DISPOSITIVO DE ALMACENAMIENTO ESTANDAR PARA SOLUCION EMBEBIDA

Di Giulio, Pablo Andrés / Grupo T.D.A. / Departamento de Ingeniería Electrónica / U.T.N. Facultad Regional San Francisco

CONTEXTO

El grupo T.D.A. (Técnicas Digitales Aplicadas) vienen llevando a cabo diversas actividades en lo referente a implementaciones digitales. Actualmente se están llevando a cabo muchos trabajos de I/D en lo que tiene que ver con la interconectividad con la PC, protocolos de comunicación, etc.

RESUMEN

Los dispositivos de almacenamiento masivo constituyen una parte muy importante de cualquier sistema o instalación informática. Hoy en día son la columna vertebral que permite que los diversos sistemas y dispositivo interactúen.
La estandarización en la forma en que se almacena y transporta esta información es muy importante, fundamentalmente a la hora de hacer un desarrollo electrónico propio que utilice un dispositivo de almacenamiento de uso estándar (CD, DVD, Pendrive, MMC, SD, etc.).
El objetivo del presente trabajo de I/D es evaluar los distintos dispositivos de almacenamiento estándar que existen en el mercado y verificar que factibilidad existe de utilizarlo en un desarrollo electrónico propio que utilice un microcontrolador como CPU.
Analizadas las diversas alternativas, observamos que las tarjetas de memorias SD son la que menos recursos del microcontrolador requieren para su manejo. Se desarrollo un pequeño hardware para el manejo físico de dicho dispositivo. Por otra parte se desarrollo una pila (Stack) de software para el acceso al dispositivo además del sistema de archivos que permita comprender la información que está montada sobre el mismo.
Como conclusión, se exponen los recursos mínimos necesarios para poder utilizar este medio de almacenamiento masivo en una aplicación propietaria.
Palabras clave: *Arquitectura de PC, Dispositivo de almacenamiento masivo, Sistema de archivos, Sistema Operativo, Microcontroladores.*

1. INTRODUCCION

El presente trabajo surgió debido a la necesidad del grupo, de utilizar un medio de almacenamiento de información estándar en el mercado para mover información desde un desarrollo electrónico propio a una PC.

Normalmente, la mayoría de los desarrollos electrónicos propietarios, utilizan como controlador un microcontrolador, por ejemplo, control de CNC, datalogger, estación meteorológica, control de potencia, instrumentos de medición, etc.

Un microcontrolador (MCU - Micro Controller Unit) es un circuito integrado que incluye en su interior las tres unidades funcionales de una computadora: CPU, memoria y unidad de entrada/salida, pero con capacidades limitadas. Los MCU se clasifican según su estructura interna en 8-bit, 16-bit y 32-bit, la cual determina el poder del mismo, siendo el MCU de 8-bit el de menor performance y el de 32-bit el de mayor.

A la hora de diseñar un sistema electrónico con MCU es muy importante tener en cuenta la cantidad de recursos que se va a necesitar de este para correr la aplicación, ya que normalmente y en los MCU de línea baja, los recursos son limitados. Los puntos a tener en cuenta, en la elección de un MCU para una determina aplicación son los siguientes:

- Cantidad Memoria Flash
- Cantidad Memoria RAM
- Velocidad de reloj interna
- Tensión de alimentación
- Consumo
- GPIO (General Propouse Input Output) a utilizar.
- Protocolos de comunicaciones (USB, Serial, Ethernet, etc.)

Uno de los puntos a tener en cuenta, en la elección del MCU a utilizar en una determinada aplicación, es si requiere del uso de la PC para algún servicio, por ejemplo visualizar graficas, realizar cálculos de mayor complejidad, realizar backup de la información, etc. Si la aplicación requiere de la PC, debemos evaluar si nuestra aplicación va a estar conectada de forma directa a la PC, a través de un protocolo de comunicación (USB, Serial, Ethernet) o si la aplicación no va a estar conectada todo el tiempo y solamente va a almacenar la información en una memoria y luego descargarla.
Habitualmente, en las aplicaciones donde la PC esta distante, se utiliza un medio de almacenamiento/transporte de información que sea de uso estándar, pudiéndose encontrar repuestos de forma masiva a la hora del recambio y sobre todo ahorrarle trabajo al diseñador del software, ya que el uso de un dispositivo de almacenamiento estándar es soportado por cualquier S.O. (Sistema Operativo).

Por lo tanto la mayor complejidad queda a cargo del desarrollo electrónico, ya que su aplicación, debe acceder a este dispositivo de almacenamiento, poder leer y escribir información de forma que el S.O. de la PC comprenda dicha información. Para poder utilizar el medio de almacenamiento primero debemos acceder al dispositivo y luego montar el sistema de archivos (Filesystem) para poder acceder a la información de forma estructurada y comprenderla.

2. DISPOSITIVOS DE ALMACENAMIENTO MASIVO

Para seleccionar el dispositivo que utilice la menor cantidad de recursos de nuestro MCU realizamos una evaluación de los distintos medios de almacenamiento que existen en el mercado. Evaluamos la capacidad de los mismo, velocidad, hardware adicional requerido, complejidad del acceso, etc.
Como conclusión de la evaluación de cada uno de los dispositivos de almacenamiento, confeccionamos la Tabla 1., que resume las ventajas y desventajas de cada unos de los dispositivos, al ser utilizados en un desarrollo electrónico propio como medio de transporte de información.

	Ventajas	Desventajas
CD-ROM, DVD	Velocidad Capacidad de almacenamiento moderada	Dimensiones físicas grande Complejidad para su manejo Requiere de partes móviles
Disco Rígido	Velocidad Capacidad de almacenamiento	Dimensiones físicas grande Complejidad para su manejo Alto costo
Pendrive	Velocidad Capacidad de almacenamiento moderada Bajo costo Dimensiones físicas pequeñas	Requiere del manejo de USB Host
Tarjetas SD/MMC	Velocidad moderada Bajo costo Capacidad de almacenamiento moderada Dimensiones físicas pequeñas	

Tabla 1: Comparativa de los medios de almacenamiento del mercado hoy en día.

Los pendrive son dispositivo que podrían ser utilizados en este estudio. El principal inconveniente, para su manejo, es que el MCU que comande a este debe poseer la capacidad de manipular el protocolo USB Host, lo cual no se consigue en todos los MCU. Como podemos ver en la Tabla 1., las tarjetas de memoria tienen las mejores prestaciones, ya que son de bajo costo, fácil acceso y velocidad moderada.

2.1.1 SECURE DIGITAL

Las SD es un formato de tarjeta de memoria flash que se utiliza en dispositivos portátiles tales como cámaras fotográficas digitales, ordenadores PDA y Palm, entre otros.

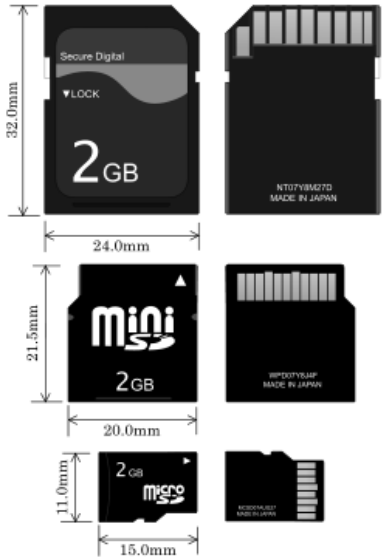

Figura 1: Dimensiones físicas de SD.

Las dimensiones físicas de las tarjetas SD son reducidas, Figura 1, las cuales la hacen muy útiles en aplicaciones donde las dimensiones físicas del producto final son un punto a tener en cuenta. Su costo en el mercado es bajo y existen distintos tipos de zócalos los cuales permiten su contención. Otro punto a favor de dichas memorias es que no requieren de partes móviles a la hora de leerlas o escribirlas, esto es importantísimo cuando la aplicación debe ser montada sobre partes móviles, lo que permite tener menor cantidad de fallas y reajustes del dispositivo. Un punto a favor muy grande además, es que tiene un tipo de interface (protocolo) que es estándar en la mayoría de los MCU grandes y pequeños.

2.1.2 INTERFAZ

Las memorias SD disponen de nueve pines de interface (clock, command, 4 x data, 3 power lines) diseñadas para aplicaciones de niveles de tensión bajos.

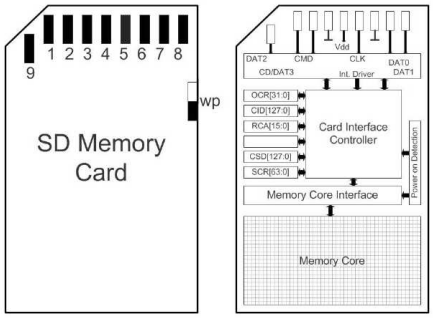

Figura 1: Interface de la tarjeta SD.

2.1.3 TIPOS DE TRANSFERENCIA

Existen 3 modos de transferencia soportados por SD:

- Modo SPI: entrada y salida serial separadas. Requiere de 4 líneas (MISO, MOSI, CLK, CS)
- Modo un-bit SD: comandos separados del canal de datos. Es un tipo de transferencia propietaria.
- Modo cuatro-bit SD: utilizado para transferencia paralelas. Utiliza más pines.

Las tarjetas de baja velocidad soportan tasas de transferencias de 0 a 400 Kbps y modo de trasferencia un-bit SD, mientras que las tarjetas de alta velocidad soportan tasas de transferencia de 0 a 100 Mbps en el modo de cuatro-bit, y de 0 a 25 Mbps en el modo un-bit SD.

Dentro de los tipos de transferencia, el modo SPI (Serial Peripheral Interfase) es el más interesante de todos ya que este tipo de bus se encuentra en la mayoría de los MCU de línea baja y no requiere de mucho conocimiento para su uso ya que es muy simple y básico.

2.1.4 MODO SPI

Básicamente el protocolo SPI consiste en el intercambio de información entre el MCU (master) y la tarjeta (slave). Este intercambio se lleva a cabo mediante el envió de comandos por parte del controlador y de respuestas por parte de las tarjetas. Así, en la lectura, el MCU envía el comando de petición de lectura a la tarjeta y esta le envía la respuesta de confirmación seguida del bloque de datos con la información contenida a partir de la dirección solicitada, mientras que el controlador envía mensajes vacios (tokens) para permitir la salida de datos desde la tarjeta. En la escritura el proceso es parecido, el MCU indica a la tarjeta mediante el comando de escritura que quiere escribir información en una determinada dirección, esta le responde indicando que esta lista y a continuación el MCU envía el bloque de datos a escribir.

Figura 2: Esquema de comunicación

Las operaciones que no requieren intercambio de datos funcionan de igual forma pero sin usar bloques de datos. En la Figura 2, se esquematiza el modelo de comunicación

2.1.5 COMANDOS

Para realizar transacciones entre el MCU y la tarjeta se deben enviar comandos, los cuales tienen la siguiente estructura:

Byte 1			Byte 2-5	Byte 6	
7	6	5-0	31-0	7	0
0	1	Command	Command Argument	CRC	1

Tabla 2: Frame de comando.

Como podemos observar en la Tabla 2, los primeros dos bits del primer byte son 01 lo que indica que se está enviando un comando. Los comandos están codificados en binario natural como se ilustra en [2]. El último byte (byte 6) es el byte de CRC para la verificación de errores y en realidad en el protocolo SPI no se utiliza, a no ser que en el registro de configuración se especifique que se desea utilizar CRC. Estos son algunos de los principales comandos:

CMD	Argum.	Respu.	Descripción
CMD0	No	R1	Resetea la tarjeta
CMD1	No	R1	Inicializa la tarjeta
CMD9	No	R1	Pide a la tarjeta su información CSD
CMD10	No	R1	Pide a la tarjeta su identificación CID
CMD13	No	R2	Consulta el estado de la tarjeta
CMD16	[31..0] Long. del bloque.	R1	Establece la longitud (en bytes) del bloque para los datos en las operaciones de lectura y escritura.
CMD17	[31..0] Long. del bloque.	R1	Lee un bloque del tamaño indicado por el comando 16.
CMD24	[31..0] Long. del bloque.	R1 R1 R1	Escribe un bloque del tamaño indicado por el comando 16.

Tabla 3: Comandos.

Toda la comunicación entre la tarjeta y el controlador se realiza según el orden del esquema,

de izquierda a derecha, es decir que primero se transmite el bit de más peso (bit 7) del byte 1, y por último el bit de menos peso (bit 0) del byte 6, es decir es una transferencia More Significant Bit First (MSB First).

Las respuestas de la tarjeta son bloques formados por 1 o 2 bytes, dependiendo del tipo de respuesta que se trate. El tipo de respuesta es función del comando, es decir que cada comando tiene asociado un tipo de respuesta.

2.2 SISTEMA DE ARCHIVOS (FILESYSTEM)

Una vez investigado el funcionamiento de la tarjeta de memoria SD, realizamos las pruebas de escritura, lectura y borrado. Con esto ya logramos desarrollar la capa de acceso a la memoria. El siguiente paso es desarrollar el sistema de archivos para poder acceder a la información que está en la memoria de forma coherente.

Desde el punto de vista del usuario de la PC todo se ve tan fácil: se copian archivos de un directorio a otro, se formatea un disquete o se borra un archivo, y todo esto con un solo comando. Pero detrás del telón existe un largo camino hasta que el controlador ha localizado la posición de un archivo o un espacio libre en la memoria que se pueda emplear para guardar un archivo. Ya que las memorias no quieren saber nada de archivos y sub-directorios; sólo conocen de posiciones de memoria; el S.O. (Sistema Operativo) ha de equipar a cada almacenamiento masivo con un sistema de archivos, que haga posible el salto del nivel físico al nivel lógico. Este sistema de archivos se compone de una serie de estructuras de datos, que describe el tamaño de la memoria su contenido.

2.2.1 ESTRUCTURA DEL VOLUMEN

Existen distintos tipos de sistemas de archivo, en nuestro caso, el sistema de archivos implementado es FAT. Cuando se habla de sistemas de archivos, se tiene que tener bien en claro el termino de volumen. Como se detalla en [3] es importante anotar que cada volumen tiene una estructura homogénea, independientemente de si está en un disquete o en un disco duro. El tamaño no tiene ninguna importancia, ya que simplemente afecta al tamaño de las diferentes estructuras de datos, que se necesitan para la gestión del volumen.

Un volumen está estructurado de la siguiente forma:

Sector de arranque
Primera tabla de localización de archivos (FAT)
Una o más copias de la FAT
Directorio raíz (eventualmente con etiqueta de volumen)
Zona de datos para archivos y sub-directorios

Tabla 4: Estructura de un volumen

Además de conocer el significado de volumen, se nombraran otros términos como sectores que son bloques de memoria que contienen una cierta cantidad de bytes. Dependiendo del sistema de archivos implementado este tamaño puede variar

2.2.2 SECTOR DE ARRANQUE

La estructura del sector de arranque es la primera que se crea luego de formateado el dispositivo de almacenamiento. Contiene todo lo referente al volumen como número de sectores por volumen, cantidad de bytes por volumen. También contiene el llamado Bootstrap-Loader, mediante el cual se puede arrancar el SO. A esta circunstancia es a la que se la llama sector de arranque.

2.2.3 LA TABLA FAT

La zona correspondiente a la FAT es donde se encuentra la administración de espacios. En ella se detallan los archivos existentes y los espacios libres. Cada entrada a la tabla corresponde a un corresponde con un numero de sectores. Estos grupos de sectores se llaman cluster y se define en [3]. Dependiendo del tipo de dispositivo el valor de la cantidad de sectores por cluster puede cambiar Como se ejemplifica en [4] el tamaño de la entrada FAT depende de una cierta cantidad de bit. En nuestro caso la FAT implementada es FAT16 por lo que el tamaño de la entrada a la tabla es de 16-bit, pudiéndose direccionar hasta 65535 clusters.

La tabla FAT además de almacenar si un cluster está ocupado, indica la posición a donde se guardan los archivos. Cada entrada a la FAT tiene la siguiente codificación:

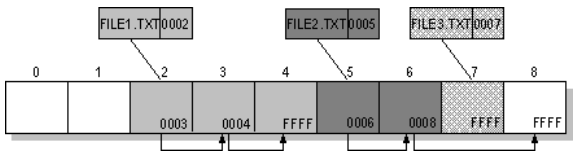

Figura 3: Estructura de la FAT.

Como podemos observar en la Figura 3, las dos primeras posiciones de memoria de la zona de la tabla FAT no se deben escribir, la ubicación de los archivos recién comienza recién en la tercera posición de memoria. El ejemplo se puede observar en [4].

2.2.4 DIRECTORIO RAIZ

Luego de la tabla FAT y su copia viene zona llamada “Directorio Raíz”. En él se ubican todas las informaciones relevantes sobre los archivos y subdirectorios allí contenidos.

Cada archivo o sub-directorio posee una estructura o entrada en el directorio raíz, el detalle de esta zona se especifica en [3].

2.2.5 ZONA DE DATOS

Al directorio raíz le sigue la zona de datos cuyo cluster se emplea para guardar archivos y subdirectorios. Para cada cluster en esta zona

naturalmente existe una entrada en la FAT, que se puede modificar mediante la manipulación de archivos en esta zona.

2.3 ENSAMBLE FINAL - LECTURA DE UN ARCHIVO

Una vez estudiado el sistema de archivos FAT, se desarrollaron las estructuras correspondientes a las distintas zonas del volumen. Con dichas estructuras y las librerías desarrolladas para la lectura y escritura de la SD se generaron las funciones para el acceso a la información. En nuestro caso, se realizaron las funciones de lectura de archivos únicamente, quedando para una segunda etapa las rutinas de escritura.

2.3.1 MONTANDO EL SISTEMA DE ARCHIVOS

Antes de realizar la lectura de un archivo se debe montar el sistema de archivo sobre el MCU. En este proceso se extrae la información almacenada en la primera zona del volumen (Sector de arranque). De dicha zona se utilizan los siguientes datos:

- Bytes por sector (offset 0x0B)
- Sectores por cluster (offset 0x0D)
- Entradas al directorio raíz (offset 0x11)
- Numero de sectores por FAT (offset 0x16)

Con dicha información se calculan las posiciones de las demás zonas del volumen.

2.3.2 LECTURA

El proceso de lectura a grandes rasgos es simple. Inicialmente se debe ingresar a la zona directorio raíz, donde se busca a través de las estructuras el nombre del archivo deseado. Una vez ubicada la estructura se extrae la posición que ocupa este en la tabla FAT. Con el valor extraído, se indexa en la tabla FAT y se extrae el offset que hay que aplicarle a la zona de datos para ubicar la información del archivo buscado.
Si el archivo deseado se encuentra en una carpeta, las mismas se encuentra en la zona de datos, pero a diferencia de los archivos, en vez de contener información contienen la estructura de los archivos como las que se encuentran en la zona directorio raíz.

3. RESULTADOS OBTENIDOS

El objetivo central de este trabajo fue investigar y desarrollar una capa de software para ser embebida en un MCU, que permita acceder a un dispositivo de almacenamiento masivo estándar para transportar y almacenar información. Este dispositivo fue seleccionado luego de realizar un análisis del consumo de recursos, que requiere la gran mayoría de los dispositivos de almacenamiento que existe hoy en el mercado.
Las tarjetas de memoria SD son las que fueron seleccionas, ya que ofrecen las mejores prestaciones, son pequeñas físicamente, su modo de acceso es simple (SPI – A través de 4 líneas), no poseen partes móviles y su velocidad de acceso es moderada para las aplicaciones que se pueden llegar a desarrollar.
Se desarrollo el código fuente sobre un MCU MC9S08AW60 de la compañía Freescale Inc., [5].
Como resultado se exponen los mínimos recursos que debe disponer el MCU que comanda el desarrollo, para poder comandar un dispositivo estándar como son las tarjetas de memoria SD:

- Tensión de alimentación 3.3V
- Memoria Flash: 2 KBytes.
- Memoria RAM: 600 Bytes.
- Modulo SPI
- Velocidad máxima de SPI CLK 2.5 MHz.
- Compilador ANSI C.

Los resultados anteriores tienen las siguientes consideraciones:

- El código fuente realizado sobre un MCU, cuyo set de instrucciones es del tipo CISC, por lo que el tamaño de la memoria flash especificada como mínima puede varias si el set de instrucciones es del tipo RISC.
- La cantidad de memoria RAM especificada contempla un buffer de datos de 512 bytes. Este se utiliza tanto para lectura como escritura. La transferencia de escritura de la tarjeta requiere del envió de 512 bytes pero en el proceso de lectura de la memoria se puede especificar, a través del CMD16, un tamaño de lectura que puede variar de 1 a 512 bytes. Si el MCU dispone de escasa memoria RAM se puede reducir el tamaño de este buffer.

4. FORMACION DE RECURSOS HUMANOS

En la actualidad existen trabajos similares, muchos de ellos tienen el inconveniente de que no indexan a través de sub-directorios en busca de archivos, por lo que solamente podemos manejar archivos que están en la raíz. Esta opción que se propone, de buscar a través de sub-directorios, es muy interesante ya que el costo en la memoria flash es mínimo al igual que la latencia de lectura. Actualmente estamos trabajando en conjunto con el Laboratorio de DSP de la U.N.C. el cual tiene muchos años ya y nos tendió su mano este proyecto de I/D.

5. BIBLIOGRAFIA

[1] SD Association. *"www.sdorg.org"*
[2] SD Media Format Expand the MAXQ2000's Space for Nonvolatile Data Storage. Application note 3969. Maxim IC.
[3] Tischer M., Jennrich B., (1996). PC Interno 5, Programación de sistemas. Marcombo.
[4] NTFS.com. *"www.ntfs.com"*
[5] Freescale Inc. *"www.freescale.com"*