Trabajo Fin de Grado

DESARROLLO DE MÓDULOS DE APOYO A LAS PRÁCTICAS EN LABORATORIO DE ELECTRÓNICA

Alejandro Agustín Terrado

Grado de Física

ÍNDICE

1. Introducción

La incorporación de nuevas tecnologías al contenido de los programas de Electrónica que imparte el Departamento de Ingeniería Electrónica y Comunicaciones en la Facultad de Ciencias, especialmente en sistemas digitales, ha convertido en habitual el uso en prácticas y trabajos/proyectos de diversos dispositivos de entrada/salida que requieren un conocimiento específico de su estructura y funcionamiento, lo que complica frecuentemente la realización global del sistema al distraer la atención hacia aspectos que pueden ser considerados secundarios desde un punto de vista de diseño funcional global.

Desde un punto de vista clásico, atendiendo a la función que desempeñan, se pueden distinguir dos grupos fundamentales: módulos de representación numérica/gráfica para visualización de resultados, habitualmente LEDs, displays BCD de 7 segmentos o matrices de LEDs, y dispositivos mecánicos/electrónicos para entrada de información, sean datos o comandos operativos.

Sin embargo, la aparición de dispositivos como las pantallas táctiles, que han revolucionado en los últimos años el mercado electrónico, rompe esta división tradicional en entrada/salida, e incorpora elementos interactivos capaces simultáneamente de visualizar objetos gráficos en una pantalla y responder identificando el punto que ha sido tocado por el usuario. Esto permite la realización de menús dinámicos interactivos que cambian ante los estímulos táctiles, siendo así posible que el sistema responda con nuevos menús contextuales.
El más claro ejemplo son los teléfonos móviles actuales, capaces incluso de responder a la activación simultánea de dos puntos de la pantalla.

Creemos que no están siendo aprovechadas las capacidades de las pantallas táctiles en la construcción de interfaces para proyectos didácticos y sistemas sencillos de instrumentación de laboratorio, unas veces por desconocimiento de su funcionamiento y otras tal vez por miedo a la dificultad que plantea su utilización. Además, su rápida evolución en los últimos años y la escasa información disponible sobre cómo integrarlas en un sistema no contribuyen tampoco a facilitar su uso.

Por tanto, profundizar en sus características y modo de operación así como elaborar el soporte necesario para facilitar su utilización como dispositivos de entrada/salida puede resultar una ayuda muy eficaz en diseños en el entorno de laboratorio antes mencionado.

Tampoco hay que olvidar los dispositivos de visualización basados en LEDs, muy útiles cuando los resultados a representar son bastante sencillos. Una labor similar, centrada esencialmente en este caso en la utilización de circuitos integrados (chips) de control de LEDS que liberen al microcontrolador de la tarea rutinaria de activar/desactivar los LEDs que componen dichos dispositivos, sería también muy útil para facilitar su uso.

2. Objetivos

De acuerdo con las consideraciones anteriores, hemos establecido una serie de objetivos básicos para el desarrollo de este Trabajo Fin de Grado que se exponen a continuación:

- Realizar una recopilación actualizada de los dispositivos de representación gráfica disponibles, para lo que es necesario efectuar una labor de búsqueda, selección y caracterización, con una serie de criterios basados en funcionalidad y costo.

- Elaborar el soporte funcional necesario para facilitar su uso en el diseño de sistemas electrónicos de instrumentación o control, ya sea en prácticas de laboratorio o en trabajos/proyectos. Dicho soporte constará de documentación clara y concisa del dispositivo así como de las librerías y/o programas necesarios para su utilización, preferentemente en lenguaje C.

- Profundizar en la programación de microcontroladores y su entorno de buses de comunicación con periféricos, para ser capaz de elaborar dichas herramientas.

- Construir programas de demostración como medio más idóneo para poner de manifiesto algunas de las posibilidades que estos dispositivos nos pueden ofrecer.

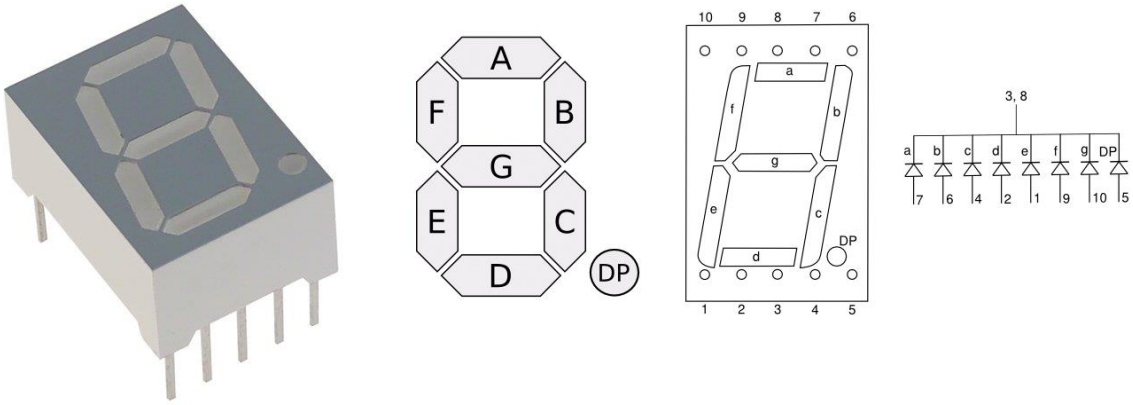
- Estudiar en profundidad el funcionamiento de las pantallas táctiles de los diferentes tipos, como alternativa a los periféricos convencionales de entrada/salida.

3. Matrices y displays de LEDs

Existen muchas formas de representar visualmente los parámetros que manejan los sistemas digitales de información, dependiendo de su tipo y propiedades: valor numérico, barra de nivel, nivel de brillo… Y por ello es muy importante disponer de dispositivos donde se puedan visualizar adecuadamente.

Las alternativas pueden ser displays numéricos de 7 segmentos, matrices de LEDS y pantallas LCD. Veamos brevemente las características de cada uno de ellas:

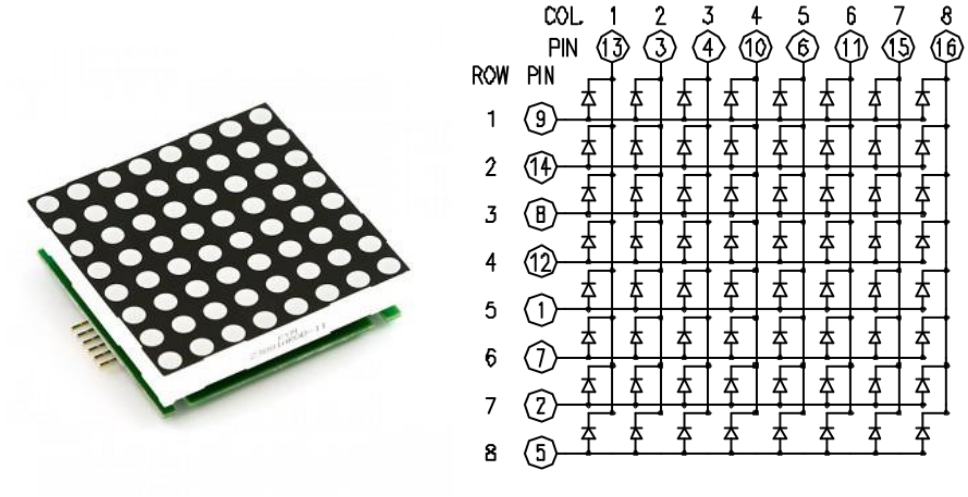
- **Displays de 7 segmentos:** Son dispositivos diseñados para representar gráficamente cifras decimales y algunos símbolos auxiliares, idóneos para aplicaciones donde la salida es básicamente numérica.



Display de 7-segmentos y su representación esquemática

Suelen utilizarse en grupo, para poder representar números de varias cifras. Los diodos que constituyen los segmentos no incorporan habitualmente resistencias en serie, siendo preciso controlar la intensidad externamente mediante resistencias o métodos más sofisticados (PWM).

- **Matrices de LEDs:** Están constituidas por LEDs ordenados uniformemente en filas y columnas, que permiten representar infinidad de símbolos (números, letras, signos de expresión, barras de nivel…). Pueden ser monocolor o RGB, en los que cada punto de la matriz contiene tres diodos: rojo, verde y azul, pudiendo así visualizar también el color. Es evidente que el mecanismo de control de las matrices RGB es mucho más complejo que el de las matrices de LED monocolor y su uso se reserva habitualmente a fines específicos donde el color sea un aspecto fundamental.
  El dispositivo que hemos seleccionado para este trabajo es una matriz de 8x8 LEDs monocromos en cátodo común, 16 pines de conexión, uno por fila y columna.



Matriz de leds monocolor de 8x8 y su representación esquemática

Suelen situarse varias matrices de este tipo en línea para formar un panel más grande donde representar datos más complejos. Como se aprecia en la vista esquemática los diodos no incluyen en serie ninguna resistencia, siendo preciso conectarla externamente o utilizar otro sistema de control de intensidad.

- **Pantallas LCD:** Son pantallas de cristal líquido, con un gran número de pixels. Utilizan la anisotropía del cristal líquido para jugar con la polarización de la luz y conseguir así que el pixel sea o no visible **[WIK04]**. Para esto se requiere la aplicación de campos eléctricos locales controlados por un sistema electrónico, habitualmente un chip específico con una serie de funciones de gestión incorporadas.
  Son alfanuméricas generalmente, y más versátiles que una matriz de LEDs, aunque mucho más complejas de utilizar. Suelen ser monocolor, con retroiluminación incluida, que dejan pasar luz en los píxeles que queremos destacar, o también RGB, teniendo cada píxel tres subpíxeles para generar cualquier color.
  Se utilizan ampliamente en máquinas de venta de productos, en electrodomésticos, y en muchos aparatos comerciales, como medio de representar textos y números de un modo simple y cómodo para el usuario.

Pantalla LCD para Arduino

Su uso generalizado, con múltiples formatos de representación y tamaño, ha provocado que exista en la actualidad una amplia colección de librerías específicas, más que suficientes para utilizarlas sin dificultad en cualquier aplicación.

Por otra parte, las pantallas con capacidad táctil que veremos más adelante mejoran considerablemente sus prestaciones gráficas a la vez que incorporan una opción adicional interactiva al tocar la pantalla en puntos determinados.

En conclusión centramos por ahora nuestro interés en las matrices de LED y displays de 7 segmentos como dispositivos más adecuados para visualizar la salida gráfica de sistemas sencillos y en el desarrollo de un interface que facilite considerablemente su utilización sin tener que entrar en detalles específicos concretos de su estructura.

3.1 Control de LEDs: MAX7219

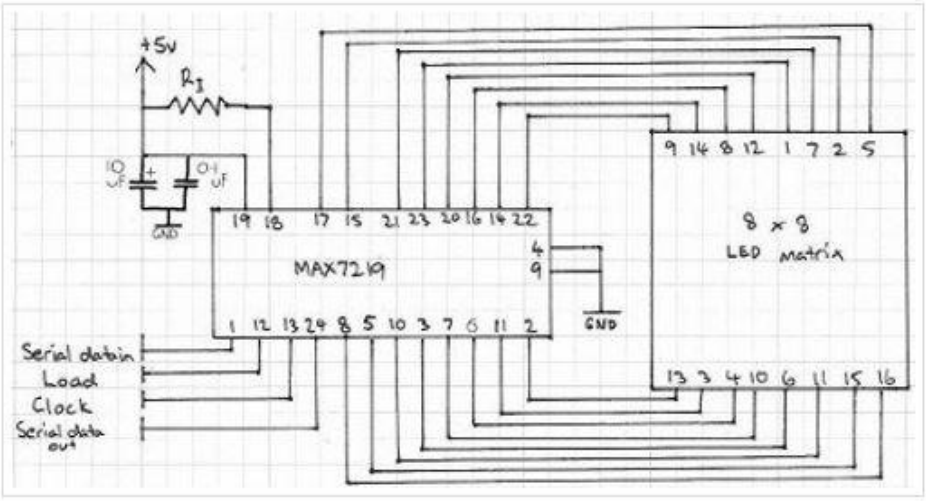
El control directo de una matriz de LEDs mediante la aplicación de tensiones a sus terminales de filas y columnas para encender determinados diodos no resulta sencillo, especialmente por la necesidad de repetir continuamente el proceso y evitar que se observen molestos parpadeos asociados a la frecuencia de apagado/encendido de las filas/columnas que forman la matriz. Además no se incluyen en la matriz resistencias de limitación de corriente, lo que complica considerablemente el control de la intensidad luminosa.

Aunque la activación de displays mediante decodificadores BCD a 7 segmentos es más sencilla el problema también resulta complicado si se trata de controlar varios simultáneamente, como sucede con contadores, relojes…

Es evidente concluir que se requiere un dispositivo adicional de control de estos visualizadores para descargar al microcontrolador de su gestión, simplificando considerablemente el software a desarrollar y elevando la velocidad del proceso.

En base a las referencias consultadas se ha seleccionado el chip MAX7219 de la empresa Maxim, que es capaz de controlar directamente una matriz de LEDs 8x8 o un conjunto de 8 displays de 7 segmentos. Utiliza el protocolo SPI para las comunicaciones, de forma que con tres terminales (reloj, datos, LOAD) podemos enviar toda la información necesaria **[MAX]**.

Otra de las ventajas que presenta este dispositivo se refiere a la regulación de la intensidad luminosa de los LEDs, de modo que, a partir de una referencia de corriente mediante una única resistencia externa, es posible controlarla por software .

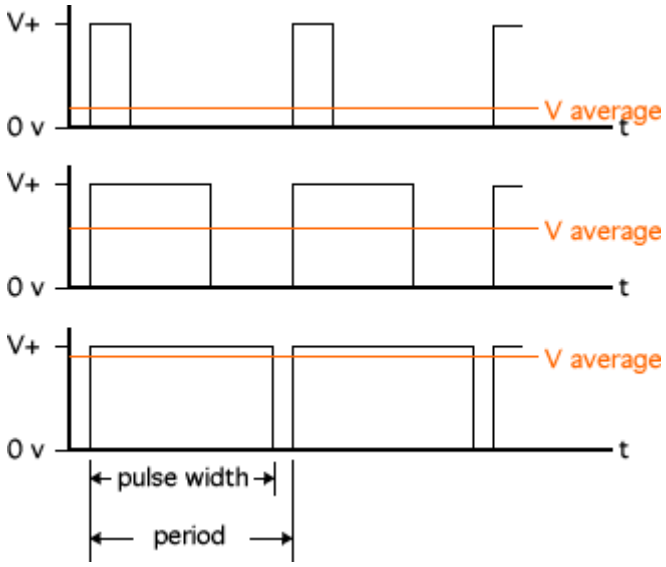


Conexiones del MAX7219

Además, es posible conectar en "serie" varias matrices de LEDs, cada una con su correspondiente MAX7219, con el fin de construir un panel más amplio donde representar lo que queramos. Se suministra también una librería **[ARD02]** (está en la página de referencia de Arduino) que nos proporciona algunas funciones muy útiles que facilitan considerablemente el control de los 64 LEDs de la matriz o los segmentos de 8 displays BCD.

- **LedControl(pin 1,pin 2,pin 3,n)**
  Inicializa los **n** MAX7219 conectados en serie y fija cuales son los pines del Arduino para datos (**pin 1**), reloj (**pin 2**) y LOAD (**pin 3**).
- **shutdown(i,modo)**
  Se activa o desactiva el ahorro de energía para el chip número **i** (según **modo** sea verdadero o falso).
- **clearDisplay(i)**
  Se apagan todos los LEDs de la pantalla para la matriz número **i**.
- **setLed(i,columna,fila,modo)**
  Se enciende o apaga el LED de la **fila** y **columna** seleccionadas de la matriz número **i** en función de si **modo** es true o false.
- **setColumn(i,columna,valor)**
  Se enciende o apaga cada LED de la **columna** seleccionada de la matriz número **i** en función del **valor** introducido, que debe ser un byte (8 bits, uno para cada LED).
- **setRow(i,fila,valor)**
  Realiza lo mismo que **setColumn()** pero para una fila en vez de una columna. Hay que tener en cuenta que debido a la estructura física de la matriz y su relación con el chip MAX7219 la función **setRow()** es ocho veces más rápida que la función **setColumn()**, que utiliza ocho veces la función **setLed()**. Así, siempre que sea posible se recomienda utilizar la función **setRow().**
- **setIntensity(i,brillo)**
  Selecciona el nivel de intensidad luminosa para la matriz número **i**, que debe ser un número entero entre 0 y 15. Dicho control de brillo se realiza mediante la técnica de modulación de anchura de pulsos (PWM) **[SID]**, que permite variar el valor medio de una tensión en forma de tren de pulsos modificando la duración y separación temporal

de los pulsos. De este modo, el ojo integra la respuesta y observará el LED más o menos iluminado, de acuerdo con la intensidad media resultante.



Modulación por anchura de pulsos

Profundizando algo más en la operación del chip MAX7219, y a partir de la estructura de la matriz, se deduce que no enciende simultáneamente todos los píxeles que le indicamos, sino que va realizando barridos de filas de forma continua, por lo que realmente cada LED esta solo activo una fracción muy pequeña de tiempo. A una velocidad de unas 800 filas por segundo el ojo no es capaz de ver estos cambios y observa únicamente LEDs encendidos o apagados.

3.2 Ventajas e inconvenientes

En primer lugar, utilizar el MAX7219 presenta un claro ahorro en la cantidad de pines del microcontrolador utilizados: se requieren 16 para el control directo de una matriz y únicamente tres si se realiza mediante dicho chip, lo que proporciona mucha más flexibilidad para conectar otros dispositivos. Además, desde un punto de vista funcional las ventajas son también evidentes al disponer de un conjunto de funciones para controlar los pixels activos y su intensidad, así como opciones auxiliares para borrar, desplazar…

Es evidente que se requiere un cierto tiempo (ciclos de reloj) para enviar los datos a representar a un registro contenido en el MAX7219, y realizar después las operaciones necesarias para visualizarlos activando secuencialmente las filas en la matriz. Tal vez con un código implementado directamente en el microcontrolador el proceso sería más rápido, pero su complejidad sería enorme y en la práctica resulta casi inviable esta opción, especialmente si el microcontrolador debe atender a otras tareas.

En resumen se puede concluir que el control de una matriz de LEDs de 8x8 se convierte en algo mucho más manejable con el chip MAX7219, y especialmente en el caso de utilizar varias matrices.

3.3 Programa Demostración

Para demostrar el potencial de la representación de resultados con matrices de LEDs hemos realizado una demo donde vamos a visualizar las lecturas de un sensor de temperatura y humedad así como la fecha y la hora. Hemos conectado varias matrices en serie para tener mayor superficie de representación y poder realizar mejor operaciones de scroll, es decir, desplazar el texto y los valores a izquierda y derecha.

La temperatura y la humedad, en grados centígrados y porcentaje, respectivamente, son proporcionadas por un sensor DHT22 **[ADA03]**. Por tanto hemos incluido en el programa su librería correspondiente, así como la librería <time.h> para obtener el día y la hora.

Para realizar la función **scroll** hemos tomado como base una función obtenida de la red **[TRO]** que operaba únicamente con textos guardados en la memoria FLASH (programa), modificando su código para poder utilizar cadenas de caracteres (char) contenidos en la memoria de datos (RAM). Se han realizado también algunas modificaciones en la función para transformar una variable numérica (int, float…) a un tipo char array, de forma que sea posible representar gráficamente cualquier tipo de dato.

Se utiliza una fuente completa de símbolos (7x5 píxeles) para representar la mayoría de los caracteres ASCII relevantes (cifras, letras mayúsculas y minúsculas, signos de puntuación, expresión, etc.).

En la demo realizada se va modificando la velocidad del scroll así como el brillo de los LEDs para poner de manifiesto las opciones funcionales que ofrecen el MAX7219 y la función de scroll.

El programa completo se adjunta en el Anexo 1.

4. Pantallas táctiles

Las pantallas táctiles constituyen en la actualidad un dispositivo periférico de entrada-salida ampliamente usado en sistemas de proceso e instrumentación, como medio para introducir órdenes fácilmente y visualizar los resultados obtenidos. Esto contrasta con la cantidad de periféricos utilizados hace unos años, como ratón, pantalla no táctil y teclado en ordenadores y móviles… Es un paso adelante significativo en el desarrollo de la tecnología, con la ventaja de ser más intuitivo y reducido en tamaño, lo que le confiere una gran potencialidad de aplicación.

El desarrollo de este tipo de pantallas se inició hace bastante tiempo, aunque ha sido en los últimos 5-10 años cuando su uso se ha generalizado a la mayoría de nuestros aparatos: móviles, tablets, ordenadores, electrodomésticos, GPS… Y, al igual que ellos, han ido evolucionando y mejorando sus prestaciones con el tiempo.

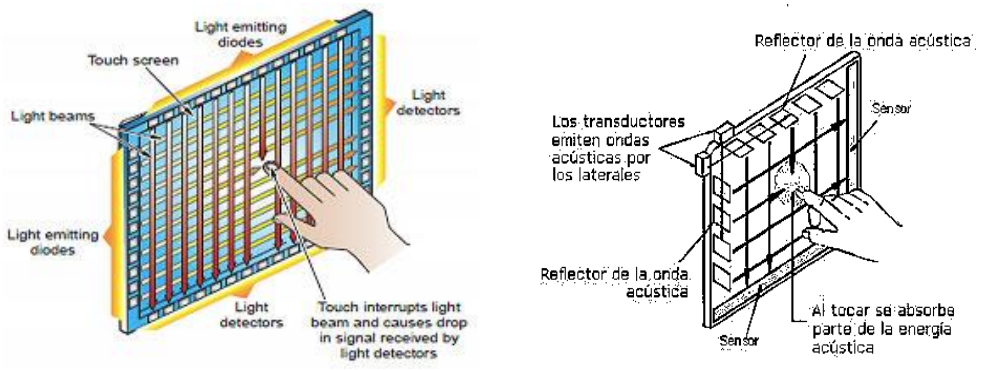
Para la elaboración de este apartado se ha realizado una amplia revisión de las distintas tecnologías disponibles actualmente para la construcción de pantallas táctiles. Parece obvio pensar que la información obtenida nunca será completa ni tal vez la más reciente, ya que son segmentos de mercado electrónico en los que la confidencialidad suele ser bastante estricta. Además, la información no está nunca detallada muy profundamente, especialmente en lo que se refiere a su utilización en aplicaciones de diseño específico.

4.1 Funcionamiento

Hay diversos tipos de pantallas táctiles **[IRT][CIE]**, que pasamos a exponer brevemente a continuación:

- **Por Infrarrojos:** Son pantallas estándar que incorporan emisores de ondas infrarrojas a lo largo de dos de sus bordes (uno en el eje X y otro en el Y) y receptores en el borde contrario. Así, cuando se pulsa la pantalla con el dedo o con un bolígrafo especial para hacerlo (stylus) se interrumpe un rayo vertical y otro horizontal, lo que permite determinar fácilmente las coordenadas del punto seleccionado a partir de un hardware relativamente simple. Fueron las primeras en salir al mercado y su mayor ventaja era que este sistema adicional no afectaba al brillo de la pantalla, pero debido a su alto coste y dificultad de reducción de tamaño están prácticamente obsoletas.
- **Resistivas:** Su estructura consiste en dos capas de material transparente resistivo separadas por un espacio muy pequeño. Al pulsar la pantalla se crea una conexión entre ellas que permite identificar en qué punto se ha pulsado. Su operación se explicarán más adelante.
- **Capacitivas:** Se crea una capacidad distribuida a lo largo de la pantalla, de forma que al tocar con el dedo u otro material no aislante se modifica la capacidad en ese punto de la pantalla y sus alrededores. Este efecto permite determinar el punto donde se ha pulsado.

- **De Onda Acústica:** Son similares a las de infrarrojos pero utilizando una onda acústica de frecuencia inaudible para los humanos. En este caso si se pulsa la pantalla la onda se atenúa, y a los receptores les llega una señal con menor intensidad. Son las más avanzadas en la actualidad y son capaces además de detectar la presión con la que se ha pulsado la pantalla, de forma que quizá sean las más empleadas en un futuro.



Pantallas táctiles por Infrarrojos y de Onda acústica, respectivamente

Una vez introducidos los cuatro tipos más importantes de pantallas táctiles vamos a estudiar en más detalle los dos tipos a los que dedicamos nuestro interés en este trabajo, las resistivas y las capacitivas.

❖ RESISTIVAS

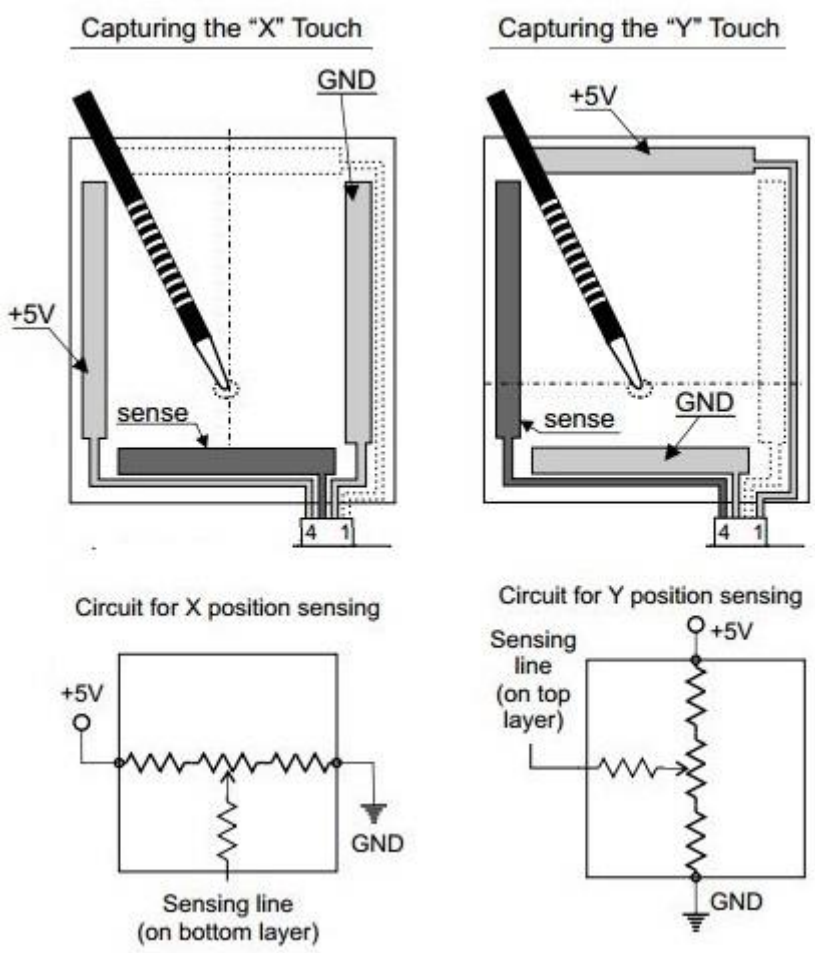
Las pantallas resistivas constan de dos láminas de material conductor, que suele ser óxido de indio dopado con estaño (ITO), separadas por un espacio muy pequeño. Se coloca sobre la lámina exterior una fina capa de un material flexible, generalmente Polietileno, para proteger la pantalla.

Las láminas conductoras constituyen una resistencia distribuida y cuando se pulsa un punto (zona) de la pantalla ambas láminas entran en contacto y se cierra un circuito eléctrico.

Si establecemos una diferencia de potencial entre los extremos horizontales (X) de una lámina y pulsamos en un punto determinado de la pantalla se produce un cortocircuito entre ambas y se forma un divisor de tensión cuyas resistencias serán proporcionales a la situación horizontal de dicho punto. Así, midiendo la tensión en la otra lámina se puede determinar su coordenada X.

Intercambiando el papel de las láminas se puede determinar también la coordenada Y correspondiente.

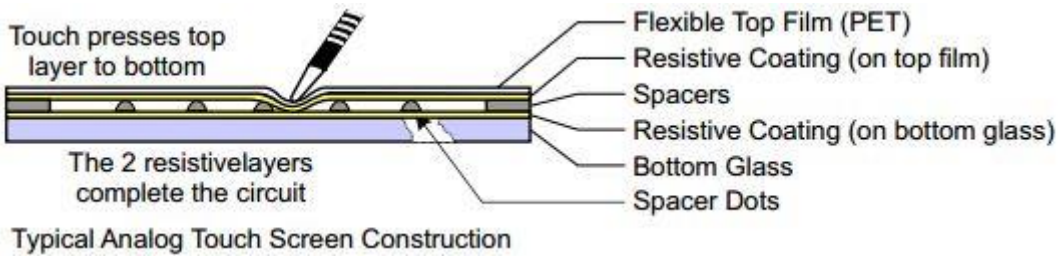
La siguiente imagen puede ayudar a comprender mejor el funcionamiento:



Resumen esquemático de la obtención de un punto en pantallas táctiles resistivas

Aunque no es posible medir ambas tensiones simultáneamente, debido a la elevada velocidad de reloj de los procesadores, esto no supone limitaciones a efectos prácticos, y es posible capturar las coordenadas de varios puntos por segundo sin problemas.

Se incluye una imagen de la estructura de dichas pantallas, con detalle de las capas que las forman, donde se puede apreciar también la necesidad de una base rígida para formar un ente compacto.



Estructura de una pantalla táctil resistiva

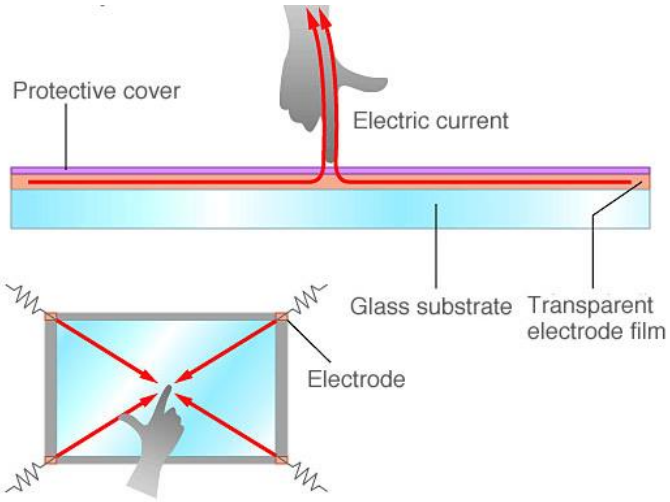
❖ CAPACITIVAS

Las pantallas capacitivas basan su funcionamiento en crear un condensador distribuido que incluye toda la pantalla. Para ello se utiliza una capa de ITO, igual que en las resistivas, y se carga a un determinado potencial. Se pone encima una pequeña capa transparente y protectora. Al situar un dedo en la pantalla circula una pequeña corriente, lo que hace que varíe la capacidad en ese punto y su entorno.

Se sitúan sensores de capacidad en las cuatro esquinas, de forma que cuanto más cerca esté el punto de pulsación, más variará la capacidad medida por el sensor y de esta forma se consigue determinar con suficiente precisión el lugar pulsado.

Por tanto, para que la pantalla funcione adecuadamente es necesario que sea tocada con algo capaz de conducir la electricidad, por ejemplo, nuestro cuerpo.

La imagen siguiente resume la estructura y operación de una pantalla capacitiva:



Operación pantalla capacitiva

Para que el microprocesador detecte la capacidad medida por los sensores situados en las esquinas se utilizan circuitos LC o RC, de forma que la evaluación de su respuesta ante una excitación cuadrada a una frecuencia determinada permite determinar con suficiente precisión el valor de la capacidad en dicho punto.

Es evidente que la construcción de este tipo de dispositivos requiere un minucioso conocimiento y dominio de la tecnología utilizada y deben tenerse especialmente en cuenta métodos para evitar la influencia de las capacidades parásitas, como se explica más detalladamente en **[BER]**.

4.2 Comparación entre pantallas táctiles resistivas y capacitivas

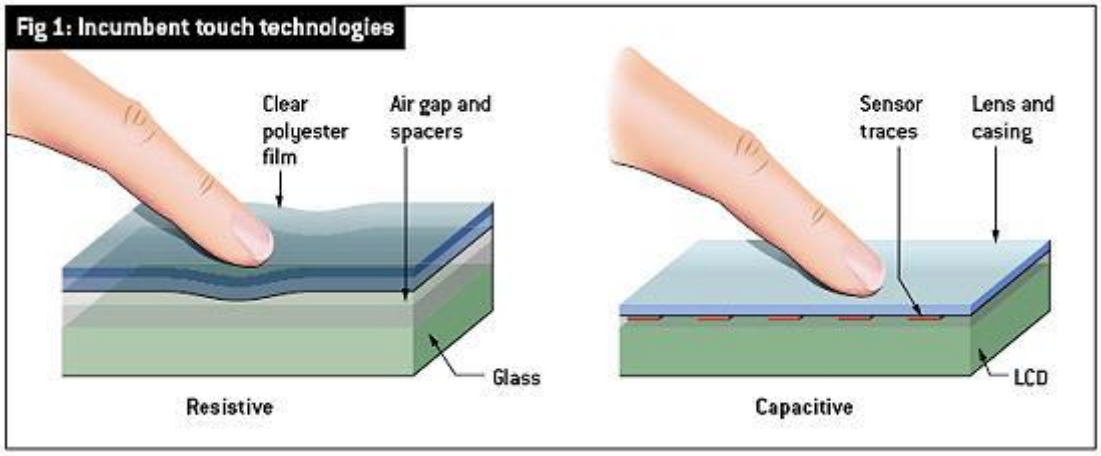
Con el paso del tiempo se han impuesto claramente las pantallas capacitivas a las resistivas y es posible comparar sus diferentes aspectos para deducir el motivo **[TGE]**:

- **Brillo:** Las pantallas capacitivas son más brillantes (hasta un 25%) y tienen una mayor calidad de imagen, lo que las hace más adecuadas para aplicaciones como los smartphones y las tablets, donde esta característica es de gran importancia. Una pantalla resistiva tiene más capas que su homóloga capacitiva, lo que implica un grosor mayor y, por tanto, una menor transparencia.
- **Duración:** La pantallas resistivas deben deformarse para poder funcionar, poniendo las dos capas en contacto entre sí. Esto provoca que el material adquiera un stress que al cabo de un determinado tiempo lo deforma permanentemente, cortocircuitándolo, y dejando la pantalla completamente inoperativa, aunque lógicamente sea necesario para ello una gran cantidad de pulsaciones. En el caso de las capacitivas no existe este problema y la principal razón que limita su vida útil es que su capa superior se raya y se deteriora la visibilidad de la pantalla, siendo por tanto mucho más elevado el tiempo de uso.
- **Sensibilidad del toque:** Como hemos dicho anteriormente, para hacer funcionar la pantalla resistiva hay que hacer una presión, a veces considerable, o nos tenemos que ayudar de un stylus. En el caso de las capacitivas con situar el dedo sobre la pantalla es suficiente, sin usar ningún tipo de fuerza, siendo por tanto mucho mayor su sensibilidad.

  Por el contrario, la pantalla resistiva se puede utilizar haciendo presión con cualquier tipo de objeto, mientras que en la capacitiva debe ser conductor de la electricidad, por lo que no funciona con guantes o stylus, entre otros. Por supuesto, se han desarrollado objetos adecuados para pantallas capacitivas.
- **Capacidad multitáctil:** La operación del sistema de adquisición de puntos de una pantalla resistiva la hace incapaz de procesar varias lecturas de manera simultánea, y para conseguirlo habría que rediseñar a fondo todo su hardware asociado.

  Las capacitivas en cambio sí tienen la posibilidad de realizarlo (con una estructura algo más compleja), extendiendo notablemente su ámbito de aplicación; no hay más que ver que hoy en día miles de dispositivos y aplicaciones utilizan la capacidad de leer varios puntos a la vez.
- **Precisión:** La precisión es en ambas muy similar, y depende fundamentalmente del objeto utilizado para pulsar: un stylus será más preciso que el dedo. Por otra parte, resulta evidente que para muchas aplicaciones no es necesario obtener una precisión hasta el pixel.
- **Precio:** Las pantallas del tipo capacitivo son más caras, aunque debido a su desarrollo cada vez mayor han bajado considerablemente de precio, siendo ya más asequibles. Aunque su precio puede ser un 30 % más elevado, sus prestaciones las hacen idóneas para un gran número de aplicaciones.

Por estas razones las pantallas capacitivas se han impuesto en el mercado, especialmente en el de móviles y tablets, ya que presentan notables ventajas diferenciales y permiten una usabilidad muy superior.
Por estos mismos motivos hemos centrado el presente trabajo en el aprendizaje y utilización de una pantalla capacitiva, dejando a la resistiva un papel más secundario, ya que es muy posible que en unos pocos años también estén obsoletas.



Diferencia en el toque en pantallas táctiles capacitivas y resistivas

4.3 One-touch vs Multi-touch

Cuando se comenzaron a desarrollar las pantallas táctiles eran capaces de responder a un único toque, ya sea por su construcción o porque la tecnología no había avanzado lo suficiente. Hoy en día, una pantalla de nivel medio puede leer varios puntos simultáneos sin ningún tipo de problema.

Para lograrlo, se requiere complicar la construcción y modificar algo su funcionamiento. En este caso el condensador está formado por dos capas conductoras separadas por un dieléctrico. Estas capas tienen unas finas líneas paralelas con un array de electrodos sensores situado en cada una de ellas, y se orientan de forma que las líneas de ambas queden perpendiculares. Así cada nodo es capaz de detectar individualmente si se ha tocado la pantalla en ese punto, puesto que al poner el dedo (u otro material conductor) afectará a la capacidad de dichas líneas **[NEO]**.

Aunque el precio de la pantalla táctil aumenta considerablemente si se trata de una multi-touch, puede ser recomendable su uso por las posibilidades adicionales que ofrecen, como puede ser aplicar un zoom a la imagen representada, trabajar varias personas a la vez en la misma pantalla, realizar diferentes acciones si se pulsa con uno o con dos dedos…

A pesar de que este tipo de pantallas predomina indudablemente en los dispositivos multimedia actuales, decidimos utilizar una pantalla one-touch, debido principalmente a dos factores: el costo económico, y la limitación en los recursos de proceso que impone la utilización de un microcontrolador Arduino, UNO o MEGA, probablemente insuficientes para gestionar adecuadamente estos dispositivos tan complejos.

Además, los objetivos fundamentales de este trabajo consistentes en desarrollar unos interfaces económicos, simples y versátiles para un entorno de prácticas o trabajos académicos de laboratorio así como para el diseño de sistemas de instrumentación sencillos son perfectamente realizables con la pantalla seleccionada.

4.4 Futuro

Estos dispositivos de visualización-interactuación con el usuario no dejan de evolucionar y perfeccionarse, como lo demuestran algunos avances significativos: pantallas curvadas, resistentes al agua…
También van surgiendo distintos modelos de pantallas en 3D, lo cual supondría otro paso adelante acercándonos aún más a formas de representación con mayor parecido al mundo natural.

Por otro lado es posible que se dejen de dar las instrucciones mediante el tacto y se haga mediante comandos de voz, movimientos oculares u otros métodos que ahora mismo no seamos capaces de imaginar.

Indudablemente esas nuevas alternativas se habrán alcanzado gracias a haber sido capaces de desarrollar cada vez más las pantallas táctiles actuales, de modo que su comprensión y estudio continuo ayudarán a descubrir y mejorar este mundo de nuevas posibilidades.

5. Pantalla y Software

Una vez recopilada suficiente información acerca de las pantallas táctiles, su funcionamiento y características se realizó un pequeño muestreo de mercado para intentar determinar la más idónea para nuestros propósitos, con el resultado expuesto a continuación.

5.1 Pantalla Táctil Capacitiva One-Touch

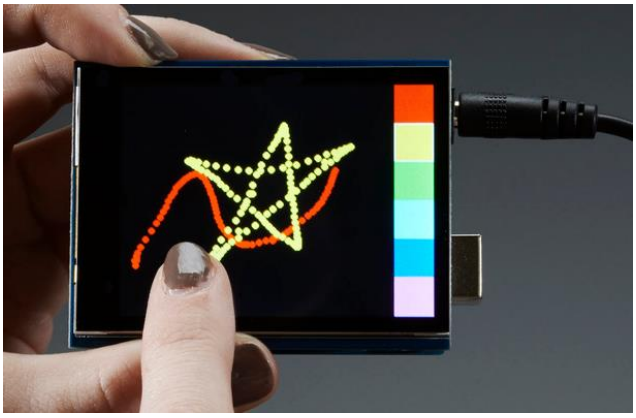
El dispositivo seleccionado es una pantalla capacitiva de la casa Adafruit, de 2.8'' de tamaño y diseñada para su utilización con Arduino **[ADA02]**. Está integrada en una placa preparada para acoplarse directamente al microcontrolador, que incluye también el correspondiente hardware de control.

Posee las siguientes especificaciones:

- 2.8'' de diagonal
- Resolución de 240x320 píxeles a color (18 bits)
- Buses de comunicaciones :
    - SPI entre el chip de control y la pantalla
    - I2C entre Arduino y el chip de control
- Lector de tarjeta microSD
- Capacitiva one-touch
- Tensión de alimentación : 3,3 V (incluye su propio regulador de tensión)

Como material de soporte se suministran las librerías necesarias para la gestión de los gráficos representados en la pantalla y para detectar cuando se toca en un punto e identificarlo. Esta información ha servido de base a nuestro trabajo para introducirnos en su estrategia de operación, descartando las funciones que no presentaban utilidad y aprovechando, con las modificaciones necesarias, las que se adaptaban mejor a nuestros objetivos.

A modo de ejemplo se incluye también con la librería un sencillo programa donde sobre un panel en negro se va dibujando con el dedo una figura, previa selección del color.



De acuerdo con las instrucciones de Adafruit se requiere una pequeña modificación (corte de pistas) en la placa para utilizarla indistintamente con Arduino UNO o MEGA. Se seleccionan así

para la comunicación I2C los pines específicos que poseen las placas Arduino en su parte central, liberando para otros usos los pines alternativos incluidos en los conectores de entrada/salida.

El entorno de interface operativo aquí desarrollado toma como referencia el Arduino MEGA, debido a su mayor capacidad de memoria y disponibilidad de pines. Se dispone así de más espacio para programas complejos y de mayor versatilidad para conectar periféricos adicionales.

	Arduino UNO	Arduino MEGA
Frecuencia reloj	16 MHz	16 MHz
Memoria RAM	2 kB	8 kB
Memoria EEPROM	1 kB	4 kB
Memoria FLASH	32 kB	256 kB
Pines digitales	14	54
Pines analógicos	6	16

Antes de empezar a trabajar con esta pantalla capacitiva se realizaron pruebas preliminares con otra resistiva de 3.2'' de diagonal, mucho más económica, si bien encontrar las librerías necesarias y algunos ejemplos de utilización resultó una ardua tarea de búsqueda en páginas y foros de internet.

Esta dificultad de obtener información y librerías "oficiales" así como las razones mencionadas anteriormente en la comparativa entre ambos tipos nos decidió definitivamente por centrar nuestro trabajo en la capacitiva.

Pensamos que no tiene mucho sentido desarrollar una aplicación sobre un hardware con información escasa y no siempre fiable, que además puede quedar obsoleto en relativamente poco tiempo.

5.2 Librería PantCapacitiva

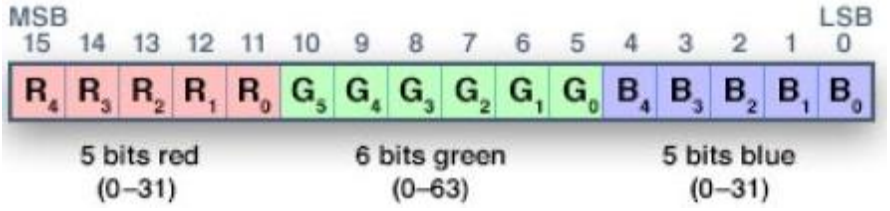
Para simplificar la utilización de la pantalla capacitiva seleccionada se ha elaborado una librería (<PantCapacitiva.h>), especialmente diseñada para aprovechar fácilmente sus prestaciones e incluirla sin dificultad en programas específicos. Constituye un elemento de gran ayuda en el diseño de prácticas de laboratorio o en la realización de sistemas de instrumentación donde se considere adecuado interactuar con el microcontrolador mediante el interface visual aquí desarrollado.

Se han utilizado algunas funciones de las librerías originales, averiguando el significado de los parámetros que utilizan y también creando algunas nuevas a partir de ellas. La librería está abierta a futuras modificaciones y mejoras, adaptando su código y sus funciones a las necesidades del proyecto concreto.

El código de la librería se adjunta en el Anexo 2, presentando a continuación un breve resumen de las funciones que contiene, para la parte de representación gráfica y de adquisición de puntos, sus argumentos y la operación que realizan.

- **Inicia ()**
  Inicializa la pantalla táctil.
- **PintaPantalla (int color)**
  Pinta la pantalla del **color** que se le introduce.
- **DibujaPixel (int x,int y,int color)**
  Dibuja el pixel (**x**,**y**) del **color** indicado.
- **DibujaLinea (int x0,int y0,int xf,int yf,int color)**
  Dibuja una línea desde (**x0**,**y0**) hasta (**xf**,**yf**) del **color** indicado.
- **DibujaCirculo (int x,int y,int radio,int color,bool modo)**
  Dibuja un círculo del **color** seleccionado y centrado en **x**,**y** y con el **radio** introducido.
  Si **modo** = 0 no rellenará el interior pero si es 1 sí que lo hará.
- **DibujaRectangulo (int x,int y,int anchura,int altura,int color,bool modo)**
  Dibuja un rectángulo del **color** seleccionado con el vértice en **x**,**y** y la **anchura** y **altura** introducidas.
  Si **modo** = 0 no lo rellenará y si es 1 lo hará.
- **DibujaTriangulo (int x1,int y1,int x2,int y2,int x3,int y3,int color,bool modo)**
  Dibuja un triángulo del **color** seleccionado a partir de sus 3 vértices.
  Si **modo** = 0 no lo rellenará y si modo = 1 sí lo hará.
- **RotaPantalla (int R)**
  Rota la pantalla: 0 - Modo normal, 1 - 90°, 2 - 180°, 3 - 270°.
- **EscribeTexto (int x0,int y0,bool sub,int color,int color2,int tam,bool salto,any type arg)**
  Escribe un texto desde el punto **x0**,**y0** (vértice superior izquierdo del texto).
  Además, si **sub** = 1 el texto se resaltará con un color de fondo (**color2**).
  El tamaño (**tam**) debe ser un número entero (entre 1 y 6).
  Si el texto no cabe en una línea se cortará si **salto** = 0, mientras que si es igual a 1 continuará en la siguiente línea.
  El último argumento (**arg**) es el propio texto a escribir
- **ObtenPuntoTactil (int *x,int *y,int R)**
  Devuelve en las variables **x** e **y** (se introducen a la función como punteros) las coordenadas del punto tocado en la pantalla, en el sistemas de referencia correspondiente al modo de rotación **R** .

En la mayoría de las funciones hay que introducir el parámetro "**color**". Debe ser una variable de 16 bits y se introduce en código RGB, siendo los 5 primeros bits para el rojo, los 6 siguientes para el verde y los 5 últimos para el azul, según el siguiente formato:



RGB en 16 bits para la pantalla táctil

Para simplificar la escritura de código se han incluido sentencias específicas (#define) para definir algunos colores frecuentemente utilizados:

- Negro
- Azul
- Rojo
- Verde
- Naranja
- Rosa
- Amarillo
- Blanco

Es importante prestar atención a la orientación de los ejes de coordenadas de la pantalla, por defecto el origen de coordenadas (0,0) es la esquina más próxima a la entrada de alimentación externa del Arduino. Se considera positivo el sentido hacia la derecha en el eje X y hacia abajo en el eje Y, de forma que la coordenada X tomará un valor entre 1 y 240 y la coordenada Y entre 1 y 320.

Al realizar una rotación se intercambian los ejes de coordenadas, aunque se mantienen los sentidos positivos hacia la derecha y hacia abajo, respectivamente. La rotación gira la pantalla en sentido antihorario los grados correspondientes al valor de **R** suministrado a la función **RotaPantalla().**

La librería completa <PantCapacitiva.h> (Anexo 2) incluye también otras librerías complementarias necesarias para el funcionamiento del conjunto, tanto las que proporciona el fabricante de la pantalla como otras de Arduino para la comunicación SPI y utilidades adicionales. En concreto, están incluidas las siguientes:

- ✓ Arduino.h
- ✓ Adafruit_FT6206.h
- ✓ Adafruit_GFX.h
- ✓ Adafruit_ILI9341.h
- ✓ SPI.h
- ✓ Wire.h

Así, la librería construida en este trabajo contiene todo lo necesario para que, de una forma rápida y sencilla, puedan diseñarse entornos gráficos interactivos con menús de opciones contextuales que aprovechen la propiedad táctil de la pantalla para introducir información al sistema.

Hay que poner de manifiesto también que no han surgido problemas importantes para su realización, aparte de la necesidad de asimilar muchos conceptos previos y comprender en profundidad el funcionamiento y significado de los parámetros de las librerías que han sido incluidas en ella.

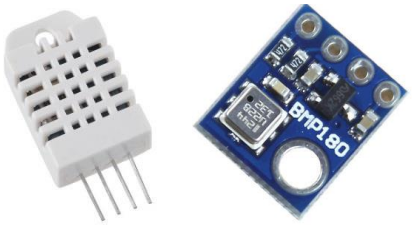
5.3 Utilidad del Interface Gráfico Desarrollado

Se cumple por tanto uno de los objetivos fundamentales de este trabajo, que era aprender a utilizar una pantalla táctil y su entorno gráfico asociado, desarrollando asimismo una herramienta de gestión que permita su utilización como dispositivo muy versátil de entrada/salida en proyectos o diseños electrónicos. Así, es posible representar información suministrada por el microcontrolador o construir menús gráficos interactivos que puedan ser activados por un simple toque en una zona de la pantalla, de una forma mucho más ágil y cómoda que utilizando alternativas clásicas para la visualización (LCD/LEDs) o entrada de órdenes (conmutadores/pulsadores).

5.4 Programa Demostración

Para poner de manifiesto el potencial que puede incorporar el interface gráfico desarrollado al diseño de un sistema de medida o instrumentación electrónica se ha diseñado un programa interactivo de demostración para la visualización de diferentes parámetros ambientales: Temperatura, Humedad, Presión y Altitud.

Para ello se utilizan dos sensores, el DHT22 **[ADA03]**, que mide la temperatura y la humedad y el BMP180 **[ADA04]**, que mide la temperatura y la presión y calcula a partir de ella la altitud. Utilizaremos para la temperatura las lecturas suministradas por el sensor DHT22.



DHT22 y BMP180, respectivamente

Se parte de un menú gráfico que permita seleccionar la magnitud a medir y sus unidades, pulsando en las correspondientes áreas de pantalla reservadas a tal efecto.

Hay que insistir especialmente en que tanto la distribución como el contenido de estos menús son completamente dinámicos y pueden cambiar de acuerdo con las opciones escogidas previamente.

Una vez seleccionada una alternativa inicial la lectura de medidas es automática, refrescando el valor mostrado en la pantalla a intervalos constantes (aproximadamente 8 segundos)

Por supuesto que en cualquier momento es posible volver al menú principal para modificar la selección inicial. Además, en el caso de la temperatura, se puede escoger también la unidad de representación del resultado: grados centígrados o Kelvin.

La humedad se muestra en %, la presión en mbar y la altitud en metros. Las precisiones son del orden de la décima de grado en la temperatura, y una unidad en las restantes, y vienen impuestas generalmente por el sensor, ya sea por sus características o por el error asociado a la medida. Utilizando técnicas de promediado es posible mejorar la precisión resultante, pero

hay que tener presente que en nuestros objetivos no está alcanzar unas medidas muy precisas, lo que pasaría sin duda por la elección de otros sensores más sofisticados y costosos.

Para la realización de este demo se ha creado una librería nueva, llamada <Demo.h> donde se han incluido todas las funciones necesarias para que el programa se ejecute adecuadamente. El código principal del programa y la librería citada se adjuntan en el Anexo 3.

Como ya se indicó anteriormente se requiere un Arduino MEGA para disponer de los pines suficientes para conectar los sensores utilizados: alimentación y datos.

El sensor BMP180 utiliza el protocolo I2C para las comunicaciones, lo que fija unos pines determinados del Arduino (20 para el reloj y 21 para los datos). En cambio, el DHT utiliza un único pin para enviar al microcontrolador el resultado de la medida en formato digital, pudiéndose utilizar por tanto cualquier pin digital disponible .

Ambos sensores deben ser inicializados antes de comenzar a adquirir medidas y suministran los resultados en formato float, lo que no constituye un problema dado que las funciones de la librería <PantCapacitiva.h> admiten cualquier tipo de formato para los datos.

6. Resultados

Se exponen a continuación los principales resultados obtenidos en este trabajo, que creemos se ajustan aceptablemente a los objetivos proyectados al inicio del mismo:

- Recopilación de información acerca de dispositivos de visualización de salida, con especial interés en los basados en LEDs: displays de 7 segmentos y matrices 8x8 .

- Revisión de chips controladores para grupos de displays de LEDs de 7 segmentos o matrices, habiendo seleccionado inicialmente el MAX7219 por sus características, precio y disponibilidad. Después de comprobar su funcionamiento, hemos profundizado en el conocimiento de la librería estándar <LedControl.h>, resumiendo en una breve documentación sus funciones principales.

- Elaboración de un programa de demostración donde se controlan varias matrices 8x8, visualizando mediante "scroll" diferentes magnitudes, que puede servir como base, con las modificaciones pertinentes, para futuras aplicaciones.

- Recopilación de información acerca de los diferentes tipos de pantallas táctiles existentes, funcionamiento y posibilidades que ofrecen.

- Estudio de mercado sobre dichos periféricos: precio y facilidades de uso. Aunque inicialmente seleccionamos dos pantallas, una resistiva y otra capacitiva, la comparación entre ambas nos ha hecho centrar el desarrollo posterior en la de tipo capacitivo.

- Elaboración de una librería adaptada a las necesidades de este trabajo, y su documentación correspondiente, que facilitará el usos de la pantalla en proyectos futuros.

- Construcción de un programa de demostración para poner de manifiesto el potencial de la pantalla táctil como periférico de entrada y salida.

- Obtención de un mayor conocimiento en la programación de los microcontroladores Arduino y sus diferentes buses de comunicación.

7. Conclusiones

Personalmente, creo que el trabajo ha sido bastante enriquecedor, puesto que me ha permitido seguir avanzando en la programación con microcontroladores, un tema que me parece bastante interesante, además del conocimiento más a fondo de otros dispositivos electrónicos.

También el haber investigado en un tema tan actual como las pantallas táctiles me ha dado una visión interesante acerca de este tipo de tecnologías, su utilización en dispositivos de uso diario y también de la dificultad de encontrar una información clara y completa sobre ellos.

El tiempo invertido realizando los distintos programas y librerías me hacen valorar la dificultad que plantea diseñar y poner a punto cualquier dispositivo electrónico de los que utilizamos habitualmente, esfuerzo a menudo frustrante porque un pequeño fallo, difícil de localizar en ocasiones, puede hacer que el sistema no funcione adecuadamente. No obstante, se trata de una labor necesaria y valiosa asociada al desarrollo de todos los dispositivos que contribuyen en alguna forma a hacernos la vida más sencilla y mejor.

Por otra parte, también me gusta pensar que se aprovechará el trabajo realizado con las pantallas táctiles y las matrices de LEDs y servirá para incluirlas sin mayor dificultad como periféricos en un sistema de control o instrumentación.

8. Referencias

Hay que tener en cuenta que nuestra búsqueda de información sobre las pantallas táctiles se ha centrado principalmente en aspectos prácticos y no en la tecnología necesaria para su construcción, claramente fuera de los objetivos de este trabajo.

Por ello, dada su actualidad y rápida evolución, la mayoría de las referencias aquí expuestas son direcciones de internet y no publicaciones en papel.

[ADA01]: http://www.adafruit.com/datasheets/ILI9341.pdf

[ADA02]: https://www.adafruit.com/products/1947

[ADA03]: https://www.adafruit.com/products/385

[ADA04]: https://www.adafruit.com/products/1603

[ARD01]: http://playground.arduino.cc/Main/MAX72XXHardware

[ARD02]: http://playground.arduino.cc/Main/LedControl

[ASC]: http://www.asciitable.com/

[BER]: http://inst.eecs.berkeley.edu/~ee16a/sp15/Labs/touchscreen/ee16a_touchscreen_lab3.html

[CIE]: https://cienciaes.com/ciencianuestra/2014/01/10/como-funcionan-las-pantallas-tactiles/

[COH]: http://comohacer.eu/analisis-comparativo-placas-arduino-oficiales-compatibles/?utm_content=bufferc52d3&utm_medium=social&utm_source=twitter.com&utm_campaign=buffer

[CUL]: http://culturacion.com/que-son-y-como-funcionan-las-pantallas-tactiles/

[EEN01]: http://www.electroensaimada.com/spi.html

[EEN02]: http://www.electroensaimada.com/i2c.html

[EIZ]: http://www.eizoglobal.com/library/basics/basic_understanding_of_touch_panel/

[ELD]: https://sites.google.com/site/electronicadigitaluvfime/5-1tipos-de-memorias-ram-rom-dram-sram

[IRT]: http://www.irontech.es/Blog%20Posts/tipos-pantallas-tactiles.html

[MAX]: https://datasheets.maximintegrated.com/en/ds/MAX7219-MAX7221.pdf

[NEO]: http://www.neoteo.com/multi-touch-touchscreen-pantallas-tactiles

[OCP]: http://octavaplanta.es/2012/05/sabes-como-funciona-la-pantalla-tactil-de-tu-smartphone/

[SID]: Contenido Asignatura Sistemas Digitales (26957) del Grado de Física (Unizar)

[SPF]: https://www.sparkfun.com/datasheets/LCD/HOW%20DOES%20IT%20WORK.pdf

[TGE]: http://www.tecnogeek.com/verpost.php?id_noticia=817

[TRO]: http://tronixstuff.com/2013/10/11/tutorial-arduino-max7219-led-display-driver-ic/

[WIK01]: https://en.wikipedia.org/wiki/Capacitive_sensing

[WIK02]: https://en.wikipedia.org/wiki/Multi-touch

[WIK03]: https://es.wikipedia.org/wiki/Pantalla_táctil

[WIK04]: https://es.wikipedia.org/wiki/Pantalla_de_cristal_líquido

ANEXOS

ANEXO 1

```
#include <avr/pgmspace.h>
#include <LedControl.h>
#include <math.h>
#include <Time.h>
#include <DHT.h>

#define DHTPIN 12 //Pin al que se conecta el sensor
#define DHTTYPE DHT22 // Tipo de sensor

void floattochar(float x,char *num);
void inttochar(int x,char *num);

const int numDevices = 3;      // number of MAX7219s used
long scrollDelay = 1;   // adjust scrolling speed
int brillo = 15;  //ajusta el brillo

unsigned long bufferLong [14] = {0};

LedControl lc=LedControl(5,4,3,numDevices);

DHT dht(DHTPIN, DHTTYPE); // Inicialización del sensor

void setup(){
  Serial.begin(9600);
    for (int x=0; x<numDevices; x++){
       lc.shutdown(x,false);       //The MAX72XX is in power-saving mode on startup
       lc.setIntensity(x,brillo);       // Set the brightness to default value
       lc.clearDisplay(x);         // and clear the display
    }
    setTime(11,57,45,21,10,2015);
}

void loop(){
    char *T,*H,*hora,*minuto,*seg,*dia,*mess,*anho;
    int h,m,s,d,mes,a;
    float hum;    // % humedad (DHT22)
    float temp;   // temperatura (DHT22 , ºC)
    h=hour();
    m=minute();
    s=second();
    d=day();
    mes=month();
    a=year();
    hum = dht.readHumidity();
    temp = dht.readTemperature();
    T=(char*)malloc((int(log10(temp)+3))*sizeof(char));
    floattochar(temp,T);
    H=(char*)malloc((int(log10(hum)+3))*sizeof(char));
    floattochar(hum,H);
```

```
  hora=(char*)malloc((int(log10(h)+1))*sizeof(char));
  inttochar(h,hora);
  minuto=(char*)malloc((int(log10(m)+1))*sizeof(char));
  inttochar(m,minuto);
  seg=(char*)malloc((int(log10(s)+1))*sizeof(char));
  inttochar(s,seg);
  dia=(char*)malloc((int(log10(d)+1))*sizeof(char));
  inttochar(d,dia);
  mess=(char*)malloc((int(log10(mes)+1))*sizeof(char));
  inttochar(mes,mess);
  anho=(char*)malloc((int(log10(a)+1))*sizeof(char));
  inttochar(a,anho);

  scrollMessage("T=");
  scrollNumber(T,int(log10(temp)+3));
  scrollMessage("~C      ");
  delay(1000);
  scrollMessage("Humedad=");
  scrollNumber(H,int(log10(hum)+3));
  scrollMessage("%      ");
  delay(1000);
  scrollNumber(hora,int(log10(h)+1));
  scrollMessage(":");
  scrollNumber(minuto,int(log10(m)+1));
  scrollMessage(":");
  scrollNumber(seg,int(log10(s)+1));
  scrollMessage("       ");
  delay(1000);
  scrollNumber(dia,int(log10(d)+1));
  scrollMessage("/");
  scrollNumber(mess,int(log10(mes)+1));
  scrollMessage("/");
  scrollNumber(anho,int(log10(a)+1));
  scrollMessage("       ");
  delay(2000);

  if(brillo != 1)
  {
    brillo=1;
  }
  else
  {
    brillo=15;
  }
  for (int x=0; x<numDevices; x++){
    lc.setIntensity(x,brillo);
  }
}

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

prog_uchar font5x7 [] PROGMEM = {      //Numeric Font Matrix (Arranged as 7x font data + 1x kerning data)
```

```
B00000000,  //Space (Char 0x20)
B00000000,
B00000000,
B00000000,
B00000000,
B00000000,
B00000000,
6,

B10000000,  //!
B10000000,
B10000000,
B10000000,
B00000000,
B00000000,
B10000000,
2,

B10100000,  //"
B10100000,
B10100000,
B00000000,
B00000000,
B00000000,
B00000000,
4,

B01010000,  //#
B01010000,
B11111000,
B01010000,
B11111000,
B01010000,
B01010000,
6,

B00100000,  //$
B01111000,
B10100000,
B01110000,
B00101000,
B11110000,
B00100000,
6,

B11000000,  //%
B11001000,
B00010000,
B00100000,
B01000000,
B10011000,
B00011000,
```

```
6,

B01100000,   //&
B10010000,
B10100000,
B01000000,
B10101000,
B10010000,
B01101000,
6,

B11000000,   //'
B01000000,
B10000000,
B00000000,
B00000000,
B00000000,
B00000000,
3,

B00100000,   //(
B01000000,
B10000000,
B10000000,
B10000000,
B01000000,
B00100000,
4,

B10000000,   //)
B01000000,
B00100000,
B00100000,
B00100000,
B01000000,
B10000000,
4,

B00000000,   //*
B00100000,
B10101000,
B01110000,
B10101000,
B00100000,
B00000000,
6,

B00000000,   //+
B00100000,
B00100000,
B11111000,
B00100000,
```

```
B00100000,
B00000000,
6,

B00000000,   //,
B00000000,
B00000000,
B00000000,
B11000000,
B01000000,
B10000000,
3,

B00000000,   //-
B00000000,
B11111000,
B00000000,
B00000000,
B00000000,
B00000000,
6,

B00000000,   //.
B00000000,
B00000000,
B00000000,
B00000000,
B11000000,
B11000000,
3,

B00000000,   ///
B00001000,
B00010000,
B00100000,
B01000000,
B10000000,
B00000000,
6,

B01110000,   //0
B10001000,
B10011000,
B10101000,
B11001000,
B10001000,
B01110000,
6,

B01000000,   //1
B11000000,
B01000000,
```

```
B01000000,
B01000000,
B01000000,
B11100000,
4,

B01110000,   //2
B10001000,
B00001000,
B00010000,
B00100000,
B01000000,
B11111000,
6,

B11111000,   //3
B00010000,
B00100000,
B00010000,
B00001000,
B10001000,
B01110000,
6,

B00010000,   //4
B00110000,
B01010000,
B10010000,
B11111000,
B00010000,
B00010000,
6,

B11111000,   //5
B10000000,
B11110000,
B00001000,
B00001000,
B10001000,
B01110000,
6,

B00110000,   //6
B01000000,
B10000000,
B11110000,
B10001000,
B10001000,
B01110000,
6,

B11111000,   //7
```

```
B10001000,
B00001000,
B00010000,
B00100000,
B00100000,
B00100000,
6,

B01110000,   //8
B10001000,
B10001000,
B01110000,
B10001000,
B10001000,
B01110000,
6,

B01110000,   //9
B10001000,
B10001000,
B01111000,
B00001000,
B00010000,
B01100000,
6,

B00000000,   //:
B11000000,
B11000000,
B00000000,
B11000000,
B11000000,
B00000000,
3,

B00000000,   //;
B11000000,
B11000000,
B00000000,
B11000000,
B01000000,
B10000000,
3,

B00010000,   //<
B00100000,
B01000000,
B10000000,
B01000000,
B00100000,
B00010000,
5,
```

```
B00000000,  //=
B00000000,
B11111000,
B00000000,
B11111000,
B00000000,
B00000000,
6,

B10000000,  //>
B01000000,
B00100000,
B00010000,
B00100000,
B01000000,
B10000000,
5,

B01110000,  //?
B10001000,
B00001000,
B00010000,
B00100000,
B00000000,
B00100000,
6,

B01110000,  //@
B10001000,
B00001000,
B01101000,
B10101000,
B10101000,
B01110000,
6,

B01110000,  //A
B10001000,
B10001000,
B10001000,
B11111000,
B10001000,
B10001000,
6,

B11110000,  //B
B10001000,
B10001000,
B11110000,
B10001000,
B10001000,
```

```
B11110000,
6,

B01110000,   //C
B10001000,
B10000000,
B10000000,
B10000000,
B10001000,
B01110000,
6,

B11100000,   //D
B10010000,
B10001000,
B10001000,
B10001000,
B10010000,
B11100000,
6,

B11111000,   //E
B10000000,
B10000000,
B11110000,
B10000000,
B10000000,
B11111000,
6,

B11111000,   //F
B10000000,
B10000000,
B11110000,
B10000000,
B10000000,
B10000000,
6,

B01110000,   //G
B10001000,
B10000000,
B10111000,
B10001000,
B10001000,
B01111000,
6,

B10001000,   //H
B10001000,
B10001000,
B11111000,
```

```
B10001000,
B10001000,
B10001000,
6,

B11100000,  //I
B01000000,
B01000000,
B01000000,
B01000000,
B01000000,
B11100000,
4,

B00111000,  //J
B00010000,
B00010000,
B00010000,
B00010000,
B10010000,
B01100000,
6,

B10001000,  //K
B10010000,
B10100000,
B11000000,
B10100000,
B10010000,
B10001000,
6,

B10000000,  //L
B10000000,
B10000000,
B10000000,
B10000000,
B10000000,
B11111000,
6,

B10001000,  //M
B11011000,
B10101000,
B10101000,
B10001000,
B10001000,
B10001000,
6,

B10001000,  //N
B10001000,
```

```
B11001000,
B10101000,
B10011000,
B10001000,
B10001000,
6,

B01110000,   //O
B10001000,
B10001000,
B10001000,
B10001000,
B10001000,
B01110000,
6,

B11110000,   //P
B10001000,
B10001000,
B11110000,
B10000000,
B10000000,
B10000000,
6,

B01110000,   //Q
B10001000,
B10001000,
B10001000,
B10101000,
B10010000,
B01101000,
6,

B11110000,   //R
B10001000,
B10001000,
B11110000,
B10100000,
B10010000,
B10001000,
6,

B01111000,   //S
B10000000,
B10000000,
B01110000,
B00001000,
B00001000,
B11110000,
6,
```

```
B11111000,  //T
B00100000,
B00100000,
B00100000,
B00100000,
B00100000,
B00100000,
6,

B10001000,  //U
B10001000,
B10001000,
B10001000,
B10001000,
B10001000,
B01110000,
6,

B10001000,  //V
B10001000,
B10001000,
B10001000,
B10001000,
B01010000,
B00100000,
6,

B10001000,  //W
B10001000,
B10001000,
B10101000,
B10101000,
B10101000,
B01010000,
6,

B10001000,  //X
B10001000,
B01010000,
B00100000,
B01010000,
B10001000,
B10001000,
6,

B10001000,  //Y
B10001000,
B10001000,
B01010000,
B00100000,
B00100000,
B00100000,
```

```
6,

B11111000,  //Z
B00001000,
B00010000,
B00100000,
B01000000,
B10000000,
B11111000,
6,

B11100000,  //[
B10000000,
B10000000,
B10000000,
B10000000,
B10000000,
B11100000,
4,

B00000000,  //(Backward Slash)
B10000000,
B01000000,
B00100000,
B00010000,
B00001000,
B00000000,
6,

B11100000,  //]
B00100000,
B00100000,
B00100000,
B00100000,
B00100000,
B11100000,
4,

B00100000,  //^
B01010000,
B10001000,
B00000000,
B00000000,
B00000000,
B00000000,
6,

B00000000,  //_
B00000000,
B00000000,
B00000000,
B00000000,
```

```
B00000000,
B11111000,
6,

B10000000,  //`
B01000000,
B00100000,
B00000000,
B00000000,
B00000000,
B00000000,
4,

B00000000,  //a
B00000000,
B01110000,
B00001000,
B01111000,
B10001000,
B01111000,
6,

B10000000,  //b
B10000000,
B10110000,
B11001000,
B10001000,
B10001000,
B11110000,
6,

B00000000,  //c
B00000000,
B01110000,
B10001000,
B10000000,
B10001000,
B01110000,
6,

B00001000,  //d
B00001000,
B01101000,
B10011000,
B10001000,
B10001000,
B01111000,
6,

B00000000,  //e
B00000000,
B01110000,
```

```
B10001000,
B11111000,
B10000000,
B01110000,
6,

B00110000,  //f
B01001000,
B01000000,
B11100000,
B01000000,
B01000000,
B01000000,
6,

B00000000,  //g
B01111000,
B10001000,
B10001000,
B01111000,
B00001000,
B01110000,
6,

B10000000,  //h
B10000000,
B10110000,
B11001000,
B10001000,
B10001000,
B10001000,
6,

B01000000,  //i
B00000000,
B11000000,
B01000000,
B01000000,
B01000000,
B11100000,
4,

B00010000,  //j
B00000000,
B00110000,
B00010000,
B00010000,
B10010000,
B01100000,
5,

B10000000,  //k
```

```
B10000000,
B10010000,
B10100000,
B11000000,
B10100000,
B10010000,
5,

B11000000,   //l
B01000000,
B01000000,
B01000000,
B01000000,
B01000000,
B11100000,
4,

B00000000,   //m
B00000000,
B11010000,
B10101000,
B10101000,
B10001000,
B10001000,
6,

B00000000,   //n
B00000000,
B10110000,
B11001000,
B10001000,
B10001000,
B10001000,
6,

B00000000,   //o
B00000000,
B01110000,
B10001000,
B10001000,
B10001000,
B01110000,
6,

B00000000,   //p
B00000000,
B11110000,
B10001000,
B11110000,
B10000000,
B10000000,
6,
```

```
B00000000,  //q
B00000000,
B01101000,
B10011000,
B01111000,
B00001000,
B00001000,
6,

B00000000,  //r
B00000000,
B10110000,
B11001000,
B10000000,
B10000000,
B10000000,
6,

B00000000,  //s
B00000000,
B01110000,
B10000000,
B01110000,
B00001000,
B11110000,
6,

B01000000,  //t
B01000000,
B11100000,
B01000000,
B01000000,
B01001000,
B00110000,
6,

B00000000,  //u
B00000000,
B10001000,
B10001000,
B10001000,
B10011000,
B01101000,
6,

B00000000,  //v
B00000000,
B10001000,
B10001000,
B10001000,
B01010000,
```

```
B00100000,
6,

B00000000,  //w
B00000000,
B10001000,
B10101000,
B10101000,
B10101000,
B01010000,
6,

B00000000,  //x
B00000000,
B10001000,
B01010000,
B00100000,
B01010000,
B10001000,
6,

B00000000,  //y
B00000000,
B10001000,
B10001000,
B01111000,
B00001000,
B01110000,
6,

B00000000,  //z
B00000000,
B11111000,
B00010000,
B00100000,
B01000000,
B11111000,
6,

B00100000,  //{
B01000000,
B01000000,
B10000000,
B01000000,
B01000000,
B00100000,
4,

B10000000,  //|
B10000000,
B10000000,
B10000000,
```

```
    B10000000,
    B10000000,
    B10000000,
    2,

    B10000000,   //}
    B01000000,
    B01000000,
    B00100000,
    B01000000,
    B01000000,
    B10000000,
    4,

    B11100000,   //~ -->°
    B10100000,
    B11100000,
    B00000000,
    B00000000,
    B00000000,
    B00000000,
    4,

    B01100000,   // (Char 0x7F)
    B10010000,
    B10010000,
    B01100000,
    B00000000,
    B00000000,
    B00000000,
    5
};

/*void scrollFont() {
    for (int counter=0x20;counter<0x80;counter++){
        loadBufferLong(counter);
        delay(scrollDelay);
    }
}*/

// Scroll Message
void scrollMessage(char * messageString) {
    int counter = 0;
    int myChar=0;
    do {
        // read back a char
        myChar =  *(messageString + counter);
        if (myChar != 0){
            loadBufferLong(myChar);
        }
        counter++;
    }
```

```
  while (myChar != 0);
}

// Scroll Number
void scrollNumber(char * messageString, int m) {
  int myChar=0;
  for(int i=0;i<m;i++) {
    // read back a char
    myChar =  *(messageString + i); ;
    if (myChar != 0){
      loadBufferLong(myChar);
    }
  }
}

// Load character into scroll buffer
void loadBufferLong(int ascii){
  if (ascii >= 0x20 && ascii <=0x7f){
    for (int a=0;a<7;a++){                    // Loop 7 times for a 5x7 font
      unsigned long c = pgm_read_byte_near(font5x7 + ((ascii - 0x20) * 8) + a);     // Index into character table
to get row data
      unsigned long x = bufferLong [a*2];     // Load current scroll buffer
      x = x | c;                              // OR the new character onto end of current
      bufferLong [a*2] = x;                   // Store in buffer
    }
    byte count = pgm_read_byte_near(font5x7 +((ascii - 0x20) * 8) + 7);     // Index into character table for
kerning data
    for (byte x=0; x<count;x++){
      rotateBufferLong();
      printBufferLong();
      delay(scrollDelay);
    }
  }
}
// Rotate the buffer
void rotateBufferLong(){
  for (int a=0;a<7;a++){                    // Loop 7 times for a 5x7 font
    unsigned long x = bufferLong [a*2];     // Get low buffer entry
    byte b = bitRead(x,31);                 // Copy high order bit that gets lost in rotation
    x = x<<1;                               // Rotate left one bit
    bufferLong [a*2] = x;                   // Store new low buffer
    x = bufferLong [a*2+1];                 // Get high buffer entry
    x = x<<1;                               // Rotate left one bit
    bitWrite(x,0,b);                        // Store saved bit
    bufferLong [a*2+1] = x;                 // Store new high buffer
  }
}
// Display Buffer on LED matrix
void printBufferLong(){
 for (int a=0;a<7;a++){                     // Loop 7 times for a 5x7 font
  unsigned long x = bufferLong [a*2+1];   // Get high buffer entry
  byte y = x;                             // Mask off first character
```

```
    lc.setColumn(3,7-a,y);                  // Send row to relevent MAX7219 chip
    x = bufferLong [a*2];                   // Get low buffer entry
    y = (x>>24);                            // Mask off second character
    lc.setColumn(2,7-a,y);                  // Send row to relevent MAX7219 chip
    y = (x>>16);                            // Mask off third character
    lc.setColumn(1,7-a,y);                  // Send row to relevent MAX7219 chip
    y = (x>>8);                             // Mask off forth character
    lc.setColumn(0,7-a,y);                  // Send row to relevent MAX7219 chip
  }
}

void floattochar(float x,char *num)
{
  float y;
  int n,a;
  char b;
  n=1+log10(x);
  int num2[n];
  //num=(char*)malloc((n+2)*sizeof(char));

  for (int j=0;j<n;j++)
  {
    num2[j]=0;
  }

  for(int i=0;i<n;i++)
  {
    y=x;
    for(int j=0;j<n;j++)
      {
        y-=num2[j]*pow(10,(n-j-1));
      }

    if(n-i-1 != 0)
    {
      a=int(y/(pow(10,(n-i-1))));
      b=char(a);
      b+=0x30;
      num[i]=b;
      num2[i]=a;
    }
    if(n-i-1 == 0)
    {
      a=int(y);
      b=char(a);
      b+=0x30;
      num[i]=b;
      num2[i]=a;
    }
  }

  num[n]='.';
```

```
  y=x;
  for(int j=0;j<n;j++)
    {
      y-=num2[j]*pow(10,(n-j-1));
    }
  y=10*y+0.01;
  a=int(y);
  b=char(a);
  b+=0x30;
  num[n+1]=b;
  num2[n+1]=a;

  //Serial.print(num);
  //Serial.println(" ");
}

void inttochar(int x,char *num)
{
  int y;
  int n,a;
  char b;
  n=1+log10(x);
  int num2[n];
  //num=(char*)malloc((n+2)*sizeof(char));

  for (int j=0;j<n;j++)
  {
    num2[j]=0;
  }

  for(int i=0;i<n;i++)
  {
    y=x;
    for(int j=0;j<n;j++)
      {
        y-=num2[j]*pow(10,(n-j-1));
      }

    if(n-i-1 != 0)
    {
      a=int(y/(pow(10,(n-i-1))));
      b=char(a);
      b+=0x30;
      num[i]=b;
      num2[i]=a;
    }
    if(n-i-1 == 0)
    {
      a=int(y);
      b=char(a);
      b+=0x30;
      num[i]=b;
```

```
      num2[i]=a;
    }
  }

  //Serial.print(num);
  //Serial.println(" ");
}
```

ANEXO 2

```
/*
Pantalla táctil capacitiva de 240x320 píxeles (en x,y respectivamente).
El eje de coordenadas es el que queda junto a la clavija de alimentación externa.
Se considera positivo el sentido hacia la derecha y hacia abajo (en x,y respectivamente).
*/

#include "Arduino.h"
#include "PantCapacitiva.h"
#include "Adafruit_FT6206.h"
#include "Adafruit_GFX.h"
#include "Adafruit_ILI9341.h"
#include "SPI.h"
#include "Wire.h"

// Use hardware SPI (on Uno, #13, #12, #11) and the above for CS/DC
Adafruit_ILI9341 tft = Adafruit_ILI9341(10,9);
Adafruit_FT6206 ctp = Adafruit_FT6206();

PantCapacitiva::PantCapacitiva()
{
}

void PantCapacitiva::Inicia()   //Inicializa la pantalla táctil
{
   //Adafruit_ILI9341 tft = Adafruit_ILI9341(10,9);
   //Adafruit_FT6206 ctp = Adafruit_FT6206();
   tft.begin();
   if (! ctp.begin(40))
   {  // pass in 'sensitivity' coefficient
      Serial.println("Couldn't start FT6206 touchscreen controller");
      while (1);
   }
   else
   {
      Serial.println("¡INICIADA!");
   }
}

void PantCapacitiva::PintaPantalla(int color)   //Pinta la pantalla del color que se lo introduce en el argumento
(16 bits)
{
   tft.fillScreen(color);
}

void PantCapacitiva::DibujaPixel(int x,int y,int color) //Dibuja un pixel del color seleccionado en la coordenada
x(entre 0 y 239), y(entre 0 y 319)
{
   tft.drawPixel(x,y,color);
}
```

```
void PantCapacitiva::DibujaLinea(int x0,int y0,int xf,int yf,int color) //Dibuja una línea del color introducido desde x0,y0 hasta xf,yf
{
  tft.drawLine(x0,y0,xf,yf,color);
}

void PantCapacitiva::DibujaCirculo(int x,int y,int radio,int color,bool modo)  //Dibuja un círculo del color selecionado centrado en x,y y con el radio introducido,. Si modo = 0 no pintará el interior pero si modo = 1 sí que lo hará
{
  if(modo==0)
  {
    tft.drawCircle(x,y,radio,color);
  }
  if(modo==1)
  {
    tft.fillCircle(x,y,radio,color);
  }
}

void PantCapacitiva::DibujaRectangulo(int x,int y,int anchura,int altura,int color,bool modo)  //Dibuja un rectángulo del color seleccionado con el vértice en x,y y la anchura y altura introducidas. Si modo = 0 no lo rellenará y si modo = 1 sí lo hará
{
  if(modo==0)
  {
    tft.drawRect(x,y,anchura,altura,color);
  }
  if(modo==1)
  {
    tft.fillRect(x,y,anchura,altura,color);
  }
}

void PantCapacitiva::DibujaTriangulo(int x1,int y1,int x2,int y2,int x3,int y3,int color,bool modo)  //Dibuja un triángulo del color seleccionado introduciendo sus 3 vértices. Si modo = 0 no lo rellenará y si modo = 1 sí lo hará
{
  if(modo==0)
  {
    tft.drawTriangle(x1,y1,x2,y2,x3,y3,color);
  }
  if(modo==1)
  {
    tft.fillTriangle(x1,y1,x2,y2,x3,y3,color);
  }
}

void PantCapacitiva::RotaPantalla(int R)    //Rota la pantalla: 0 - Modo normal,    1 - 90º,    2 - 180º,    3 - 270º
{
  tft.setRotation(R);
}
```

```
void PantCapacitiva::EscribeTexto(int x0,int y0,bool sub,int color,int color2,int tam,bool salto,char arg[])
//Escribe un texto desde el punto x0,y0 (vértice superior izquierdo del texto). Si sub vale 1 el texto se resaltará con un color de forndo (color 2) y si vale cero no lo hará. El tamañano debe ser un número entero (entre 1 y 6). Si eliges salto igual a cero si el texto no cabe en una linea se cortará, pero si lo eliges igual a 1 saltará a la siguiente línea. El último argumento es el propio texto
{
  tft.setCursor(x0,y0);

  if(sub==0)
  {
    tft.setTextColor(color);
  }
  if(sub==1)
  {
    tft.setTextColor(color,color2);
  }

  tft.setTextSize(tam);

  if(salto==0)
  {
    tft.setTextWrap(false);
  }
  if(salto==1)
  {
    tft.setTextWrap(true);
  }

  tft.print(arg);
}

void PantCapacitiva::EscribeTexto(int x0,int y0,bool sub,int color,int color2,int tam,bool salto,int arg)
//Escribe un texto desde el punto x0,y0 (vértice superior izquierdo del texto). Si sub vale 1 el texto se resaltará con un color de forndo (color 2) y si vale cero no lo hará. El tamañano debe ser un número entero (entre 1 y 6). Si eliges salto igual a cero si el texto no cabe en una linea se cortará, pero si lo eliges igual a 1 saltará a la siguiente línea. El último argumento es el propio texto
{
  tft.setCursor(x0,y0);

  if(sub==0)
  {
    tft.setTextColor(color);
  }
  if(sub==1)
  {
    tft.setTextColor(color,color2);
  }

  tft.setTextSize(tam);

  if(salto==0)
```

```
    {
      tft.setTextWrap(false);
    }
    if(salto==1)
    {
      tft.setTextWrap(true);
    }

    tft.print(arg);
}

void PantCapacitiva::EscribeTexto(int x0,int y0,bool sub,int color,int color2,int tam,bool salto,double arg)
//Escribe un texto desde el punto x0,y0 (vértice superior izquierdo del texto). Si sub vale 1 el texto se resaltará con un color de forndo (color 2) y si vale cero no lo hará. El tamañano debe ser un número entero (entre 1 y 6). Si eliges salto igual a cero si el texto no cabe en una linea se cortará, pero si lo eliges igual a 1 saltará a la siguiente línea. El último argumento es el propio texto
{
    tft.setCursor(x0,y0);

    if(sub==0)
    {
      tft.setTextColor(color);
    }
    if(sub==1)
    {
      tft.setTextColor(color,color2);
    }

    tft.setTextSize(tam);

    if(salto==0)
    {
      tft.setTextWrap(false);
    }
    if(salto==1)
    {
      tft.setTextWrap(true);
    }

    tft.print(arg,1);
}

void PantCapacitiva::EscribeTexto(int x0,int y0,bool sub,int color,int color2,int tam,bool salto,float arg)
//Escribe un texto desde el punto x0,y0 (vértice superior izquierdo del texto). Si sub vale 1 el texto se resaltará con un color de forndo (color 2) y si vale cero no lo hará. El tamañano debe ser un número entero (entre 1 y 6). Si eliges salto igual a cero si el texto no cabe en una linea se cortará, pero si lo eliges igual a 1 saltará a la siguiente línea. El último argumento es el propio texto
{
    tft.setCursor(x0,y0);

    if(sub==0)
    {
```

```
        tft.setTextColor(color);
    }
    if(sub==1)
    {
        tft.setTextColor(color,color2);
    }

    tft.setTextSize(tam);

    if(salto==0)
    {
        tft.setTextWrap(false);
    }
    if(salto==1)
    {
        tft.setTextWrap(true);
    }

    tft.print(arg,1);
}

void PantCapacitiva::ObtenPuntoTactil (int *x,int *y,int R)
{
    TS_Point p;

    // Espera a que la pantalla sea tocada
    if (! ctp.touched())
    {
        //Serial.println("Esperando...");
        return;
    }

    // Obten un punto
    p = ctp.getPoint();
    //Serial.print(p.x); Serial.print(", ");
    //Serial.print(p.y); Serial.print("\n");

    //Lo ajusto a la rotación de la pantalla
    if (R==0)
    {
        *x = map(p.x, 0, 240, 240, 0);
        *y = map(p.y, 0, 320, 320, 0);
    }
    if (R==1) //Lo ajusto a la rotación de la pantalla
    {
        *x = map(p.y, 0, 320, 0, 320);
        *y = map(p.x, 0, 240, 240, 0);
    }
    if (R==2) //Lo ajusto a la rotación de la pantalla
    {
        *x = map(p.x, 0, 240, 0, 240);
        *y = map(p.y, 0, 320, 0, 320);
```

```
    }
    if (R==3) //Lo ajusto a la rotación de la pantalla
    {
       *x = map(p.y, 0, 320, 320, 0);
       *y = map(p.x, 0, 240, 0, 240);
    }

    //Serial.print(*x); Serial.print(", ");
    //Serial.print(*y); Serial.print("\n");
}
```

ANEXO 3

Programa DEMO

```
#include "Wire.h"
#include "Adafruit_FT6206.h"
#include "SPI.h"
#include "PantCapacitiva.h"
#include "Adafruit_ILI9341.h"
#include "Adafruit_GFX.h"
#include "Demo.h"
#include "DHT.h"
#include "SFE_BMP180.h"

Demo d;

int x,y;
int j=0;

void setup()
{
  Serial.begin(115200);
  d.Inicia();
  d.Menu();
}

void loop()
{
  if(j==0)
  {
    j=d.SeleccionaMenu(&x,&y);
  }
  if(j==1)
  {
    j=d.SeleccionaTemperatura(&x,&y);
  }
  if(j==2)
  {
    j=d.SeleccionaPresion(&x,&y);
  }
  if(j==3)
  {
    j=d.SeleccionaHumedad(&x,&y);
  }
  if(j==4)
  {
    j=d.SeleccionaAltitud(&x,&y);
  }
}
```

Librería DEMO

```
#include "Arduino.h"
#include "Adafruit_FT6206.h"
#include "Adafruit_GFX.h"
#include "Adafruit_ILI9341.h"
#include "SPI.h"
#include "Wire.h"
#include "PantCapacitiva.h"
#include "Demo.h"
#include "DHT.h"
#include "SFE_BMP180.h"

#define DHTPIN 19 //Pin al que se conecta el sensor
#define DHTTYPE DHT22 // Tipo de sensor
#define ALTITUD 240   // Altitud de Zaragoza

float humedad;    // % humedad (DHT22)
float temp_DHT;   // temperatura (DHT22 , ºC)
float temp_BM180; // temperatura (BMP180 , ºC)
float presion;    // presión (BMP180)
double T0,P0;
bool b=0;   //Para seleccionar ºC o K

PantCapacitiva pru;
DHT dht(DHTPIN, DHTTYPE); // Inicialización del sensor DHT
SFE_BMP180 BMP180;  //Sensor BMP

Demo::Demo()
{
}

void Demo::Inicia()
{
  char status;
  double T,P;
  pru.Inicia();
  BMP180.begin();
  status=BMP180.startTemperature();
  delay(status);
  status=BMP180.getTemperature(T);

  status=BMP180.startPressure(3);
  delay(status);
  status=BMP180.getPressure(P,T);
  T0=T;
  P0=P;
}

void Demo::Menu()
{
  pru.PintaPantalla(Negro);
```

```
  pru.EscribeTexto(15,15,0,0xEE8A,Azul,3,0,"Elige Sensor");
  pru.DibujaRectangulo(0,48,240,4,Rojo,1);
  pru.DibujaRectangulo(30,72,180,42,0xEE8A,1);
  pru.DibujaRectangulo(30,134,180,42,0xEE8A,1);
  pru.DibujaRectangulo(30,196,180,42,0xEE8A,1);
  pru.DibujaRectangulo(30,258,180,42,0xEE8A,1);
  pru.EscribeTexto(54,85,0,Rojo,Azul,2,0,"TEMPERATURA");
  pru.EscribeTexto(75,147,0,Rojo,Azul,2,0,"PRESION");
  pru.EscribeTexto(75,209,0,Rojo,Azul,2,0,"HUMEDAD");
  pru.EscribeTexto(75,271,0,Rojo,Azul,2,0,"ALTITUD");
}

void Demo::Temperatura()
{
  pru.PintaPantalla(Negro);
  pru.EscribeTexto(25,15,0,0xEE8A,Azul,3,0,"TEMPERATURA");
  pru.DibujaRectangulo(0,48,240,4,Rojo,1);
  //pru.EscribeTexto(80,85,0,0xEE8A,Azul,6,0,"22");
  //pru.EscribeTexto(154,80,0,0xEE8A,Azul,2,0,"o");
  pru.DibujaCirculo(50,270,30,Rojo,1);
  pru.DibujaTriangulo(30,270,45,255,45,285,0xEE8A,1);
  pru.DibujaRectangulo(45,262,25,16,0xEE8A,1);
  pru.DibujaRectangulo(120,270,80,45,0xEE8A,1);
  pru.DibujaRectangulo(120,215,80,45,0xEE8A,1);
  pru.EscribeTexto(140,217,0,Rojo,Azul,2,0,"o");
  pru.EscribeTexto(155,221,0,Rojo,Azul,5,0,"C");
  pru.EscribeTexto(148,276,0,Rojo,Azul,5,0,"K");
}

void Demo::Presion()
{
  pru.PintaPantalla(Negro);
  pru.EscribeTexto(60,15,0,0xEE8A,Azul,3,0,"PRESION");
  pru.DibujaRectangulo(0,48,240,4,Rojo,1);
  //pru.EscribeTexto(15,85,0,0xEE8A,Azul,6,0,"1.0");
  //pru.EscribeTexto(125,85,0,0xEE8A,Azul,6,0,"atm");
  pru.DibujaCirculo(50,270,30,Rojo,1);
  pru.DibujaTriangulo(30,270,45,255,45,285,0xEE8A,1);
  pru.DibujaRectangulo(45,262,25,16,0xEE8A,1);
}

void Demo::Humedad()
{
  pru.PintaPantalla(Negro);
  pru.EscribeTexto(60,15,0,0xEE8A,Azul,3,0,"HUMEDAD");
  pru.DibujaRectangulo(0,48,240,4,Rojo,1);
  //pru.EscribeTexto(70,85,0,0xEE8A,Azul,6,0,"23");
  //pru.EscribeTexto(150,85,0,0xEE8A,Azul,6,0,"%");
  pru.DibujaCirculo(50,270,30,Rojo,1);
  pru.DibujaTriangulo(30,270,45,255,45,285,0xEE8A,1);
  pru.DibujaRectangulo(45,262,25,16,0xEE8A,1);
}
```

```
void Demo::Altitud()
{
  pru.PintaPantalla(Negro);
  pru.EscribeTexto(60,15,0,0xEE8A,Azul,3,0,"ALTITUD");
  pru.DibujaRectangulo(0,48,240,4,Rojo,1);
  //pru.EscribeTexto(60,85,0,0xEE8A,Azul,6,0,"--");
  //pru.EscribeTexto(150,85,0,0xEE8A,Azul,6,0,"m");
  pru.DibujaCirculo(50,270,30,Rojo,1);
  pru.DibujaTriangulo(30,270,45,255,45,285,0xEE8A,1);
  pru.DibujaRectangulo(45,262,25,16,0xEE8A,1);
}

int Demo::SeleccionaMenu(int *x,int *y)
{
  pru.ObtenPuntoTactil(x,y,0);
  if(*x>30 & *x<210)
  {
    if(*y>72 & *y<114)
    {
      Temperatura();
      *x=0;
      *y=0;
      return(1);
    }
    if(*y>134 & *y<172)
    {
      Presion();
      *x=0;
      *y=0;
      return(2);
    }
    if(*y>196 & *y<238)
    {
      Humedad();
      *x=0;
      *y=0;
      return(3);
    }
    if(*y>258 & *y<300)
    {
      Altitud();
      *x=0;
      *y=0;
      return(4);
    }
  }
  return(0);
}

int Demo::SeleccionaTemperatura(int *x,int *y)
{
```

```
  temp_DHT = dht.readTemperature();
  if(b==0)
  {
    pru.DibujaRectangulo(0,54,240,100,Negro,1);
    pru.EscribeTexto(40,85,0,0xEE8A,Azul,6,0,temp_DHT);
    pru.EscribeTexto(184,80,0,0xEE8A,Azul,2,0,"o");
  }
  if(b==1)
  {
    pru.DibujaRectangulo(0,54,240,100,Negro,1);
    pru.EscribeTexto(40,85,0,0xEE8A,Azul,6,0,int(temp_DHT+273.5));
    pru.EscribeTexto(170,85,0,0xEE8A,Azul,6,0,"K");
  }
  for (int i=0;i<4300;i++) //A mayor número de pasos, más tiempo pasará entre las medidas. Delay casero
  {
    pru.ObtenPuntoTactil(x,y,0);
    if(*x>20 & *x<80 & *y>240 & *y<300)
    {
      Menu();
      *x=0;
      *y=0;
      return(0);
    }
    if(*x>120 & *x<200 & *y>215 & *y<260)
    {
      b=0;
      *x=0;
      *y=0;
      return(1);
    }
    if(*x>120 & *x<200 & *y>270 & *y<315)
    {
      b=1;
      *x=0;
      *y=0;
      return(1);
    }
  }
  return(1);
}

int Demo::SeleccionaPresion(int *x,int *y)
{
  char status;
  double P,T;
  int i;
  for (i=0;i<4300;i++)
  {
    pru.ObtenPuntoTactil(x,y,0);
    if(i==1)
    {
      status=BMP180.startTemperature();
```

```
      delay(status);
      status=BMP180.getTemperature(T);
      status=BMP180.startPressure(3);
      delay(status);
      status=BMP180.getPressure(P,T);
      pru.DibujaRectangulo(0,54,240,100,Negro,1);
      pru.EscribeTexto(15,85,0,0xEE8A,Azul,6,0,int(P+0.5));
      pru.EscribeTexto(125,85,0,0xEE8A,Azul,6,0,"hPa");
    }
    if(*x>20 & *x<80 & *y>240 & *y<300)
    {
      Menu();
      *x=0;
      *y=0;
      return(0);
    }
  }
  return(2);
}

int Demo::SeleccionaHumedad(int *x,int *y)
{
  int i;
  for (i=0;i<4300;i++)
  {
    pru.ObtenPuntoTactil(x,y,0);
    if(i==1)
    {
      humedad = dht.readHumidity();
      pru.DibujaRectangulo(0,54,240,100,Negro,1);
      pru.EscribeTexto(70,85,0,0xEE8A,Azul,6,0,int(humedad));
      pru.EscribeTexto(150,85,0,0xEE8A,Azul,6,0,"%");
    }
    if(*x>20 & *x<80 & *y>240 & *y<300)
    {
      Menu();
      *x=0;
      *y=0;
      return(0);
    }
  }
  return(3);
}

int Demo::SeleccionaAltitud(int *x, int *y)
{
  char status;
  double P,T,aR;
  int i;
  for (i=0;i<4300;i++)
  {
    pru.ObtenPuntoTactil(x,y,0);
```

```
    if(i==1)
    {
      status=BMP180.startTemperature();
      delay(status);
      status=BMP180.getTemperature(T);
      status=BMP180.startPressure(3);
      delay(status);
      status=BMP180.getPressure(P,T);
      aR=BMP180.altitude(P,P0);
      pru.DibujaRectangulo(0,54,240,100,Negro,1);
      pru.EscribeTexto(40,85,0,0xEE8A,Azul,6,0,int(ALTITUD+aR+0.5));//Aquí va la medida
      pru.EscribeTexto(170,85,0,0xEE8A,Azul,6,0,"m");
    }
    if(*x>20 & *x<80 & *y>240 & *y<300)
    {
      Menu();
      *x=0;
      *y=0;
      return(0);
    }
  }
  return(4);
}
```

Anexo 4

19-4452; Rev 4; 7/03

MAXIM

Serially Interfaced, 8-Digit LED Display Drivers

General Description

The MAX7219/MAX7221 are compact, serial input/output common-cathode display drivers that interface microprocessors (µPs) to 7-segment numeric LED displays of up to 8 digits, bar-graph displays, or 64 individual LEDs. Included on-chip are a BCD code-B decoder, multiplex scan circuitry, segment and digit drivers, and an 8x8 static RAM that stores each digit. Only one external resistor is required to set the segment current for all LEDs. The MAX7221 is compatible with SPI™, QSPI™, and MICROWIRE™, and has slew-rate-limited segment drivers to reduce EMI.

A convenient 4-wire serial interface connects to all common µPs. Individual digits may be addressed and updated without rewriting the entire display. The MAX7219/MAX7221 also allow the user to select code-B decoding or no-decode for each digit.

The devices include a 150µA low-power shutdown mode, analog and digital brightness control, a scan-limit register that allows the user to display from 1 to 8 digits, and a test mode that forces all LEDs on.

For applications requiring 3V operation or segment blinking, refer to the MAX6951 data sheet.

Applications

Bar-Graph Displays
Industrial Controllers
Panel Meters
LED Matrix Displays

Features

- **10MHz Serial Interface**
- **Individual LED Segment Control**
- **Decode/No-Decode Digit Selection**
- **150µA Low-Power Shutdown (Data Retained)**
- **Digital and Analog Brightness Control**
- **Display Blanked on Power-Up**
- **Drive Common-Cathode LED Display**
- **Slew-Rate Limited Segment Drivers for Lower EMI (MAX7221)**
- **SPI, QSPI, MICROWIRE Serial Interface (MAX7221)**
- **24-Pin DIP and SO Packages**

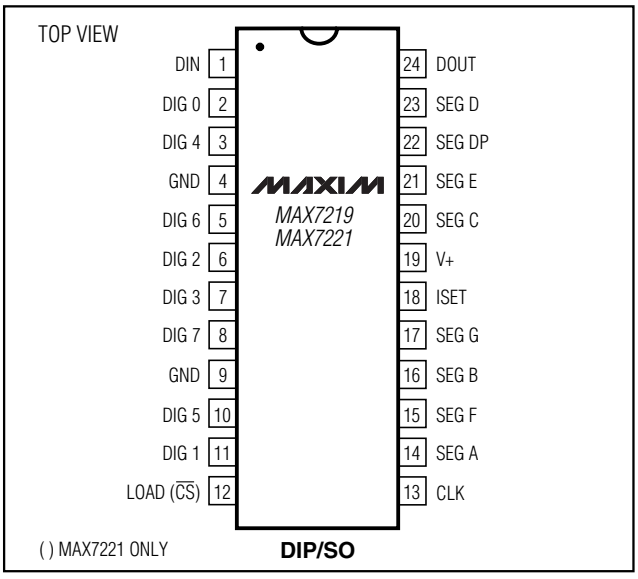
Ordering Information

PART	TEMP RANGE	PIN-PACKAGE
MAX7219CNG	0°C to +70°C	24 Narrow Plastic DIP
MAX7219CWG	0°C to +70°C	24 Wide SO
MAX7219C/D	0°C to +70°C	Dice*
MAX7219ENG	-40°C to +85°C	24 Narrow Plastic DIP
MAX7219EWG	-40°C to +85°C	24 Wide SO
MAX7219ERG	-40°C to +85°C	24 Narrow CERDIP

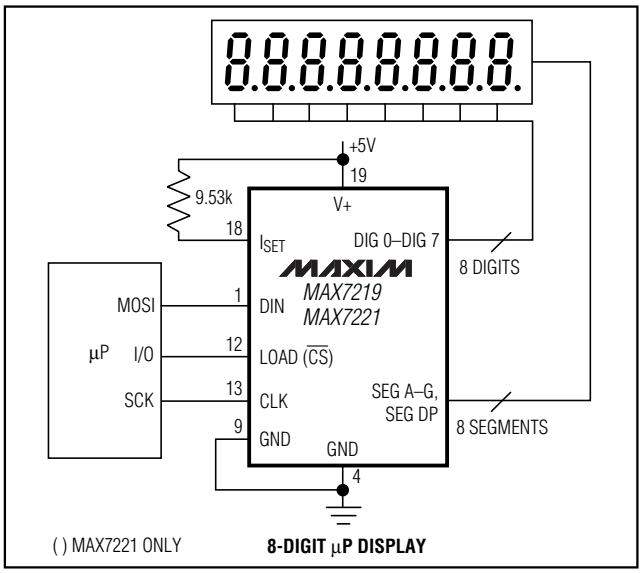
Ordering Information continued at end of data sheet.

**Dice are specified at T_A = +25°C.*

Pin Configuration



Typical Application Circuit



SPI and QSPI are trademarks of Motorola Inc. MICROWIRE is a trademark of National Semiconductor Corp.

ABSOLUTE MAXIMUM RATINGS

Voltage (with respect to GND)
- V+ .....-0.3V to 6V
- DIN, CLK, LOAD, $\overline{CS}$ .....-0.3V to 6V
- All Other Pins.....-0.3V to (V+ + 0.3V)

Current
- DIG0–DIG7 Sink Current.....500mA
- SEGA–G, DP Source Current.....100mA

Continuous Power Dissipation (T_A = +85°C)
- Narrow Plastic DIP (derate 13.3mW/°C above +70°C).....1066mW
- Wide SO (derate 11.8mW/°C above +70°C).....941mW
- Narrow CERDIP (derate 12.5mW/°C above +70°C) ...1000mW

Operating Temperature Ranges (T_{MIN} to T_{MAX})
- MAX7219C_G/MAX7221C_G .....0°C to +70°C
- MAX7219E_G/MAX7221E_G .....-40°C to +85°C

Storage Temperature Range .....-65°C to +160°C

Lead Temperature (soldering, 10s) .....+300°C

Stresses beyond those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated in the operational sections of the specifications is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

ELECTRICAL CHARACTERISTICS

(V+ = 5V ±10%, R_{SET} = 9.53kΩ ±1%, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
Operating Supply Voltage	V+		4.0		5.5	V
Shutdown Supply Current	I+	All digital inputs at V+ or GND, T_A = +25°C			150	µA
Operating Supply Current	I+	R_{SET} = open circuit			8	mA
		All segments and decimal point on, $I_{SEG\_}$ = -40mA		330		
Display Scan Rate	f_{OSC}	8 digits scanned	500	800	1300	Hz
Digit Drive Sink Current	I_{DIGIT}	V+ = 5V, V_{OUT} = 0.65V	320			mA
Segment Drive Source Current	I_{SEG}	T_A = +25°C, V+ = 5V, V_{OUT} = (V+ - 1V)	-30	-40	-45	mA
Segment Current Slew Rate (MAX7221 only)	$\Delta I_{SEG}/\Delta t$	T_A = +25°C, V+ = 5V, V_{OUT} = (V+ - 1V)	10	20	50	mA/µs
Segment Drive Current Matching	ΔI_{SEG}			3.0		%
Digit Drive Leakage (MAX7221 only)	I_{DIGIT}	Digit off, V_{DIGIT} = V+			-10	µA
Segment Drive Leakage (MAX7221 only)	I_{SEG}	Segment off, V_{SEG} = 0V			1	µA
Digit Drive Source Current (MAX7219 only)	I_{DIGIT}	Digit off, V_{DIGIT} = (V+ - 0.3V)	-2			mA
Segment Drive Sink Current (MAX7219 only)	I_{SEG}	Segment off, V_{SEG} = 0.3V	5			mA

ELECTRICAL CHARACTERISTICS (continued)

(V+ = 5V ±10%, R_{SET} =9.53kΩ ±1%, T_A = T_{MIN} to T_{MAX}, unless otherwise noted.)

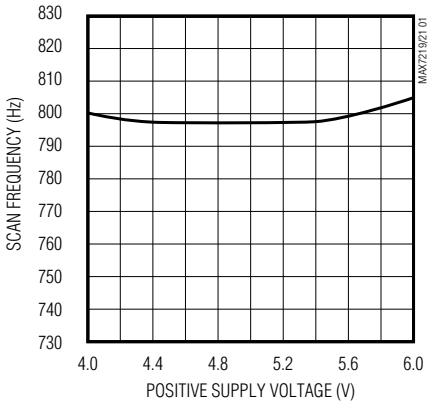
PARAMETER	SYMBOL	CONDITIONS	MIN	TYP	MAX	UNITS
LOGIC INPUTS						
Input Current DIN, CLK, LOAD, $\overline{CS}$	I_{IH}, I_{IL}	V_{IN} = 0V or V+	-1		1	µA
Logic High Input Voltage	V_{IH}		3.5			V
Logic Low Input Voltage	V_{IL}				0.8	V
Output High Voltage	V_{OH}	DOUT, I_{SOURCE} = -1mA	V+ - 1			V
Output Low Voltage	V_{OL}	DOUT, I_{SINK} = 1.6mA			0.4	V
Hysteresis Voltage	ΔV_I	DIN, CLK, LOAD, $\overline{CS}$		1		V
TIMING CHARACTERISTICS						
CLK Clock Period	t_{CP}		100			ns
CLK Pulse Width High	t_{CH}		50			ns
CLK Pulse Width Low	t_{CL}		50			ns
$\overline{CS}$ Fall to SCLK Rise Setup Time (MAX7221 only)	t_{CSS}		25			ns
CLK Rise to $\overline{CS}$ or LOAD Rise Hold Time	t_{CSH}		0			ns
DIN Setup Time	t_{DS}		25			ns
DIN Hold Time	t_{DH}		0			ns
Output Data Propagation Delay	t_{DO}	C_{LOAD} = 50pF			25	ns
Load-Rising Edge to Next Clock Rising Edge (MAX7219 only)	t_{LDCK}		50			ns
Minimum $\overline{CS}$ or LOAD Pulse High	t_{CSW}		50			ns
Data-to-Segment Delay	t_{DSPD}				2.25	ms

Serially Interfaced, 8-Digit LED Display Drivers

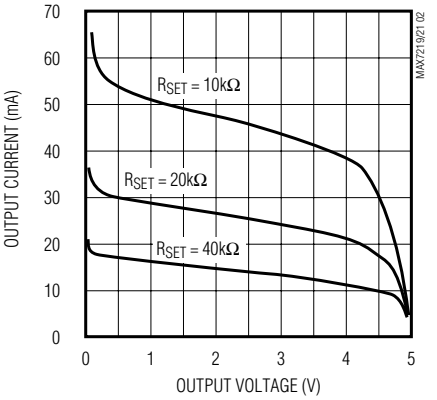
Typical Operating Characteristics

(V+ = +5V, T_A = +25°C, unless otherwise noted.)

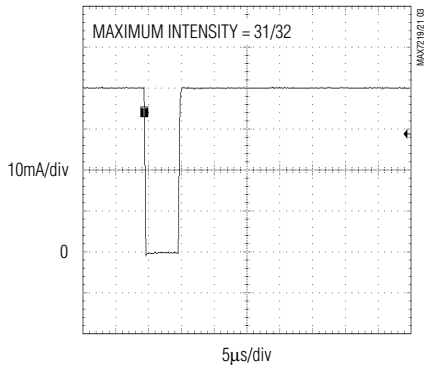
SCAN FREQUENCY vs. POSITIVE SUPPLY VOLTAGE



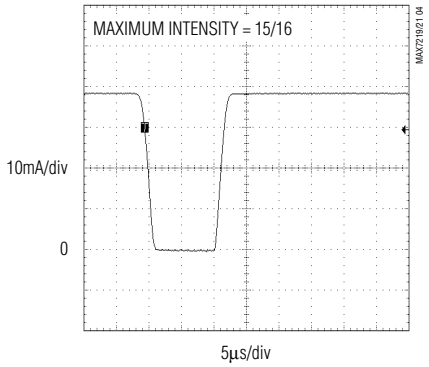
SEGMENT DRIVER OUTPUT CURRENT vs. OUTPUT VOLTAGE



MAX7219 SEGMENT OUTPUT CURRENT



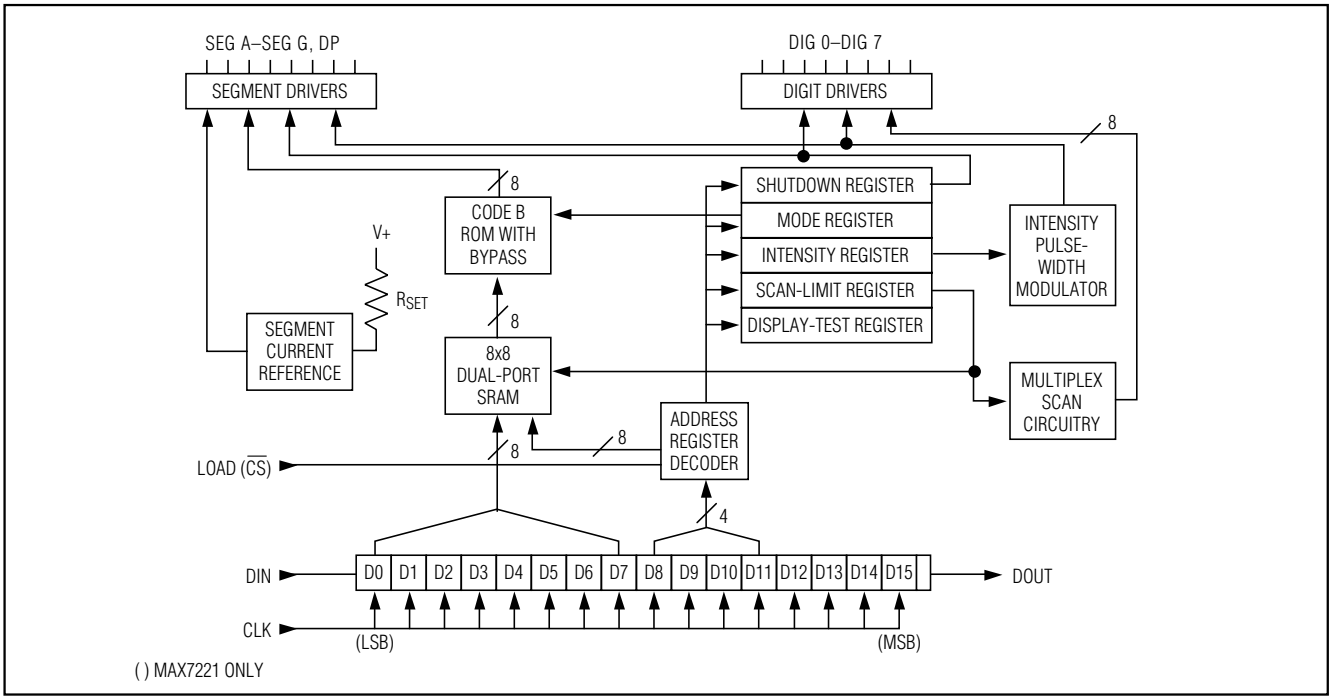
MAX7221 SEGMENT OUTPUT CURRENT



Pin Description

PIN	NAME	FUNCTION
1	DIN	Serial-Data Input. Data is loaded into the internal 16-bit shift register on CLK's rising edge.
2, 3, 5–8, 10, 11	DIG 0–DIG 7	Eight-Digit Drive Lines that sink current from the display common cathode. The MAX7219 pulls the digit outputs to V+ when turned off. The MAX7221's digit drivers are high-impedance when turned off.
4, 9	GND	Ground (both GND pins must be connected)
12	LOAD (MAX7219)	Load-Data Input. The last 16 bits of serial data are latched on LOAD's rising edge.
	$\overline{CS}$ (MAX7221)	Chip-Select Input. Serial data is loaded into the shift register while $\overline{CS}$ is low. The last 16 bits of serial data are latched on $\overline{CS}$'s rising edge.
13	CLK	Serial-Clock Input. 10MHz maximum rate. On CLK's rising edge, data is shifted into the internal shift register. On CLK's falling edge, data is clocked out of DOUT. On the MAX7221, the CLK input is active only while $\overline{CS}$ is low.
14–17, 20–23	SEG A–SEG G, DP	Seven Segment Drives and Decimal Point Drive that source current to the display. On the MAX7219, when a segment driver is turned off it is pulled to GND. The MAX7221 segment drivers are high-impedance when turned off.
18	ISET	Connect to V_{DD} through a resistor (R_{SET}) to set the peak segment current (Refer to *Selecting R_{SET} Resistor* section).
19	V+	Positive Supply Voltage. Connect to +5V.
24	DOUT	Serial-Data Output. The data into DIN is valid at DOUT 16.5 clock cycles later. This pin is used to daisy-chain several MAX7219/MAX7221's and is never high-impedance.

Functional Diagram



Serially Interfaced, 8-Digit LED Display Drivers

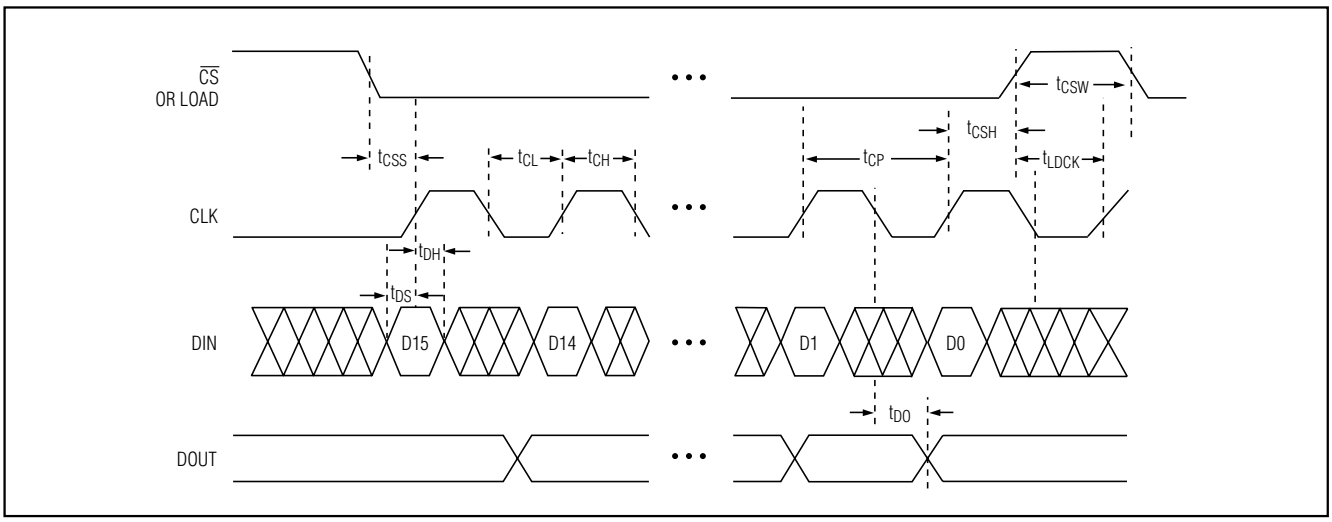


Figure 1. Timing Diagram

Table 1. Serial-Data Format (16 Bits)

D15	D14	D13	D12	D11	D10	D9	D8	D7	D6	D5	D4	D3	D2	D1	D0
X	X	X	X	ADDRESS				MSB	DATA						LSB

Detailed Description

MAX7219/MAX7221 Differences

The MAX7219 and MAX7221 are identical except for two parameters: the MAX7221 segment drivers are slew-rate limited to reduce electromagnetic interference (EMI), and its serial interface is fully SPI compatible.

Serial-Addressing Modes

For the MAX7219, serial data at DIN, sent in 16-bit packets, is shifted into the internal 16-bit shift register with each rising edge of CLK regardless of the state of LOAD. For the MAX7221, $\overline{CS}$ must be low to clock data in or out. The data is then latched into either the digit or control registers on the rising edge of LOAD/$\overline{CS}$. LOAD/$\overline{CS}$ must go high concurrently with or after the 16th rising clock edge, but before the next rising clock edge or data will be lost. Data at DIN is propagated through the shift register and appears at DOUT 16.5 clock cycles later. Data is clocked out on the falling edge of CLK. Data bits are labeled D0–D15 (Table 1). D8–D11 contain the register address. D0–D7 contain the data, and D12–D15 are "don't care" bits. The first received is D15, the most significant bit (MSB).

Digit and Control Registers

Table 2 lists the 14 addressable digit and control registers. The digit registers are realized with an on-chip, 8x8 dual-port SRAM. They are addressed directly so that individual digits can be updated and retain data as long as V+ typically exceeds 2V. The control registers consist of decode mode, display intensity, scan limit (number of scanned digits), shutdown, and display test (all LEDs on).

Shutdown Mode

When the MAX7219 is in shutdown mode, the scan oscillator is halted, all segment current sources are pulled to ground, and all digit drivers are pulled to V+, thereby blanking the display. The MAX7221 is identical, except the drivers are high-impedance. Data in the digit and control registers remains unaltered. Shutdown can be used to save power or as an alarm to flash the display by successively entering and leaving shutdown mode. For minimum supply current in shutdown mode, logic inputs should be at ground or V+ (CMOS-logic levels).

Typically, it takes less than 250µs for the MAX7219/MAX7221 to leave shutdown mode. The display driver can be programmed while in shutdown mode, and shutdown mode can be overridden by the display-test function.

Table 2. Register Address Map

REGISTER	ADDRESS					HEX CODE
	D15–D12	D11	D10	D9	D8	
No-Op	X	0	0	0	0	0xX0
Digit 0	X	0	0	0	1	0xX1
Digit 1	X	0	0	1	0	0xX2
Digit 2	X	0	0	1	1	0xX3
Digit 3	X	0	1	0	0	0xX4
Digit 4	X	0	1	0	1	0xX5
Digit 5	X	0	1	1	0	0xX6
Digit 6	X	0	1	1	1	0xX7
Digit 7	X	1	0	0	0	0xX8
Decode Mode	X	1	0	0	1	0xX9
Intensity	X	1	0	1	0	0xXA
Scan Limit	X	1	0	1	1	0xXB
Shutdown	X	1	1	0	0	0xXC
Display Test	X	1	1	1	1	0xXF

Initial Power-Up

On initial power-up, all control registers are reset, the display is blanked, and the MAX7219/MAX7221 enter shutdown mode. Program the display driver prior to display use. Otherwise, it will initially be set to scan one digit, it will not decode data in the data registers, and the intensity register will be set to its minimum value.

Decode-Mode Register

The decode-mode register sets BCD code B (0-9, E, H, L, P, and -) or no-decode operation for each digit. Each bit in the register corresponds to one digit. A logic high selects code B decoding while logic low bypasses the decoder. Examples of the decode mode control-register format are shown in Table 4.

When the code B decode mode is used, the decoder looks only at the lower nibble of the data in the digit registers (D3–D0), disregarding bits D4–D6. D7, which sets the decimal point (SEG DP), is independent of the decoder and is positive logic (D7 = 1 turns the decimal point on). Table 5 lists the code B font.

When no-decode is selected, data bits D7–D0 correspond to the segment lines of the MAX7219/MAX7221. Table 6 shows the one-to-one pairing of each data bit to the appropriate segment line.

Table 3. Shutdown Register Format (Address (Hex) = 0xXC)

MODE	ADDRESS CODE (HEX)	REGISTER DATA							
		D7	D6	D5	D4	D3	D2	D1	D0
Shutdown Mode	0xXC	X	X	X	X	X	X	X	0
Normal Operation	0xXC	X	X	X	X	X	X	X	1

Table 4. Decode-Mode Register Examples (Address (Hex) = 0xX9)

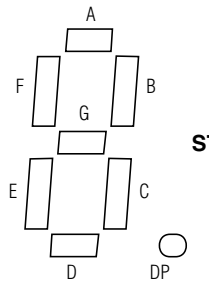
DECODE MODE	REGISTER DATA								HEX CODE
	D7	D6	D5	D4	D3	D2	D1	D0	
No decode for digits 7–0	0	0	0	0	0	0	0	0	0x00
Code B decode for digit 0 No decode for digits 7–1	0	0	0	0	0	0	0	1	0x01
Code B decode for digits 3–0 No decode for digits 7–4	0	0	0	0	1	1	1	1	0x0F
Code B decode for digits 7–0	1	1	1	1	1	1	1	1	0xFF

Table 5. Code B Font

7-SEGMENT CHARACTER	REGISTER DATA						ON SEGMENTS = 1							
	D7*	D6–D4	D3	D2	D1	D0	DP*	A	B	C	D	E	F	G
0		X	0	0	0	0		1	1	1	1	1	1	0
1		X	0	0	0	1		0	1	1	0	0	0	0
2		X	0	0	1	0		1	1	0	1	1	0	1
3		X	0	0	1	1		1	1	1	1	0	0	1
4		X	0	1	0	0		0	1	1	0	0	1	1
5		X	0	1	0	1		1	0	1	1	0	1	1
6		X	0	1	1	0		1	0	1	1	1	1	1
7		X	0	1	1	1		1	1	1	0	0	0	0
8		X	1	0	0	0		1	1	1	1	1	1	1
9		X	1	0	0	1		1	1	1	1	0	1	1
—		X	1	0	1	0		0	0	0	0	0	0	1
E		X	1	0	1	1		1	0	0	1	1	1	1
H		X	1	1	0	0		0	1	1	0	1	1	1
L		X	1	1	0	1		0	0	0	1	1	1	0
P		X	1	1	1	0		1	1	0	0	1	1	1
blank		X	1	1	1	1		0	0	0	0	0	0	0

*The decimal point is set by bit D7 = 1

Table 6. No-Decode Mode Data Bits and Corresponding Segment Lines



STANDARD 7-SEGMENT LED

	REGISTER DATA							
	D7	D6	D5	D4	D3	D2	D1	D0
Corresponding Segment Line	DP	A	B	C	D	E	F	G

Intensity Control and Interdigit Blanking

The MAX7219/MAX7221 allow display brightness to be controlled with an external resistor (R_{SET}) connected between V+ and ISET. The peak current sourced from the segment drivers is nominally 100 times the current entering ISET. This resistor can either be fixed or variable to allow brightness adjustment from the front panel. Its minimum value should be 9.53kΩ, which typically sets the segment current at 40mA. Display brightness can also be controlled digitally by using the intensity register.

Digital control of display brightness is provided by an internal pulse-width modulator, which is controlled by the lower nibble of the intensity register. The modulator scales the average segment current in 16 steps from a maximum of 31/32 down to 1/32 of the peak current set by R_{SET} (15/16 to 1/16 on MAX7221). Table 7 lists the intensity register format. The minimum interdigit blanking time is set to 1/32 of a cycle.

Table 7. Intensity Register Format (Address (Hex) = 0xXA)

DUTY CYCLE		D7	D6	D5	D4	D3	D2	D1	D0	HEX CODE
MAX7219	MAX7221									
1/32 (min on)	1/16 (min on)	X	X	X	X	0	0	0	0	0xX0
3/32	2/16	X	X	X	X	0	0	0	1	0xX1
5/32	3/16	X	X	X	X	0	0	1	0	0xX2
7/32	4/16	X	X	X	X	0	0	1	1	0xX3
9/32	5/16	X	X	X	X	0	1	0	0	0xX4
11/32	6/16	X	X	X	X	0	1	0	1	0xX5
13/32	7/16	X	X	X	X	0	1	1	0	0xX6
15/32	8/16	X	X	X	X	0	1	1	1	0xX7
17/32	9/16	X	X	X	X	1	0	0	0	0xX8
19/32	10/16	X	X	X	X	1	0	0	1	0xX9
21/32	11/16	X	X	X	X	1	0	1	0	0xXA
23/32	12/16	X	X	X	X	1	0	1	1	0xXB
25/32	13/16	X	X	X	X	1	1	0	0	0xXC
27/32	14/16	X	X	X	X	1	1	0	1	0xXD
29/32	15/16	X	X	X	X	1	1	1	0	0xXE
31/32	15/16 (max on)	X	X	X	X	1	1	1	1	0xXF

Table 8. Scan-Limit Register Format (Address (Hex) = 0xXB)

SCAN LIMIT	REGISTER DATA								HEX CODE
	D7	D6	D5	D4	D3	D2	D1	D0	
Display digit 0 only*	X	X	X	X	X	0	0	0	0xX0
Display digits 0 & 1*	X	X	X	X	X	0	0	1	0xX1
Display digits 0 1 2*	X	X	X	X	X	0	1	0	0xX2
Display digits 0 1 2 3	X	X	X	X	X	0	1	1	0xX3
Display digits 0 1 2 3 4	X	X	X	X	X	1	0	0	0xX4
Display digits 0 1 2 3 4 5	X	X	X	X	X	1	0	1	0xX5
Display digits 0 1 2 3 4 5 6	X	X	X	X	X	1	1	0	0xX6
Display digits 0 1 2 3 4 5 6 7	X	X	X	X	X	1	1	1	0xX7

*See *Scan-Limit Register* section for application.

Scan-Limit Register

The scan-limit register sets how many digits are displayed, from 1 to 8. They are displayed in a multiplexed manner with a typical display scan rate of 800Hz with 8 digits displayed. If fewer digits are displayed, the scan rate is $8f_{OSC}/N$, where N is the number of digits scanned. Since the number of scanned digits affects the display brightness, the scan-limit register should not be used to blank portions of the display (such as leading zero suppression). Table 8 lists the scan-limit register format.

If the scan-limit register is set for three digits or less, individual digit drivers will dissipate excessive amounts of power. Consequently, the value of the R_{SET} resistor must be adjusted according to the number of digits displayed, to limit individual digit driver power dissipation. Table 9 lists the number of digits displayed and the corresponding maximum recommended segment current when the digit drivers are used.

Display-Test Register

The display-test register operates in two modes: normal and display test. Display-test mode turns all LEDs on by overriding, but not altering, all controls and digit registers (including the shutdown register). In display-test mode, 8 digits are scanned and the duty cycle is 31/32 (15/16 for MAX7221). Table 10 lists the display-test register format.

Table 9. Maximum Segment Current for 1-, 2-, or 3-Digit Displays

NUMBER OF DIGITS DISPLAYED	MAXIMUM SEGMENT CURRENT (mA)
1	10
2	20
3	30

Table 10. Display-Test Register Format (Address (Hex) = 0xXF)

MODE	REGISTER DATA							
	D7	D6	D5	D4	D3	D2	D1	D0
Normal Operation	X	X	X	X	X	X	X	0
Display Test Mode	X	X	X	X	X	X	X	1

Note: The MAX7219/MAX7221 remain in display-test mode (all LEDs on) until the display-test register is reconfigured for normal operation.

No-Op Register

The no-op register is used when cascading MAX7219s or MAX7221s. Connect all devices' LOAD/$\overline{CS}$ inputs together and connect DOUT to DIN on adjacent devices. DOUT is a CMOS logic-level output that easily drives DIN of successively cascaded parts. (Refer to the *Serial Addressing Modes* section for detailed information on serial input/output timing.) For example, if four MAX7219s are cascaded, then to write to the fourth chip, sent the desired 16-bit word, followed by three no-op codes (hex 0xXX0X, see Table 2). When LOAD/$\overline{CS}$ goes high, data is latched in all devices. The first three chips receive no-op commands, and the fourth receives the intended data.

Applications Information

Supply Bypassing and Wiring

To minimize power-supply ripple due to the peak digit driver currents, connect a 10µF electrolytic and a 0.1µF ceramic capacitor between V+ and GND as close to the device as possible. The MAX7219/MAX7221 should be placed in close proximity to the LED display, and connections should be kept as short as possible to minimize the effects of wiring inductance and electromagnetic interference. Also, both GND pins must be connected to ground.

Selecting R_{SET} Resistor and Using External Drivers

The current per segment is approximately 100 times the current in ISET. To select R_{SET}, see Table 11. The MAX7219/MAX7221's maximum recommended segment current is 40mA. For segment current levels above these levels, external digit drivers will be needed. In this application, the MAX7219/MAX7221 serve only as controllers for other high-current drivers or transistors. Therefore, to conserve power, use R_{SET} = 47kΩ when using external current sources as segment drivers.

The example in Figure 2 uses the MAX7219/MAX7221's segment drivers, a MAX394 single-pole double-throw analog switch, and external transistors to drive 2.3" AND2307SLC common-cathode displays. The 5.6V zener diode has been added in series with the decimal point LED because the decimal point LED forward voltage is typically 4.2V. For all other segments the LED forward voltage is typically 8V. Since external transistors are used to sink current (DIG 0 and DIG 1 are used as logic switches), peak segment currents of 45mA are allowed even though only two digits are displayed. In applications where the MAX7219/MAX7221's digit drivers are used to sink current and fewer than four digits are displayed, Table 9 specifies the maximum allowable segment current. R_{SET} must be selected accordingly (Table 11).

Refer to the Power Dissipation section of the Absolute Maximum Ratings to calculate acceptable limits for ambient temperature, segment current, and the LED forward-voltage drop.

Table 11. RSET vs. Segment Current and LED Forward Voltage

I_{SEG} (mA)	V_{LED} (V)				
	1.5	2.0	2.5	3.0	3.5
40	12.2	11.8	11.0	10.6	9.69
30	17.8	17.1	15.8	15.0	14.0
20	29.8	28.0	25.9	24.5	22.6
10	66.7	63.7	59.3	55.4	51.2

Computing Power Dissipation

The upper limit for power dissipation (PD) for the MAX7219/MAX7221 is determined from the following equation:

$$PD = (V + \times 8mA) + (V+ - V_{LED})(DUTY \times I_{SEG} \times N)$$

where:

V+ = supply voltage

DUTY = duty cycle set by intensity register

N = number of segments driven (worst case is 8)

V_{LED} = LED forward voltage

I_{SEG} = segment current set by R_{SET}

Dissipation Example:

I_{SEG} = 40mA, N = 8, DUTY = 31/32, V_{LED} = 1.8V at 40mA, V+ = 5.25V

PD = 5.25V(8mA) + (5.25V - 1.8V)(31/32 x 40mA x 8) = 1.11W

Thus, for a CERDIP package (θ_{JA} = +80°C/W from Table 12), the maximum allowed ambient temperature T_A is given by:

$$T_{J(MAX)} = T_A + PD \times \theta_{JA} + 150°C = T_A + 1.11W \times 80°C/W$$

where T_A = +61.2°C.

The T_A limits for PDIP and SO Packages in the dissipation example above are +66.7°C and +55.6°C, respectively.

Table 12. Package Thermal Resistance Data

PACKAGE	THERMAL RESISTANCE (θ_{JA})
24 Narrow DIP	+75°C/W
24 Wide SO	+85°C/W
24 CERDIP	+80°C/W
Maximum Junction Temperature (T_J) = +150°C	
Maximum Ambient Temperature (T_A) = +85°C	

Cascading Drivers

The example in Figure 3 drives 16 digits using a 3-wire µP interface. If the number of digits is not a multiple of 8, set both drivers' scan limits registers to the same number so one display will not appear brighter than the other. For example, if 12 digits are need, use 6 digits per display with both scan-limit registers set for 6 digits so that both displays have a 1/6 duty cycle per digit. If 11 digits are needed, set both scan-limit registers for 6 digits and leave one digit driver unconnected. If one display for 6 digits and the other for 5 digits, the second display will appear brighter because its duty cycle per digit will be 1/5 while the first display's will be 1/6. Refer to the *No-Op Register* section for additional information.

Serially Interfaced, 8-Digit LED Display Drivers

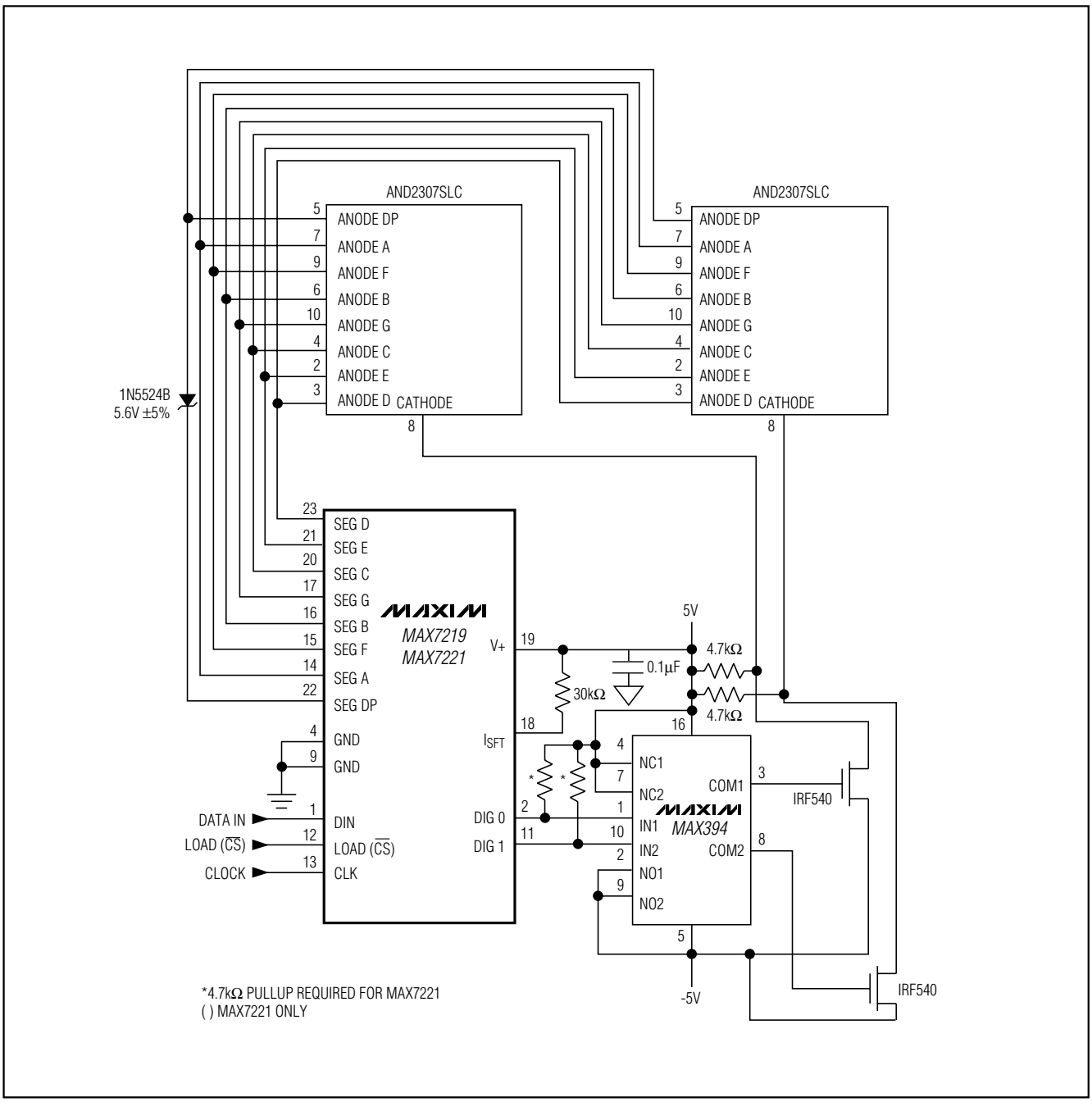


Figure 2. MAX7219/MAX7221 Driving 2.3in Displays

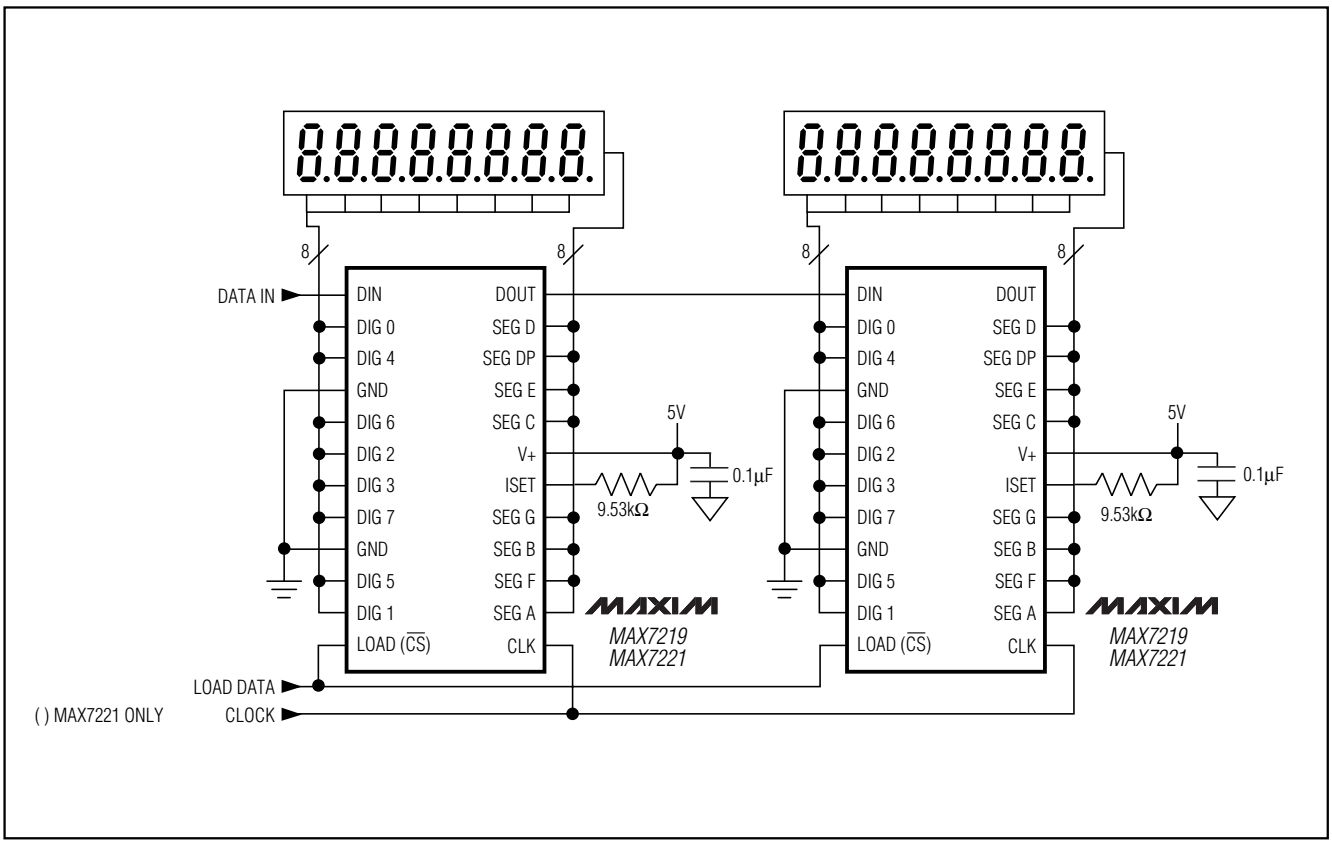


Figure 3. Cascading MAX7219/MAX7221s to Drive 16 Seven-Segment LED Digits

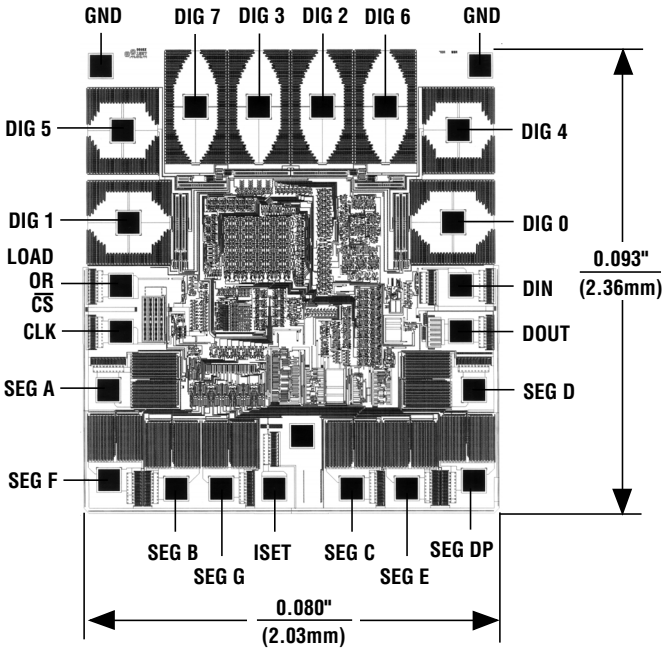
Serially Interfaced, 8-Digit LED Display Drivers

Ordering Information (continued)

PART	TEMP RANGE	PIN-PACKAGE
MAX7221CNG	0°C to +70°C	24 Narrow Plastic DIP
MAX7221CWG	0°C to +70°C	24 Wide SO
MAX7221C/D	0°C to +70°C	Dice*
MAX7221ENG	-40°C to +85°C	24 Narrow Plastic DIP
MAX7221EWG	-40°C to +85°C	24 Wide SO
MAX7221ERG	-40°C to +85°C	24 Narrow CERDIP

**Dice are specified at T_A = +25°C.*

Chip Topography



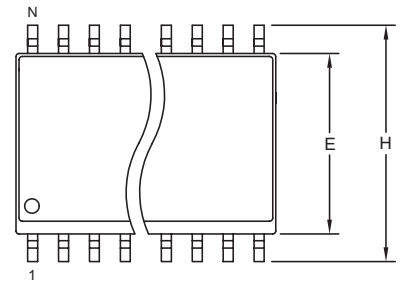
TRANSISTOR COUNT: 5267

SUBSTRATE CONNECTED TO GND

Package Information

(The package drawing(s) in this data sheet may not reflect the most current specifications. For the latest package outline information go to **www.maxim-ic.com/packages**.)

SOICW.EPS

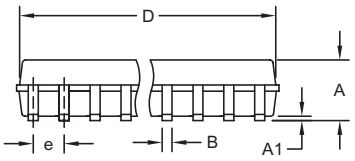


TOP VIEW

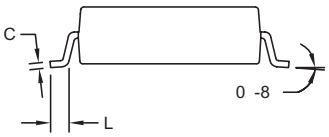
	INCHES		MILLIMETERS	
DIM	MIN	MAX	MIN	MAX
A	0.093	0.104	2.35	2.65
A1	0.004	0.012	0.10	0.30
B	0.014	0.019	0.35	0.49
C	0.009	0.013	0.23	0.32
e	0.050		1.27	
E	0.291	0.299	7.40	7.60
H	0.394	0.419	10.00	10.65
L	0.016	0.050	0.40	1.27

VARIATIONS:

	INCHES		MILLIMETERS			
DIM	MIN	MAX	MIN	MAX	N	MS013
D	0.398	0.413	10.10	10.50	16	AA
D	0.447	0.463	11.35	11.75	18	AB
D	0.496	0.512	12.60	13.00	20	AC
D	0.598	0.614	15.20	15.60	24	AD
D	0.697	0.713	17.70	18.10	28	AE



FRONT VIEW



SIDE VIEW

NOTES:
1. D&E DO NOT INCLUDE MOLD FLASH.
2. MOLD FLASH OR PROTRUSIONS NOT TO EXCEED 0.15mm (.006").
3. LEADS TO BE COPLANAR WITHIN 0.10mm (.004").
4. CONTROLLING DIMENSION: MILLIMETERS.
5. MEETS JEDEC MS013.
6. N = NUMBER OF PINS.

DALLAS SEMICONDUCTOR MAXIM

PROPRIETARY INFORMATION

TITLE:

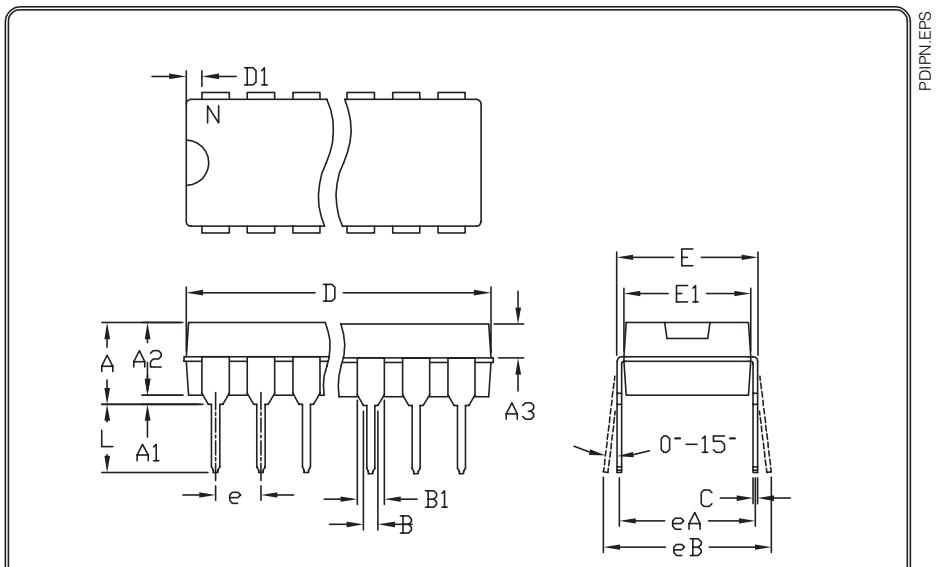
PACKAGE OUTLINE, .300" SOIC

APPROVAL	DOCUMENT CONTROL NO.	REV.	
	21-0042	B	1/1

Serially Interfaced, 8-Digit LED Display Drivers

Package Information (continued)

(The package drawing(s) in this data sheet may not reflect the most current specifications. For the latest package outline information go to **www.maxim-ic.com/packages**.)



	INCHES		MILLIMETERS	
	MIN	MAX	MIN	MAX
A	---	0.180	---	4.572
A1	0.015	---	0.38	---
A2	0.125	0.175	3.18	4.45
A3	0.055	0.080	1.40	2.03
B	0.015	0.022	0.381	0.56
B1	0.045	0.065	1.14	1.65
C	0.008	0.014	0.2	0.355
D1	0.005	0.080	0.13	2.03
E	0.300	0.325	7.62	8.26
E1	0.240	0.310	6.10	7.87
e	0.100 BSC.		2.54 BSC.	
eA	0.300 BSC.		7.62 BSC.	
eB	0.400 BSC.		10.16 BSC.	
L	0.115	0.150	2.921	3.81

	INCHES		MILLIMETERS			
	MIN	MAX	MIN	MAX	N	MS001
D	0.348	0.390	8.84	9.91	8	AB
D	0.735	0.765	18.67	19.43	14	AC
D	0.745	0.765	18.92	19.43	16	AA
D	0.885	0.915	22.48	23.24	18	AD
D	1.015	1.045	25.78	26.54	20	AE
D	1.14	1.265	28.96	32.13	24	AF
D	1.360	1.380	34.54	35.05	28	*5

NOTES:
1. D&E DO NOT INCLUDE MOLD FLASH
2. MOLD FLASH OR PROTRUSIONS NOT TO EXCEED .15mm (.006")
3. CONTROLLING DIMENSION: MILLIMETER
4. MEETS JEDEC MS001-XX AS SHOWN IN ABOVE TABLE
5. SIMILIAR TO JEDEC MO-058AB
6. N = NUMBER OF PINS

MAXIM 120 SAN GABRIEL DR. SUNNYVALE CA 94086 FAX (408) 737 7194 PROPRIETARY INFORMATION

TITLE: PACKAGE FAMILY OUTLINE: PDIP .300" | 1/1 | 21-0043 D (DOCUMENT CONTROL NUMBER REV.)