PmmCLoader para Windows. Se obtiene un sencillo y único fichero ejecutable. Esta aplicación es la que permite comunicar el módulo de visualización con el PC y grabar sobre el procesador PICASO-SGC el fichero que contiene el firmware interno.
3. Ejecutar la aplicación PmmC Loader. Aparece una ventana como la mostrada en la figura 5.

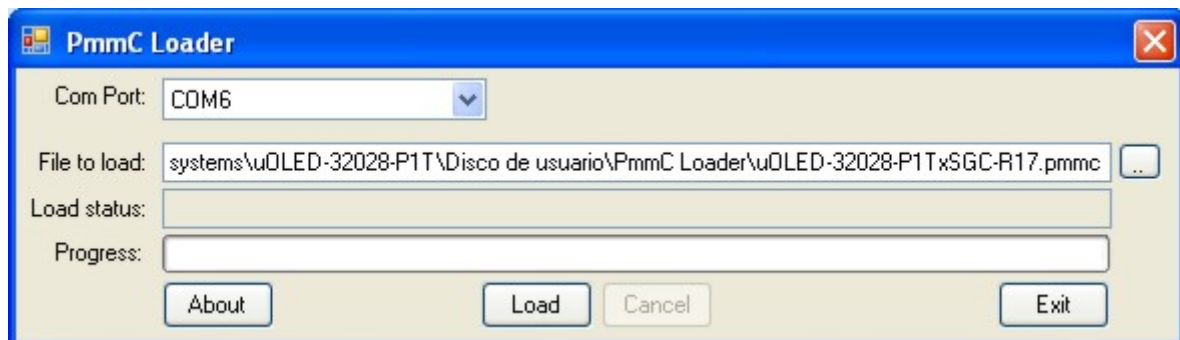


Figura 5. El programa PmmC Loader

Debemos elegir el puerto COM seleccionando el puerto COMn que se creó al instalar los drivers del adaptador USB a serie TTL (COM6 en la figura). También debemos seleccionar la carpeta donde se encuentra el fichero **uOLED-32028-P1TxSGC-R17.PMMC** con el firmware a grabar. Finalmente pulsamos el botón **Load** para proceder a la grabación del fichero sobre el procesador gráfico.

Realizadas estas operaciones, el procesador gráfico y por extensión, el módulo de visualización, ya están listos para conectarse y realizar la comunicación serie con nuestro Host o controlador preferido. A partir de este momento podemos enviar y hacer uso de todas las funciones y comandos disponibles (consultar el correspondiente documento). El módulo de visualización μ OLED-3208-P1T se comporta ahora como un periférico inteligente capaz de realizar cualquier tipo de aplicación gráfica.

6. HERRAMIENTAS DE DESARROLLO

El módulo μ OLED-3208-P1T puede trabajar conectado con un controlador Host que le transmite vía serie las diferentes funciones y comandos disponibles, gracias a las cuales el usuario podrá desarrollar potentes proyectos y aplicaciones que incluyen gráficos, imágenes, sonidos, videos, etc., mejorando enormemente el interface humano de cualquier diseño.

Por otra parte no debemos olvidar que el mismo módulo de visualización es un dispositivo totalmente autónomo. Efectivamente, en si mismo se trata de todo un sistema completo que dispone de su propio procesador con su memoria interna, externa a través de tarjetas μ SD, pantalla gráfica en color, touch panel, sonido y líneas de E/S. Es decir, podemos diseñar aplicaciones totalmente independientes de ningún Host e instalar el propio módulo en el hardware final.

Por ello, el fabricante pone a disposición del usuario y, de forma gratuita, un buen número de herramientas de desarrollo que permiten experimentar con el módulo de visualización así como desarrollar aplicaciones con el mismo.

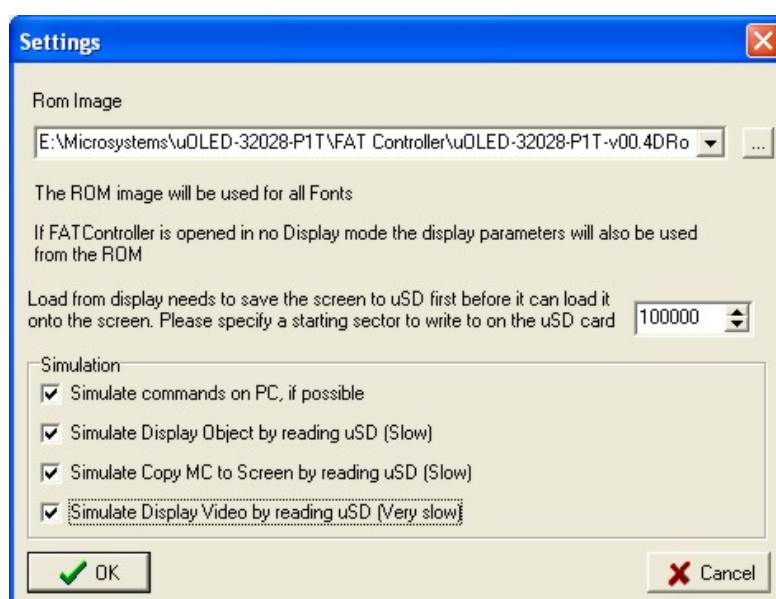
6.1 FAT Controller

Esta herramienta de desarrollo es, a nuestro juicio, la más versátil e interesante. En un cómodo y clásico entorno Windows disponemos de los botones y recursos necesarios para transmitir y ejecutar cualquiera de las funciones y comandos disponibles en el firmware interno del procesador gráfico. Todo ello desde el PC como si de un Host se tratara. Esto permite experimentar con todas las funciones así como crear nuestros propios programas autónomos que luego se ejecutarán de forma independiente.

El PC se conecta con el módulo mediante el adaptador USB a serie TTL cuyos drivers se suponen instalados como ya lo hicimos en el apartado 5.1.3. En el CDROM se incluye el fichero **FAT-Controller.ZIP** que, tras descomprimirlo se obtiene la herramienta en cuestión. La última versión disponible, así como la guía del usuario, se puede descargar desde: <http://www.4dsystems.com.au/prod.php?id=70>

Una vez descomprimido y ejecutado el programa de aplicación FAT-Controller dispondremos del correspondiente área de trabajo. En un primer momento seleccionamos el botón **Settings** que nos presenta una cuadro de diálogo como el mostrado en la figura 6.

Figura 6. Ajustes del programa FAT-Controller



Estos ajustes nos permiten determinar la carpeta en la que se encuentra el fichero **uOLED-32028-P1T-v00.4DRom** que contiene información útil para la simulación. Se adjunta en el CDROM. También permite activar una serie de opciones de simulación:

- Simular, si es posible, los comandos que se envían al módulo de visualización sobre el propio PC.
- Simular sobre el PC la visualización de objetos mediante su lectura desde la tarjeta de memoria μ SD.
- Simular en el PC una copia de la pantalla mediante su lectura desde la tarjeta μ SD.
- Simular en el PC la visualización de un video mediante la lectura del mismo desde la tarjeta μ SD.

Finalizamos estos ajustes mediante el botón **OK** que nos devuelve a la pantalla principal. A partir de este momento se selecciona el canal serie que se va a emplear en la comunicación entre el PC y el módulo de visualización (COM6 en el ejemplo) y pulsamos el botón **Open**.

Comienza la ejecución del programa FAT-Controller. A partir de este momento el PC actúa como un Host y se encargará de enviar al módulo de visualización μ OLED-3208-P1T todos los comandos o funciones que su firmware interno es capaz de interpretar y ejecutar, con la importante ventaja de que el PC puede simular la mayor parte de esas funciones sobre su propia pantalla. Esto nos permite analizar y estudiar dichas funciones, observar qué efectos produce, qué códigos genera cada función y mucho más. La figura 7 presente el área de trabajo del programa FAT-Controller.

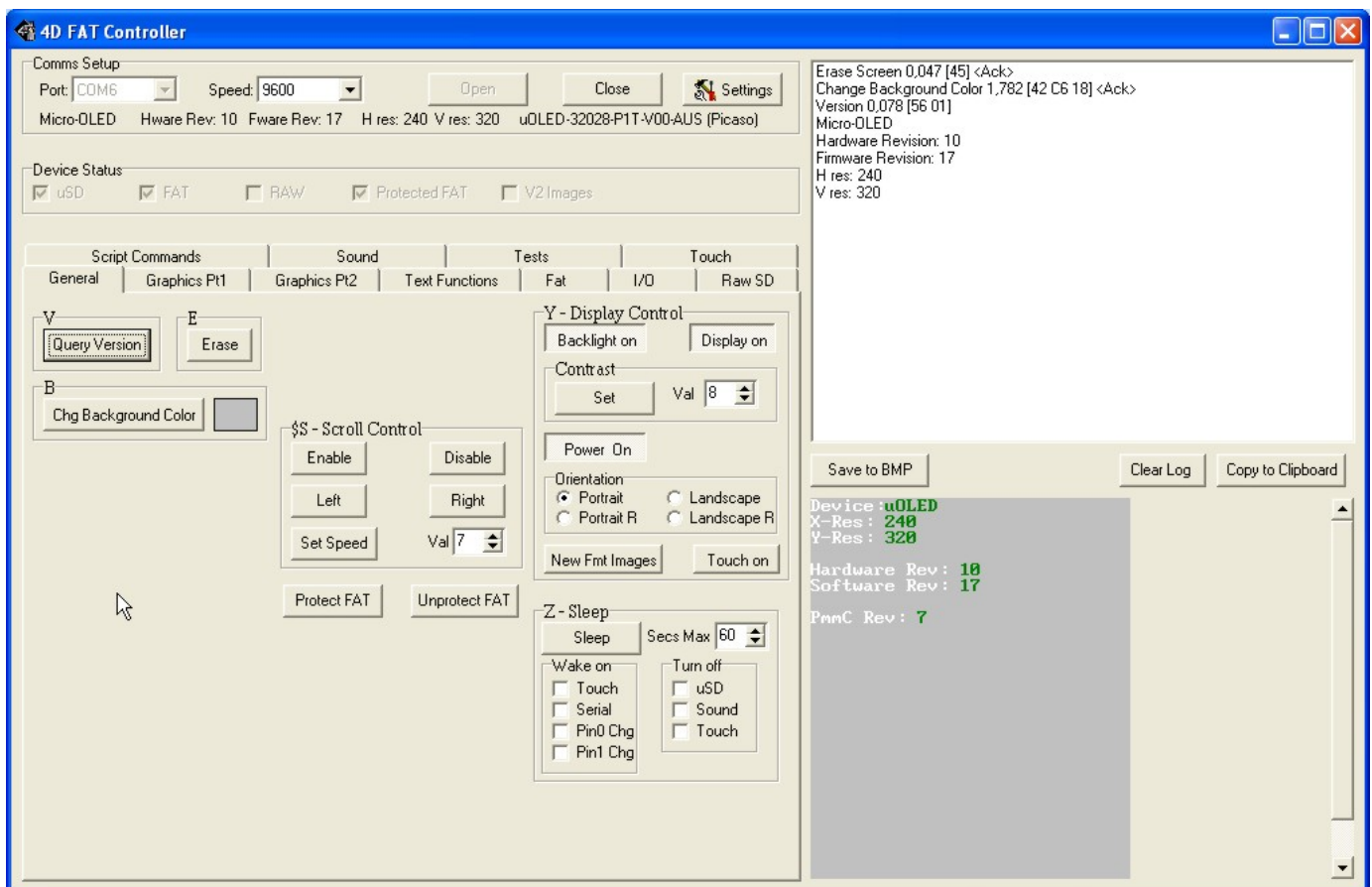


Figura 7. El área de trabajo de FAT-Controller

Todas las funciones disponibles se seleccionan y envían al módulo de visualización de una forma sencilla y clara mediante todo tipo de botones y selecciones. A la derecha del área de trabajo, en la parte de arriba, aparece una ventana de texto donde se irán registrando, a modo de data logger, todos los comandos y funciones que se van enviando. Cabe destacar que en ella podemos ver los códigos hexadecimales de cada función enviada y ejecutada.

Estos códigos van cerrados entre corchetes y son los mismos que debiéramos mandar desde cualquier otro controlador Host (PIC, AVT, ATmel, STAMP, etc..) si estuviéramos realizando una aplicación real con ellos.

En la parte inferior derecha del área de trabajo muestra una representación de la propia pantalla gráfica. Sobre ella se va simulando el resultado de la ejecución de prácticamente todas las funciones y comandos. Herramienta muy útil que nos permite apreciar sobre el PC lo que luego ocurrirá sobre la pantalla real.

A la izquierda del área de trabajo nos encontramos con todas los comandos y funciones disponibles así como sus correspondientes ajustes y parámetros. Están clasificados por fichas y se corresponden con los comandos descritos en el documento **PICASO-SGC-COMMANDS-SIS-rev3.PDF** (adjunto en el CDROM).

6.1.1 Ficha General

Es la que está seleccionada en la figura 7 anterior. Contiene una serie de comandos básicos: obtener la versión del firmware, borrar pantalla, orientación de la misma, color de fondo, control del display, etc.. Si observamos el área correspondiente al data logger de la figura 7, veremos que en el ejemplo se han enviado, ejecutado y simulado tres comandos:

- Erase Screen 0,047 [45] <Ack>: Borrar pantalla, código 0x45
- Change Background Color 1,782 [42 C6 18] <Ack>: Cambiar color de fondo a gris, código 0x42, 0xC6, 0x18
- Version 0,078 [56 01]: Obtener la versión del firmware interno, código 0x56, 0x01

Se puede apreciar que cada comando o función tiene su correspondiente código hexadecimal. Se muestra cerrado entre corchetes. Cada vez que se ejecuta un comando el módulo de visualización devuelve ACK o NACK según se ejecute correctamente o no. También podemos observar el resultado de la ejecución de esos comandos sobre la propia pantalla real del módulo de visualización, si estuviera conectada al PC, o sobre la pantalla simulada del PC.

6.1.2 Ficha Graphics Pt1

Esta ficha contiene funciones gráficas como el dibujo de triángulos, círculos, líneas, imágenes desde un archivo en el PC, generación y visualización de caracteres gráficos.

Cada uno de estos comandos necesita de una serie de parámetros como son las coordenadas iniciales para su dibujo, coordenadas finales, el color del mismo, etc.. Todos estos parámetros se determinan en sus correspondientes campos.

Cuando se pulsa el botón del comando correspondiente, el PC transmite al módulo de visualización una trama completa de bytes en hexadecimal que representan el comando propiamente dicho y sus correspondientes parámetros. Esta trama recibe el nombre de **script**. El módulo ejecuta entonces el comando y retransmite ACK si su ejecución es correcta. De la misma manera sobre la pantalla simulada del PC se puede apreciar el mismo efecto de ejecución.

Se sugiere experimentar con todos ellos para ver los resultados obtenidos así como analizar la ventana logger con los códigos hexadecimales (**scripts**) que se generan. Estos **scripts** son los mismos que debiéramos transmitir, por ejemplo desde un PIC actuando como Host.

6.1.3 Ficha Graphics Pt2

Esta ficha contiene otro grupo de comandos o funciones relacionados con el dibujo de gráficos: dibujar un pixel, leer un pixel, dibujar un polígono, una elipse o un rectángulo, seleccionar el tipo de lápiz, etc..

Cada uno de estos comandos necesita de una serie de parámetros como son las coordenadas iniciales para su dibujo, coordenadas finales, el color del mismo, etc.. Todos estos parámetros se determinan en sus correspondientes campos.

Cuando se pulsa el botón del comando correspondiente, el PC transmite al módulo de visualización una trama completa de bytes en hexadecimal que representan el comando propiamente dicho y sus correspondientes parámetros. Esta trama recibe el nombre de **script**. El módulo ejecuta entonces el comando y retransmite ACK si su ejecución es correcta. De la misma manera sobre la pantalla simulada del PC se puede apreciar el mismo efecto de ejecución.

Se sugiere experimentar con todos ellos para ver los resultados obtenidos así como analizar la ventana logger con los códigos hexadecimales (**scripts**) que se generan. Estos **scripts** son los mismos que debiéramos transmitir, por ejemplo desde un PIC actuando como Host.

6.1.4 Ficha Text Functions

Esta ficha contiene todo tipo de comandos y funciones relacionadas con la visualización de texto: selección del tipo de carácter, visualizar caracteres y/o cadenas de texto con formato, sin formato, botones de texto pulsados, sin pulsar, colores, etc...

6.1.5 Ficha FAT

Esta ficha está relacionada con funciones y comandos dedicados a la gestión de la memoria μ SD de hasta 2Gb, externa y opcional que podemos emplear junto con el módulo de visualización μ OLED-3208-P1T.

La gestión se realiza a nivel de archivos completos con formato FAT16 estándar. Estas funciones nos permiten iniciar la propia tarjeta de memoria, listado de directorio de archivos contenidos en ella, borrar archivos, leer archivos, escribir archivos, ejecutar archivos que contienen scripts con otras funciones, copiar el contenido actual de la pantalla sobre un archivo, leer archivos con imágenes, videos, sonido, etc..

En resumidas cuentas esta ficha contiene una serie de funciones potentes y sencillas que permiten el uso de una tarjeta μ SD.

6.1.6 Ficha I/O

Las funciones contenidas en esta ficha permiten el manejo de las líneas de E/S de propósito general que contiene el generador gráfico PICASO-SGC para el control de periféricos externos.

Entre esas funciones caben destacar la lectura o escritura de datos sobre un bus formado por 8 de esas líneas de E/S así como la lectura/escritura de líneas individuales de E/S.

6.1.7 Ficha Raw SD

Las funciones aquí contenidas permite el acceso a bajo nivel de la tarjeta μ SD externa de memoria. Con ellas se puede acceder a la tarjeta a nivel de bytes y sectores para escribir o leer datos.

Son tareas muy delicadas ya que escribir un simple byte sobre un sector inadecuado puede suponer la destrucción de ficheros completos o incluso exigir un nuevo formateo de la tarjeta, con lo que supone el borrado total de la misma.

Recomendamos el empleo de las funciones de la ficha FAT anteriormente comentada. En caso de necesitar emplear los comandos de ésta ficha, sugerimos encarecidamente la lectura de la información técnica de las mismas.

6.1.8 Ficha Touch

Contiene una serie de funciones relacionadas con el panel táctil (Touch panel) con el que está dotado el módulo de visualización μ OLED-3208-P1T: tiempos de espera, establecer la región o área activa, esperar a que se detecte un toque, un desplazamiento, una ausencia de toque, buscar coordenadas, etc..

6.1.9 Ficha Tests

Contiene una serie de tests sobre la tarjeta de memoria externa µSD. Precaución con su empleo porque puede suponer la pérdida de información de la misma.

6.1.10 Ficha Sound

Contiene tareas propias para la reproducción de sonidos almacenados previamente en la tarjeta de memoria externa µSD: reproducir ficheros WAV con diferentes opciones, aumentar/disminuir el volumen, modo de silencio o no, etc...

6.1.12 Ficha Script Commands

Como ya se ha comentado, cualquier comando o función con sus correspondientes parámetros forman un script. Un conjunto de scripts forman lo que podríamos llamar un programa en el que esos scripts se ejecutarán secuencialmente.

En la pantalla logger vamos teniendo una muestra de todas las funciones o comandos que se han ido enviando y ejecutando. En esta ficha tenemos funciones que nos permiten recoger esos comandos a partir de un momento dado (**Clear Script**) para luego salvarlos en nuestro PC mediante **Save Script**, en un fichero tipo 4DSCRPOBJ.

Mediante la función **Load** podemos leer ese fichero para su ejecución (**Run**), ejecución script a script (**Step**) o bien pausar la ejecución (**Pause**).

También podemos hacer uso de la función **Write File** (de la ficha FAT) para coger ese fichero de scripts (4DSCRPOBJ) y salvarlo sobre la tarjeta µSD con formato de fichero ejecutable (*.4DS).

Mediante la función **Execute Program** podemos ejecutar cualquier fichero tipo 4DS almacenado en la tarjeta µSD y con un contenido basado en scripts.

Hay que indicar que si a un fichero de scripts almacenado en la tarjeta µSD se le asigna el nombre de AUTOEXEC.4DS, se ejecutará de forma automática nada más conectar la alimentación del sistema.

Por último indicar que en esta ficha también hay una serie de funciones relacionadas con el control de flujo en la ejecución de un fichero de scripts: temporizar entre un script y el siguiente, salto incondicional a un script, salto condicional y final de un programa de scripts.

6.2 Graphics Composer

Se trata de otra potente herramienta para el diseño de aplicaciones basadas en el módulo de visualización µOLED-3208-P1T y su procesador gráfico PICASO-SGC. Tanto el programa como su correspondiente manual de usuario se facilita en el CDROM. La última versión disponible se puede descargar en <http://www.4dsystems.com.au/prod.php?id=91>

Se trata de herramienta gratuita para Windows que permite hacer una composición con ficheros de imágenes y videos desde el PC y con diferentes formatos. Como resultado de esa composición se obtiene un fichero que se graba sobre la tarjeta de memoria µSD y con un formato legible por el procesador gráfico PICASO-SGC.

Una vez se descomprime e instala el programa, procedemos a ejecutarlo siguiendo las instrucciones del manual. Aparece un área de trabajo como el mostrado en la figura 8.

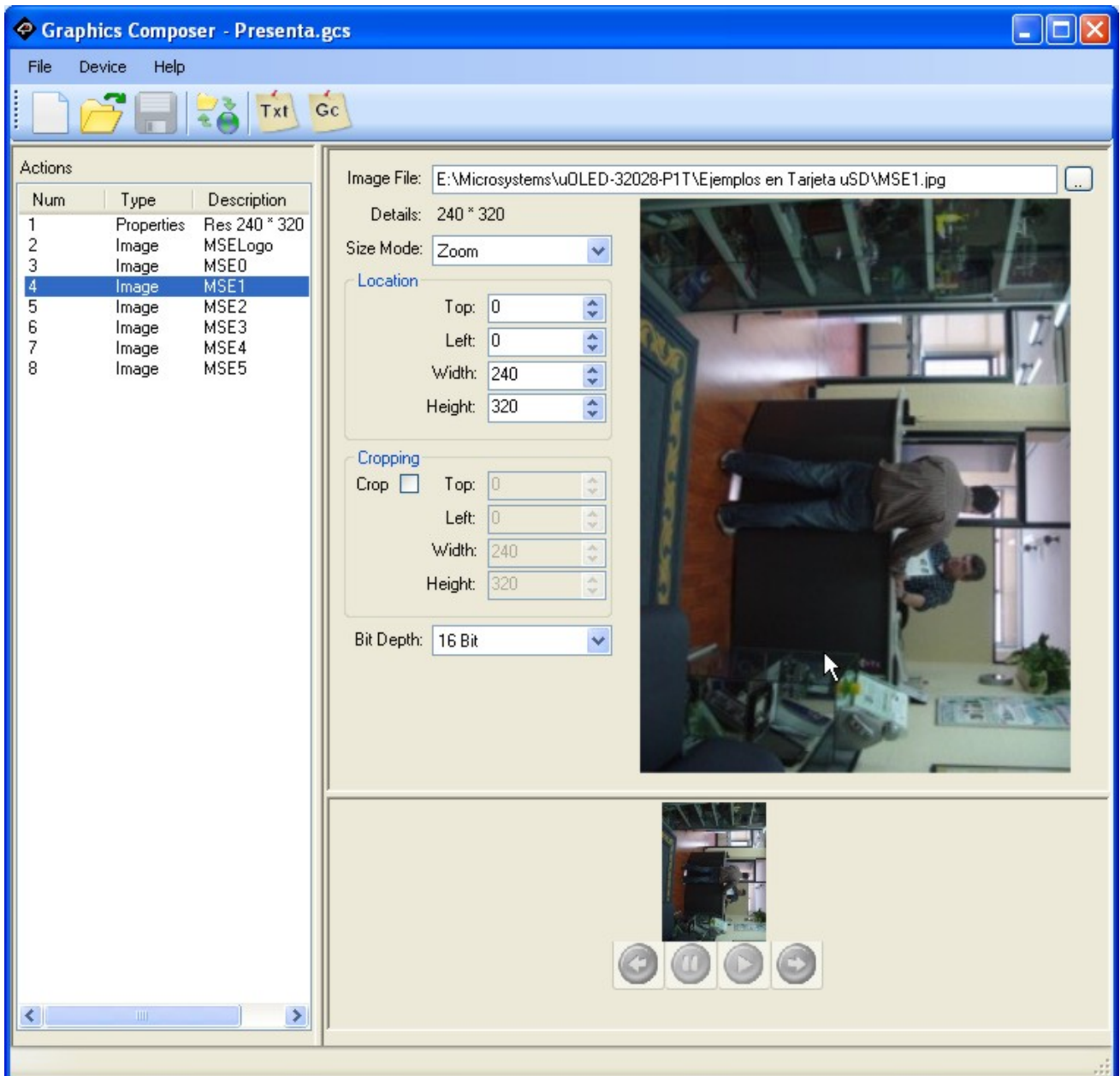


Figura 8. El programa Graphics Composer

Según la figura anterior, hemos creado un proyecto llamado "Presenta.GCS" que se salva sobre el PC. Este proyecto incluye la definición de las propiedades de la pantalla y una serie de imágenes que se pueden encontrar en el propio PC o en la tarjeta de memoria μ SD y que se han ido añadiendo en la columna de la izquierda **Actions**. De la misma manera se hubieran podido añadir ficheros o clips de video. A cada imagen añadida se le asocia una serie de atributos como son el tamaño y la posición que ocuparán dentro de la pantalla.

Cuando ya tenemos el proyecto completo y salvado, basta con pulsar el botón **Load Devide**. En ese momento comienza a generarse un fichero que contiene la compilación de esas imágenes y que se almacenará en la tarjeta μ SD. Este fichero tiene la extensión GCI (PRESENTA.GCI en el ejemplo) y puede ser leído e interpretado por el módulo de visualización mediante el comando o función apropiada. También se genera otro fichero de tipo texto y con extensión DAT (PRESENTA.DAT en el ejemplo). Este fichero puede **y debe** imprimirse y nos indica el sector inicial de cada imagen dentro del fichero GCI en la tarjeta de memoria μ SD. Según el ejemplo su contenido será algo parecido a esto:

```
"MSELogo.jpg" 0000 0000  
"MSE0.jpg" 4400 0000  
"MSE1.jpg" 9E00 0002  
"MSE2.jpg" F800 0004  
"MSE3.jpg" 5200 0007  
"MSE4.jpg" AC00 0009  
"MSE5.jpg" 0600 000C
```

Se puede traducir algo así como que la imagen MSELogo.jpg comienza en el sector 0-0 del fichero PRESENTA.GCI. La imagen MSE0.jpg comienza en el sector 0x0000 (MSB) - 0x4400 (LSB) que en decimal se expresaría como 0 - 17408. La imagen MSE1.jpg comienza en el sector 0x0002 (MSB) – 0x9E00 (LSB) que en decimal se expresa como 0 – 40448... y así sucesivamente.

Esta información nos servirá posteriormente para visualizar una imagen en particular dentro de un fichero GCI obtenido mediante el programa Graphics Composer. Efectivamente, tal y como se muestra en la figura 9, haciendo uso del programa FAT-Controller, en la ficha FAT y mediante el comando o función Display Image/Icon Fat, se visualiza la imagen que empieza en el sector 2 (MSB) – 40448 (LSB) del fichero PRESENTA.GCI de la tarjeta de memoria µSD (MSE1.jpg).

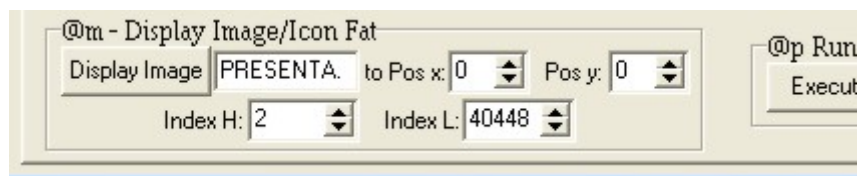


Figura 9. Visualizando una imagen del fichero PRESENTA.GCI



Si consultamos el documento **PICASO-SGC-COMMANDS-SIS-rev3.PDF** (adjunto en el CDROM), la trama o el script de bytes en hexadecimal que un Host (PIC, ATmel, Stamp, etc..) debiera enviar al módulo de visualización para ejecutar este comando o función, será algo así como esto:

```
40 6D 50 52 45 53 45 4E 54 41 2E 47 43 49 00 00 00 00 00 00  
02 9E 00
```

El resultado de la ejecución visualiza sobre la pantalla la imagen MSE1.jpg contenida en el fichero PRESENTA.GCI que se obtuvo mediante la aplicación Graphics Composer y se guardó en la tarjeta de memoria µSD. Ver la figura 10.

Figura 10. Visualización de una imagen

6.3 Font Tool

Se trata de otro programa gratuito para plataformas basadas en Windows. También se adjunta en el CDROM junto con guía de usuario y la última versión disponible se puede descargar desde <http://www.4dsystems.com.au/prod.php?id=80>

La aplicación es un asistente para la conversión de cualquiera de los fonts disponibles en Windows en el formato Bitmap empleado por el display. Al seleccionar uno de esos fonts tendremos una representación de los caracteres del mismo.

Cuando se selecciona un carácter en particular, tenemos la posibilidad de modificarlo pixel a pixel, cambiarle el tamaño y/o desplazarle horizontal y/o verticalmente.

Realizados, si se precisa, los cambios oportunos sobre los caracteres de un determinado Font, se puede proceder a exportarlos como Bitmaps para que puedan ser utilizados en nuestros proyectos y aplicaciones.

6.4 4DGL Workshop3

Otra aplicación gratuita que se adjunta, junto con su guía de usuario, en el CDROM. La última versión disponible se puede descargar en <http://www.4dsystems.com.au/developers/>

Consiste en un entorno integrado de desarrollo (IDE) de software para todos los procesadores gráficos de la familia 4D y sus correspondientes pantallas. El entorno incluye un editor de textos, un compilador, un linker y un sistema downloader para grabar el programa ejecutable obtenido, sobre el módulo de visualización.

Se trata pues de una herramienta para el desarrollo de aplicaciones en la que se emplea un lenguaje de alto nivel similar al C o al Basic, para generar el programa ejecutable final.

Para hacer uso de la misma es necesario readaptar el firmware interno del procesador gráfico PICASO-SGC. Para ello tenemos que hacer uso de la aplicación PmmC Loader como se hizo en la sección 5.1. En esta ocasión debemos grabar el fichero **uOLED-32028-P1T-4DGL-rev1_08.PMMC**. Una vez realiza esta operación dispondremos de un procesador gráfico adaptado a las necesidades y funciones de el entorno integrado 4DGL Workshop3.

Estamos hablando ahora de desarrollar aplicaciones autónomas e independientes de un Host. El propio módulo de visualización es en sí mismo un sistema completo con su memoria, pantalla, touch panel, líneas de E/S, etc... que se instala en el hardware final del proyecto en cuestión.

Empleamos para ello un lenguaje de alto nivel muy potente y versátil con sentencias y funciones similares a las de otros lenguajes de alto nivel como el C, el Basic, el Pascal, etc.. El fabricante 4D Systems pone a disposición del interesado toda una página dedicada al entorno 4DGL (enlace indicado anteriormente). En ella se puede proceder a la descarga gratuita del mismo así como a la explicación de todas y cada una de las sentencias disponibles en el lenguaje.

7.- LIBRERIAS

En el CDROM, junto con los programas fuente de los ejemplos propiamente dichos, se proporcionan una serie de librerías escritas tanto en ensamblador (*.INC) como en lenguaje C de alto nivel (*.H). A su vez cada librería incluye una serie de rutinas o funciones que facilitan la realización de diferentes tareas. Estas librerías se incluyen en los programas fuente mediante la directiva `<include>`.

7.1 Librería: *MSE_Mat_PIC16.inc*

Contiene una serie de funciones que resuelven las operaciones matemáticas más elementales y que son empleadas por algunos de los ejemplos escritos en ensamblador. Los ejemplos escritos en C no necesitan de esta librería ya que la mayor parte de las funciones matemáticas están integradas en el propio compilador.

NOMBRE	PARAM. DE ENTRADA	PARAM. DE SALIDA	DESCRIPCION
Sum_BCD	Mat_Dato_AL+Mat_Dato_BL	Mat_Dato_BL Mat_Dato_AH	Suma dos números BCD (00-99): Mat_Dato_AL + Mat_Dato_BL. El resultado se almacena en Mat_Dato_BL. En Mat_Dato_AH queda, si lo hubiera, el overflow (resultado mayor de 99)
Sub_BCD	Mat_Dato_AL- Mat_Dato_BL		Resta dos números BCD (00-99): Mat_Dato_AL - Mat_Dato_BL. El resultado se almacena en Mat_Dato_AL. En Mat_Dato_BH queda, si lo hubiera, el overflow (A<B).
Sum16	Mat_Dato_BH:Mat_Dato_BL+ Mat_Dato_AH:Mat_Dato_AL	Mat_Dato_BH Mat_Dato_BL C	Suma dos números de 16 bits. Mat_Dato_BH:Mat_Dato_BL+ Mat_Dato_AH:Mat_Dato_AL. El resultado se deposita en

			Mat_Dato_BH:Mat_Dato_BL. Si hay llevada en el bit 16º, C=1
Sub16	Mat_Dato_AH:Mat_Dato_AL- Mat_Dato_BH:Mat_Dato_BL.	Mat_Dato_BH Mat_Dato_BL C	Resta dos números de 16 bits. Mat_Dato_AH:Mat_Dato_AL- Mat_Dato_BH:Mat_Dato_BL. El resultado se almacena en Mat_Dato_BH:Mat_Dato_BL. Si hay llevada en el bit 16º, C=0
Comp16	Mat_Dato_AH:Mat_Dato_AL(A) Mat_Dato_BH:Mat_Dato_BL(B)	C, Z	Compara dos nº de 16 bits contenidos en Mat_Dato_AH:Mat_Dato_AL(A) con Mat_Dato_BH:Mat_Dato_BL(B) Si A>B --> STATUS<C>=1 y <Z>=0; Si A<B --> STATUS<C>=0 y <Z>=0; Si A=B --> STATUS<C>=1 y <Z>=1
Mul8x8	Mat_Dato_AL * Mat_Dato_BL.	Mat_RES2:Mat_RES3	Multiplica dos nºs de 8 bits contenidos en Mat_Dato_AL y Mat_Dato_BL. El resultado de 16 bits se almacena en Mat_RES2 y Mat_RES3 (LSB).
Mul16x16	Mat_Dato_BH:Mat_Dato_BL* Mat_Dato_AH:Mat_Dato_AL	Mat_RES0:Mat_RES3	Multiplica dos números de 16 bits: Mat_Dato_BH:Mat_Dato_BL* Mat_Dato_AH:Mat_Dato_AL El resultado de 32 bits se almacena en Mat_RES0:Mat_RES3 (LSB)
Div16x16	Mat_Dato_AH:Mat_Dato_AL/ Mat_Dato_BH:Mat_Dato_BL	Mat_Dato_AH:Mat_Dato_AL Mat_RES3:Mat_RES2	Esta rutina divide dos números de 16 bits. El dividendo se almacena en Mat_Dato_AH:Mat_Dato_AL y el divisor en Mat_Dato_BH:Mat_Dato_BL. El cociente se almacena en Mat_Dato_AH:Mat_Dato_AL y el resto en Mat_RES3:Mat_RES2.
Bits8_BCD	W	Mat_RES0	Convierte un número binario de 8 bits en el registro W, en 2 dígitos BCD (de 00 a 99). El resultado se almacena en Mat_RES0. P.e. W=0x2A, Mat_RES0=42 (0x2A = 42).
Bits16_BCD	Mat_Dato_AH:Mat_Dato_AL	Mat_RES0 Mat_RES1 Mat_RES2 (LSB)	Esta rutina convierte un número binario de 16 bits situado en Mat_Dato_AH y Mat_Dato_AL y, lo convierte en 5 dígitos BCD que se depositan en las variables Mat_RES0, Mat_RES1 y Mat_RES2, siendo esta última la de menos peso.
BCD_Bits16	Mat_RES0:Mat_RES2 (LSB)	Mat_Dato_AH:Mat_Dato_AL	Convierte un número de 5 dígitos en BCD, en un número de 16 bits. En Mat_RES0:Mat_RES2 (LSB) se encuentra en nº de 5 dígitos. El resultado se almacena en Mat_Dato_AH:Mat_Dato_AL.
BCD_ASCII	Mat_RES0:Mat_RES2 (LSB) Mat_Point	Mat_RES0(MSB):Mat_RES6(LSB)	Convierte un número BCD de hasta 6 dígitos almacenados en Mat_RES0:Mat_RES2 (LSB) En una cadena ASCII que se almacena a partir de Mat_RES0(MSB) hasta Mat_RES6(LSB). La variable Mat_Point indica el lugar donde colocar un punto decimal (0 =ninguno). P.e. Mat_Point=1, el resultado sería .xxxxxx; Mat_Point=3, el resultado sería xx.xxxx

8.- EJEMPLOS

Junto con el módulo de visualización se proporciona un CDROM con una serie de ejemplos didácticos desarrollados por Ingeniería de Microsistemas Programados al objeto de facilitar el uso del módulo. Se facilitan las librerías y programas fuente escritos tanto en ensamblador como en lenguaje C de alto nivel para PIC16F886.

Los distintos ejemplos van haciendo uso de algunas de las funciones o comandos que es capaz de recibir, interpretar y ejecutar el módulo de visualización µOLED-3208-P1T y el procesador gráfico PICASO-GCS que integra.

En todos esos ejemplos hemos seguido un mismo tratamiento para los comandos y funciones que se envían al módulo de visualización. Por supuesto que nuestra fórmula no es la única ni la mejor. Simplemente hemos tratado de presentarlos de la forma más sencilla y didáctica que se nos ha ocurrido. El usuario tiene absoluta libertad de realizar los tratamientos como mejor se adapten a sus necesidades.

Todos los comandos que se envían al módulo de visualización los tratamos como si de una cadena de bytes fijos y almacenados en la memoria Flash de programa se tratara, **una cadena un comando**. Cada cadena está compuesta del código del comando propiamente dicho y de los parámetros que necesita el mismo.

Por ejemplo, el código del comando para seleccionar cambio de color de fondo es 0x42 seguido de dos bytes 0xCH y 0xCL que definen el color. Además, cada cadena de cada comando va precedida de dos bytes de control. Estos los hemos establecido nosotros para su empleo con nuestras rutinas y no son propios del módulo de visualización. El primero de esos bytes determina cuántos bytes ocupa el comando propiamente dicho y el segundo determina cuántos bytes devuelve el módulo de visualización cada vez que ejecuta ese comando. Si este segundo byte es 0x00 es porque el módulo sólo devolverá el código ACK/NACK. Si vale 0xff es porque el módulo no devuelve nada. Así pues, el comando para cambiar el color de fondo de la pantalla estaría formado por una cadena similar a esta:

0x03 0x00 0x42 0xCH 0xCL

Esto se puede traducir como que el comando para ajustar el color de fondo necesita 3 bytes (0x42, 0xCH y 0xCL) y el módulo sólo devuelve el código ACK (0x06) o NACK (0x15).

Consultar el documento *PICASO-SGC-COMMANDS-SIS-rev3* que se adjunta en el CDROM donde se explican todos los comandos disponibles.

Todos los comandos se gestionan desde una rutina o función fundamental llamada “**Send_Com**”. Esta rutina lee la cadena de bytes del comando que se desea y los transmite al módulo de visualización. A continuación queda a la espera de recibir los bytes de respuesta que se almacenarán en un buffer en RAM que empieza a partir de **uOLED_Buffer**.

8.1 Ejemplo 1: Funciones básicas

Objetivos

Con este ejemplo inicial se pretende familiarizar al usuario con el tratamiento que nosotros damos a las cadenas que forman los distintos comandos a ejecutar por el módulo de visualización. En este caso vamos a emplear unos pocos y básicos comandos: borrado de pantalla, obtención y visualización de la versión del firmware interno y cambio de color del fondo de la pantalla.

Esquema

Se muestra en la figura 11. Consiste en las conexiones a realizar entre el controlador Host PIC16F886 de nuestro laboratorio USB-PIC'School y el módulo de visualización. Estas mismas conexiones y quizá, alguna más, se emplearán para todos los demás ejemplos propuestos.

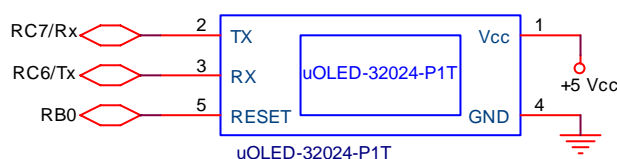


Figura 11. Esquema del ejemplo 11

Comentarios

En este primer ejemplo queremos resaltar la forma en que hemos codificado los diferentes comandos que se van a transmitir al módulo de visualización. Cada comando se almacena en la memoria flash de programa en forma de una cadena de bytes como se muestra en la siguiente tabla. Posteriormente, en función del comando que se desea ejecutar, una única rutina llamada **Send_Com** se encargará de ir extrayendo cada byte de la cadena para transmitírselo al módulo de visualización.

Hemos empleado esta técnica porque nos parece bastante sencilla y legible. Sin embargo presenta un inconveniente. Las cadenas se almacena en la flash de programa y por tanto no se pueden modificar. Así pues tenemos definidos unos comandos fijos cuyos parámetros son siempre iguales.

Tabla_Com	clrf	PCLATH	
Version	movwf	PCL	;Calcula el desplazamiento sobre la tabla
	equ	\$;Comando Versión del firmware interno
	dt	0x02,0x05,0x56,0x01	
Erase	equ	\$;Comando borra pantalla
	dt	0x01,0x00,0x45	
Fondo_gris	equ	\$;Selecciona fondo gris
	dt	0x03,0x00,0x42,0xc6,0x18	
Fondo_rojo	equ	\$;Selecciona fondo rojo
	dt	0x03,0x00,0x42,0xf8,0x00	
Fondo_azul	equ	\$;Selecciona fondo azul
	dt	0x03,0x00,0x42,0x04,0x1f	
Fondo_negro	equ	\$;Selecciona fondo negro
	dt	0x03,0x00,0x42,0x00,0x00	

Otra técnica más efectiva pero más compleja y laboriosa consistiría en crear rutinas o funciones específicas para cada comando y donde sus parámetros se establecieran en memoria RAM y fueran por tanto variables. De esta forma una misma función puede obtener resultados distintos alterando esos parámetros en el curso de la ejecución del programa de aplicación.

En la figura 12 se aprecia el resultado de la ejecución de este ejemplo que se monta y prueba sobre nuestro laboratorio USB-PIC'School.



Figura 12. Ejecución del ejemplo 1

8.2 Ejemplo 2: Funciones de texto sin formato

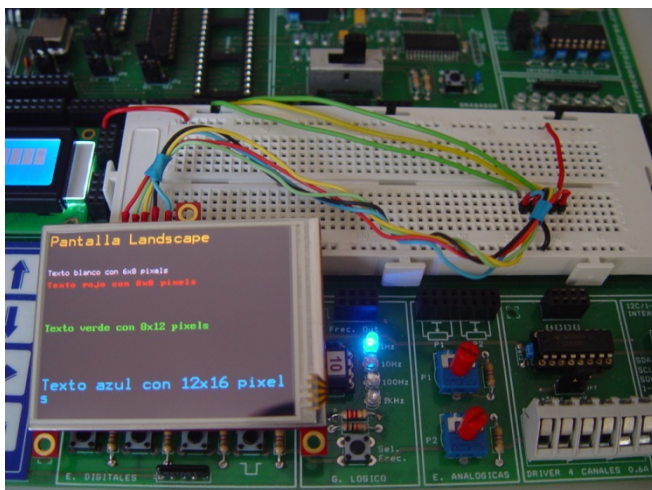
Objetivos

Experimentar con algunas de las funciones o comandos que permiten visualizar texto sobre la pantalla.

Esquema

El mismo que en el ejemplo anterior.

Comentarios



El texto sin formato permite representar cualquier carácter individual y/o también cadenas de caracteres sobre la pantalla. Existen varios tamaños fijos de letras que se pueden representar en cualquier color y en cualquier posición de la misma.

De entre todas las funciones caben destacar la función Portrait y la función Landscape que permite orientar la pantalla en forma vertical u horizontal respectivamente. En el ejemplo, tal y como se muestra en la figura 12, se representan diferentes textos de diferentes colores y con los cuatro tamaños disponibles.

Figura 13. Ejecución del ejemplo 2

8.3 Ejemplo 3: Funciones de texto con formato gráfico

Objetivos

Mostrar el empleo de caracteres y/o cadenas de texto con formato gráfico.

Esquema

El mismo que en el ejemplo anterior

Comentarios

La diferencia fundamental entre el texto sin formato y el texto con formato gráfico es que este último puede verse modificado, a partir de cualquiera de sus tamaños originales, tanto en el aspecto vertical como en el horizontal. Tal y como se muestra en la figura 14, se aprecia diferentes textos de diferentes tamaños que se ven modificados por un factor horizontal y/o vertical.

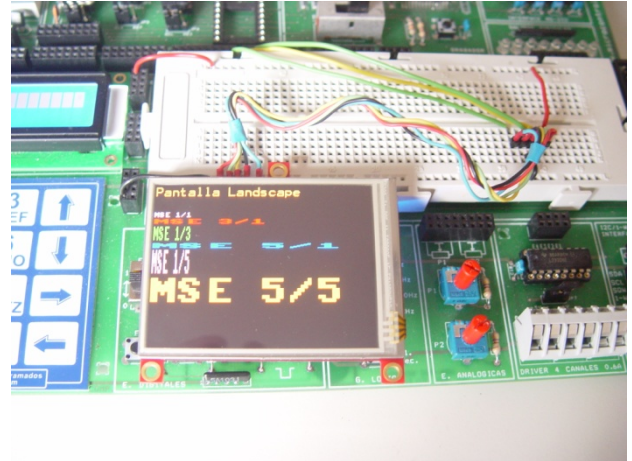


Figura 14. Textos con formato gráfico

8.4 Ejemplo 4: Figuras geométricas

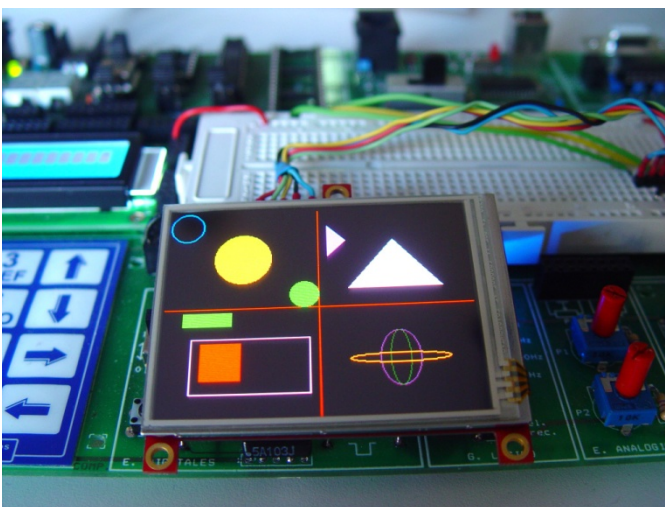
Objetivos

Experimentar con algunas de las funciones o comandos gráficos que el módulo de visualización uOLED-32028-P1T es capaz de interpretar y ejecutar.

Esquema

El mismo que en el ejemplo anterior.

Comentarios



De entre las múltiples funciones disponibles, el módulo de visualización es capaz de dibujar directamente diferentes figuras geométricas como son líneas, círculos, elipses, etc., tal y como se muestran en la figura 15. Además, cada una de esas figuras se puede dibujar en forma sólida (rellena) o no, en cualquier color y en cualquier coordenada de la pantalla. En definitiva, se trata de unas funciones potentes que permiten la inclusión de gráficos en nuestras aplicaciones.

Figura 15. Dibujo de figuras del ejemplo 4

8.5 Ejemplo 5: Reproducción de sonidos WAV

Objetivos

Mostrar la reproducción de un fichero que contienen sonidos en formato wav.

Esquema

El esquema de montaje se muestra en la figura 16. Se emplean las mismas conexiones que las empleadas en los ejemplos anteriores. Además se emplea el pulsador E4 del laboratorio USB-PIC'School que se conecta con la línea de entrada RB1. Cada vez que se acciona este pulsador, se repite la reproducción del fichero WAV de sonido.

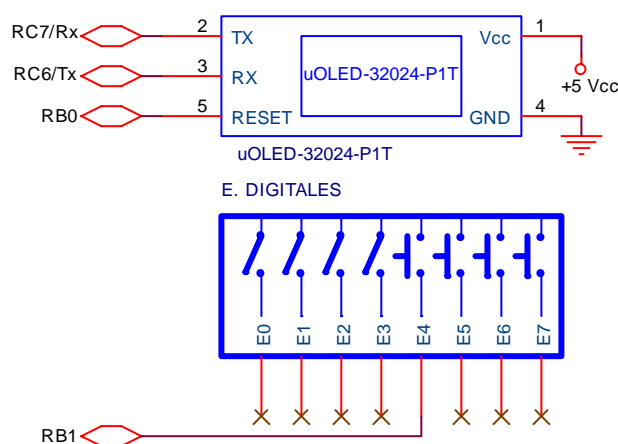


Figura 16. Esquema del ejemplo 5

Comentarios

Una de las grandes aportaciones del módulo de visualización uOLED-32028-P1T es su capacidad para reproducir cualquier tipo de sonido. Efectivamente se trata de un periférico de múltiples posibilidades. Los ficheros de sonido se almacenan sobre una tarjeta de memoria externa uSD que a su vez se inserta en el conector correspondiente dentro del propio módulo.

Existen múltiples aplicaciones y herramientas para Windows capaces de grabar sonidos y guardarlos en un fichero con formato wav. Sin ir más lejos, nosotros hemos empleado la aplicación "**Grabación de Sonidos**" que se incluye en el propio sistema operativo Windows. Con ella hemos creado el fichero "Saludos.wav" que vamos a reproducir en este ejemplo. Ni qué decir tiene que se puede emplear cualquier otro programa de edición de sonidos, de los muchos que existen. El objetivo final es obtener un fichero con formato wav y que se almacenará en la tarjeta de memoria uSD que insertaremos en nuestro módulo.

En el presente ejemplo hacemos uso de los correspondientes comandos y funciones que permiten activar el sistema de reproducción, ajustar el volumen y reproducir el fichero wav deseado. Cada vez que aplicamos un pulso 1-0-1 por la entrada RB1, se reproduce el citado fichero de salutación al tiempo que en pantalla aparece un mensaje informativo.

8.6 Ejemplo 6: Reproducción de ficheros wav de sonido

Objetivos

Reproducir ficheros wav de sonido

Esquema

El mismo que en el ejemplo anterior

Comentarios

Es un ejemplo similar al anterior. En este caso se trata de reproducir varios ficheros wav de sonido que previamente se suponen almacenados en la tarjeta de memoria uSD. La reproducción se realiza de forma secuencial conforme se va accionando el pulsador E4 conectado con la línea de entrada RB1 y en la pantalla van apareciendo el nombre del fichero reproducido como se muestra en la figura 17.

¡¡Pon sonido en tus aplicaciones!!

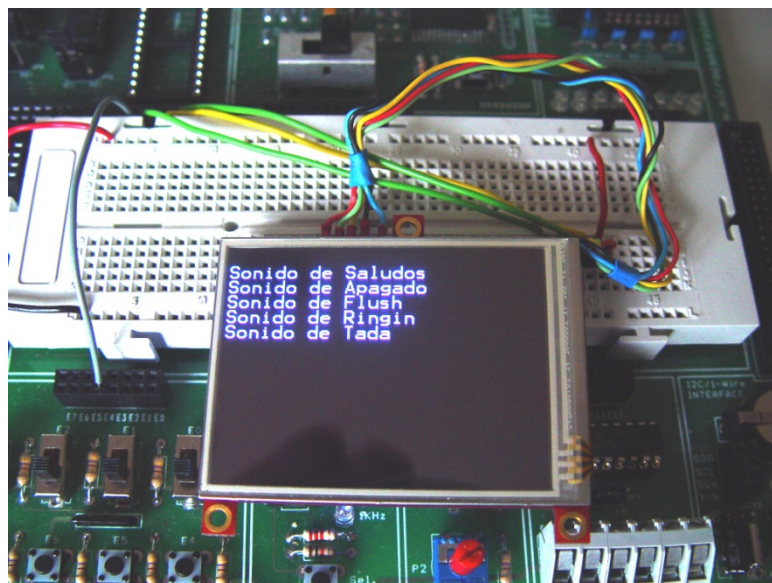


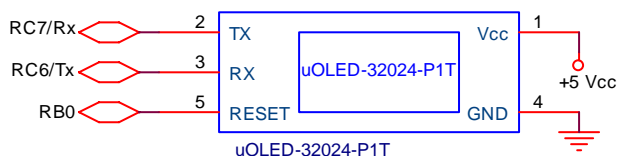
Figura 17. Reproducción de diferentes sonidos wav

8.7 Ejemplo 7: Reproducción de gráficos

Objetivos

Emplear los comandos disponibles para visualizar una imagen almacenada previamente en la tarjeta uSD de memoria

Esquema



El esquema de montaje del presente ejemplos se muestra en la figura 18 y es idéntico al ya empleado en los cuatro primeros ejemplos.

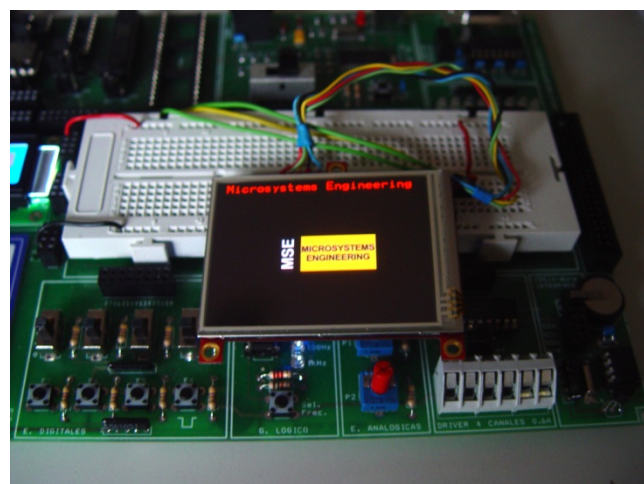
Figura 18. Esquema del ejemplo 7

Comentarios

Otra característica importante del módulo de visualización uOLED-32028-P1T es la posibilidad de visualizar en color cualquier tipo de imagen sobre la pantalla. Dicha imagen se supone almacenada previamente en la tarjeta de uSD de memoria.

El fabricante pone a nuestra disposición y de forma gratuita de una aplicación Windows llamada "Graphics Composer". Dicha aplicación permite aglutinar diferentes imágenes de diferentes formatos y almacenadas en nuestro PC, en un único fichero con extensión GCI que se almacena directamente sobre la tarjeta de memoria uSD y que puede ser leído por el controlador gráfico PICASO de nuestro módulo de visualización. También genera un fichero con extensión DAT que contiene el nombre de las imágenes y en qué sectores se encuentran dentro del fichero GCI almacenado en la tarjeta.

Figura 19. Ejecución del ejemplo 7



Mediante los comandos o funciones oportunas, como las empleadas en el presente ejemplo, indicamos el nombre del fichero GCI en que se encuentra la imagen que deseamos visualizar, y en qué sectores de ese fichero se encuentra. Nosotros hemos empleado el fichero "Presenta.GCI" que contiene la imagen "MSELogo.JPG" que se encuentra en el sector 0 de ese fichero. Como consecuencia de ejecutar este comando, la pantalla visualiza el logo de nuestra empresa como se muestra en la figura 19.

8.8 Ejemplo 8: Reproducción de varias imágenes

Objetivos

Mostrar que la visualización de diferentes imágenes puede estar controlada mediante la acción de señales externas. En el ejemplo se trata de visualizar secuencialmente una serie de imágenes cada vez que se acciona un pulsador.

Esquema

El esquema de montaje para este ejemplo se muestra en la figura 20 y ya se empleó anteriormente. La conexión entre el PIC y el módulo de visualización es la misma que hemos venido utilizando. También empleamos el pulsador E4 del laboratorio USB-PIC'School que se conecta con la línea de entrada RB1. Cada vez que se acciona se visualiza la siguiente imagen sobre el módulo de visualización.

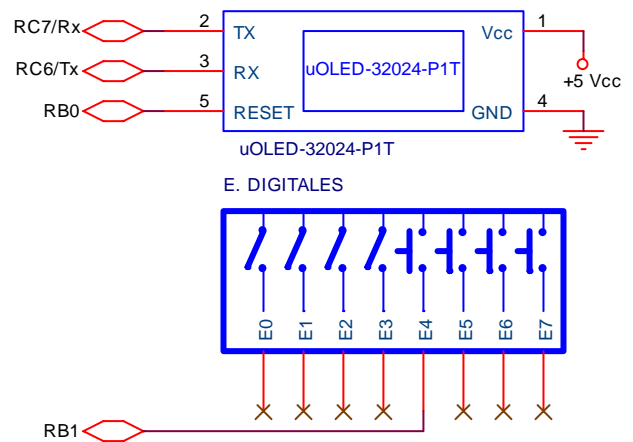


Figura 20. Esquema de montaje del ejemplo 8

Comentarios

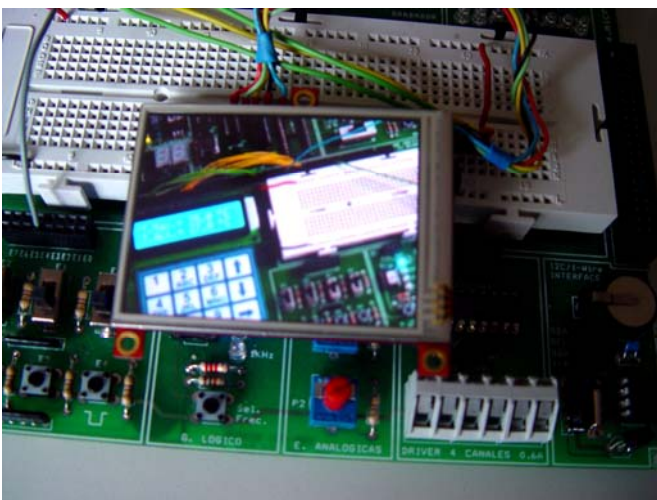


Figura 21. Ejecución del ejemplo 8

En este caso se trata de ir extrayendo y visualizando las diferentes imágenes contenidas en el fichero USBPIC2.GCI que se supone almacenado en la tarjeta de memoria uSD, cada vez que se acciona el pulsador E4 conectado con la línea RB1. Ver la figura 21. Como en el ejemplo anterior, se emplea la aplicación Graphics Composer para aglutinar todas las imágenes que deseamos visualizar en un único fichero USBPICS.GCI que será grabado en la tarjeta uSD. Nuestro programa como tal simplemente hace uso de los comandos oportunos para extraer, en función del sector en que se encuentren dentro del fichero, las distintas imágenes.

Se trata de una buena aplicación que emula un clásico pase de diapositivas.

8.9 Ejemplo 9: Ejecución de un programa tipo script

Objetivos

Emplear los comandos y funciones necesarias para ejecutar, de forma controlada, programas tipo script almacenados en la tarjeta uSD de memoria.

Esquema

Emplearemos el mismo esquema que en el ejemplo anterior

Comentarios

Un programa tipo script es un programa que contiene los comandos o instrucciones que el módulo de visualización ejecutará secuencialmente. Para crear un programa de este tipo, se emplea la aplicación FAT-Controller para Windows que proporciona el fabricante y se supone que, inicialmente, dicho módulo está conectado al PC. Con esta aplicación podemos ejecutar cualquiera de los comandos disponibles y almacenarlos en un fichero sobre el PC con extensión 4DS. Dicho fichero contiene por tanto el conjunto de comandos recién ejecutados y que queremos reproducir posteriormente. A esto se le llama “*programa script*”. A continuación ese fichero se copia sobre la tarjeta uSD que se supone insertada en el módulo. Para ello se emplea el comando adecuado. Todo esto está indicado en el documento “**FAT-Controller-User-Guide**” .

En este momento podemos conectar el módulo a nuestro controlador Host. Nuestro programa puede ahora ordenar que se ejecute cualquiera de los programas script que estén en la tarjeta de memoria uSD. En este ejemplo, cada vez que se acciona el pulsador E4 conectado a RB1, se ordena la ejecución del programa script “PRESENTA .4DS”.

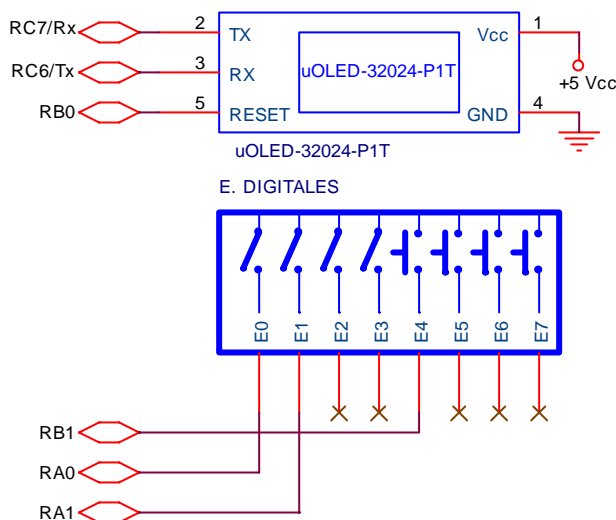
Por las pruebas que hemos realizado observamos que, cuando finaliza la ejecución de un programa script, el módulo no devuelve ningún byte de control tipo ACK o NACK. Así pues lo que nosotros hemos hecho para devolver el control a nuestro PIC Host es realizar una temporización algo mayor que lo que tarda en ejecutarse el programa script que hemos ordenado. AL finalizar esa temporización nuestro PIC podrá continuar con su trabajo.

8.10 Ejemplo 10: Ejecución de varios programas script

Objetivos

Mostrar la ejecución de diferentes programas script de forma totalmente controlada por nuestro PIC Host.

Esquema



Tal y como se muestra en la figura 22, además de las conexiones típicas con el módulo de visualización, se emplean tres señales de entrada. Mediante los interruptores E1:E0 conectados con RA1:RA0 se selecciona qué programa script se va a ejecutar. Mediante el pulsador E4 conectado a RB1 se inicia la ejecución del programa seleccionado.

Figura 22. Esquema de montaje para el ejemplo 10

Comentarios

Nos basamos en los mismos principios que en el ejemplo anterior. Suponemos que en la tarjeta uSD disponemos de tres programas script que se van a ejecutar de forma controlada por nuestro Host. Efectivamente según el estado de los interruptores conectados a RA1:RA0 se selecciona el programa script a ejecutar:

<u>RA1</u>	<u>RA0</u>	<u>Programa script a ejecutar</u>
0	0	Se ejecuta "PRESENTA.4DS" (imágenes de MSE)
0	1	Se ejecuta "USBPICS.4DS" (imágenes del laboratorio USB-PIC'School)
1	X	Se ejecuta "UNITRAM.4DS" (imágenes del entrenador Universal Trainer)

La ejecución del programa seleccionado se inicia cuando se acciona el pulsador E4 conectado en RB1.

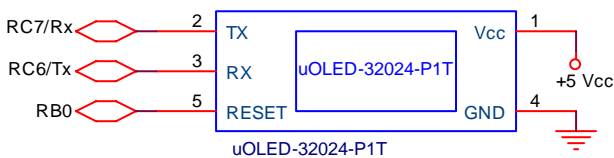
Conviene incidir en las importantes posibilidades que nos proporciona el poder ejecutar uno o varios programas script de forma controlada por nuestro Host. Pensemos que la mayor parte de comandos a ejecutar por el módulo de visualización están contenidos en sus correspondientes ficheros script (4DS), con lo que libramos de esa ardua tarea a nuestro Host que, simplemente, se limita a ordenar la ejecución de uno de esos scripts cuando se considere oportuno.

8.11 Ejemplo 11: El Touch Panel

Objetivos

Mostrar el funcionamiento del panel táctil que integra el módulo de visualización uOLED-32028-P1T.

Esquema



Se muestra en la figura 23 y es idéntico al empleado en otros ejemplos.

Figura 23. Esquema de montaje

Comentarios

La disponibilidad de un Touch Panel convierte al módulo uOLED32024-P1T en un potente periférico de E/S. No sólo visualiza información de salida si no que también permite la entrada de información.

Este ejemplo hace uso de las funciones necesarias para activar el Touch Panel, Determinar el área activa del mismo y obtener las coordenadas del lugar donde se produzca un toque. Con todo ello, el programa se limita a dibujar un pequeño círculo amarillo en el mismo de la pantalla lugar donde se produzca un toque. También se reproduce el clásico sonido "click" almacenado en un fichero wav de sonido. Ver la fotografía de la figura 24.

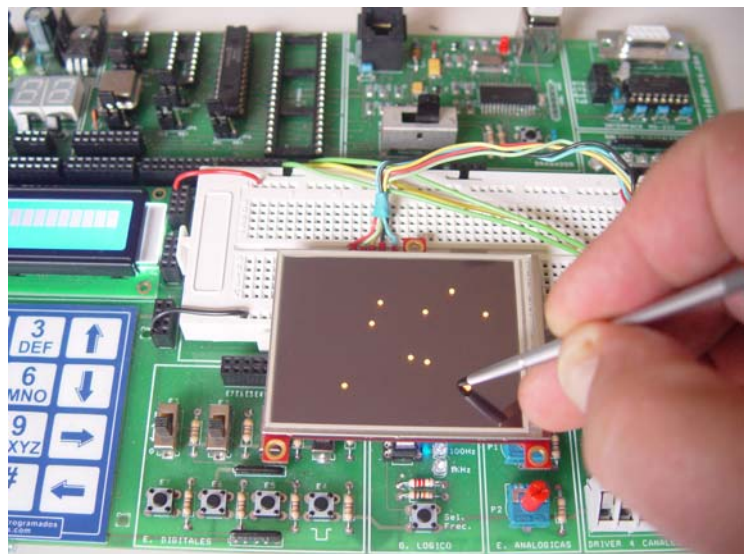


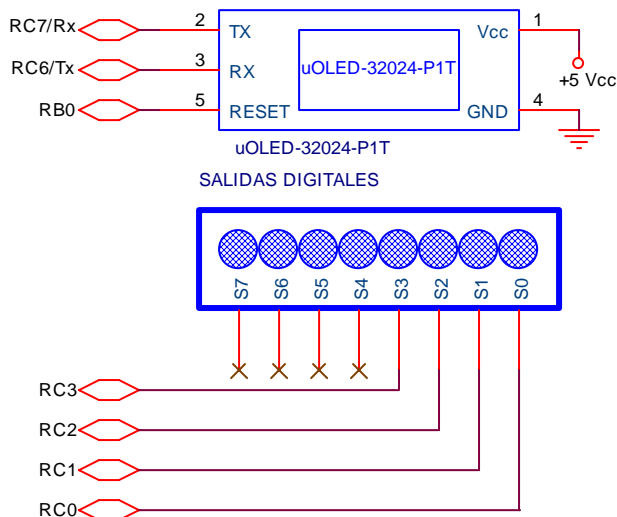
Figura 24. Ejecución del ejemplo 11

8.12 Ejemplo 12: El Touch Panel, detectando zonas o regiones

Objetivos

Mostrar un posible método que permita detectar si el toque sobre el Touch Panel se realiza en una determinada zona o región.

Esquema



Se muestra en la figura 25. El módulo de visualización se conecta como lo hemos venido haciendo en todos los anteriores ejemplo. Ahora también vamos a conectar los leds de salida S3:S0 del laboratorio USB-PIC'School con las líneas RC3:RC0.

Figura 25. Esquema de montaje del ejemplo 12

Comentarios

Tenemos aquí un buen ejemplo real de aplicación: controlar unos periféricos de salida (leds) mediante el interface gráfico que nos proporciona el módulo de visualización y el Touch Panel que integra. Ver la figura 26. Sobre la pantalla se dibujan cuatro círculos que representan a los leds de salida a controlar. Estos aparecerán en azul o negro según estén o no activados. También se dibujan otros tantos botones que, según se accionen o no, controlarán esas salidas. Lo importante de este ejemplo es la técnica que hemos empleado para averiguar si el toque se realiza sobre el área que ocupa cualquiera de los cuatro botones y actuar en consecuencia.

Sobre la memoria flash se han establecido las coordenadas fijas donde se han dibujado los botones. Cada vez que se produce un toque sobre el Touch Panel, se obtienen las coordenadas del mismo. La función "Explora" determina si esas coordenadas pertenecen al área que ocupa cualquiera de los botones y retorna con un código que lo identifica. Si no pertenecieran a ninguno, será un toque inválido y retorna con el código 0xFF.

Con el código de botón que se ha pulsado, el programa actúa haciendo que la línea de salida correspondiente (RC3:RC0) cambie de estado. También se redibuja tanto la tecla como el círculo que representa al led correspondiente.

Figura 26. Ejecución del ejemplo 26

