ESCUELA POLITÉCNICA DEL EJÉRCITO

FACULTAD DE INGENIERÍA ELECTRÓNICA

PROYECTO DE GRADO PARA LA OBTENCIÓN DEL TÍTULO EN INGENIERÍA ELECTRÓNICA

"DESARROLLO DE UN SISTEMA PARA LECTURA Y ESCRITURA DE TARJETAS MULTIMEDIA (MultiMediaCard MMC) BASADO EN EL MICROCONTROLADOR ATMEL AT89C51SND1C"

EDWIN XAVIER MORA NUÑEZ

MARCO ANTONIO VILLALBA ORELLANA

SANGOLQUÍ – ECUADOR

JUNIO 2006

CERTIFICACIÓN

Certifico que el presente Proyecto de Grado “Desarrollo de un Sistema para Lectura y Escritura de tarjetas Multimedia (MultiMedia Card MMC) basado en el Microcontrolador ATMEL AT89C51SND1C” fue realizado en su totalidad por los señores Edwin Xavier Mora Núñez y Marco Antonio Villalba Orellana, como requerimiento parcial a la obtención del título de Ingeniero Electrónico con especialidad en Automatización y Control, bajo nuestra dirección.

Atentamente,

Ing. Byron Navas	Ing. Víctor Proaño
Director	Codirector

AGRADECIMIENTOS

A nuestros padres por apoyarnos a lo largo de nuestra vida, ya que sin ellos nuestros sueños no serían una realidad. Gracias por su ejemplo de constancia, honestidad y trabajo, que son valores que nunca podremos olvidar en cualquier camino que nos encontremos. A nuestra familia, por brindarnos su afecto y crítica en los errores y aciertos que hemos tenido.

A todos nuestros profesores, en especial al Ing. Byron Navas e Ing. Víctor Proaño, que han sembrado en nosotros los conocimientos para alcanzar la madurez necesaria para enfrentar los problemas y adversidades que se presenten en nuestra vida profesional.

Reiterar nuestros agradecimientos al Ing. Byron Navas por todos los conocimientos compartidos basados en su investigación previa realizada en el campo de los microcontroladores ATMEL, sin la cual la realización de este proyecto no hubiese sido posible.

A nuestros amigos por ser quienes han estado siempre brindándonos su apoyo a pesar de nuestras diferencias, que en lugar de separarnos nos han unido más.

Al servicio técnico de ATMEL, especialmente a Patrice Graziotin, por impulsar y guiarnos en la realización de este proyecto.

A la Escuela Politécnica del Ejército por habernos formado como personas y profesionales activos que contribuyan al desarrollo y crecimiento de este país.

Xavier y Marco

DEDICATORIA

A mis padres Edwin y Lupita por su paciencia y dedicación, con la cual han intentado encontrar lo bueno que hay en mí; y por siempre apoyarme en todas las aventuras de mi vida sin juzgar mis motivos, siendo mis amigos y el espejo en el que me miro para ser mejor.

A mis hermanos Andrés y Nicolás por ser mis cómplices y confidentes. Por brindarme su apoyo incondicional en todos los buenos o malos momentos.

A todos mis amigos, en especial a Marco, por su amistad, confianza y por permitirme compartir con ellos nuestros buenos y malos momentos.

A Mell, por ser mi compañera, amiga y mi inspiración.

Xavier

DEDICATORIA

A mis padres Marco y Pilar por su incondicional apoyo prestado durante toda mi vida, ya que han sido un pilar invaluable en mi formación.

A mi familia, en especial a mis hermanos Lorena, Paola, Santiago y Anibal Palacios por ser las personas que siempre me han extendido su mano en los momentos más difíciles de mi vida, siendo la base para ser una mejor persona y un profesional que enfrente todos los retos de la vida.

A mis amigos y en especial a Xavier Mora por su confianza y afecto que me han impulsado a alcanzar mis metas y objetivos.

Marco

PRÓLOGO

En el presente proyecto está relacionado en la investigación previa realizada en el proyecto de grado titulado ““DESARROLLO DE UN SISTEMA REPRODUCTOR DE ARCHIVOS MPEG I/II LAYER 3 (MP3) BASADO EN EL MICROCONTROLADOR ATMEL AT89C51SND1C”.

El presente proyecto implementa un prototipo que permite la lectura y escritura de tarjetas MultiMediaCard (MMC) basándose en una comunicación en modo MMC por bloques, permitiendo acceder a la información y mostrar los archivos almacenados en la tarjeta MMC, dependiendo de las diferentes extensiones que soporta el prototipo. Permite borrar todos los archivos almacenados en la tarjeta.

En el primer capítulo se introducirá a la teoría de las principales tarjetas multimedias que existen en el mercado, basándose en las especificaciones físicas y eléctricas, velocidad de transmisión, descripción de pines y aplicaciones.

En el segundo capítulo se presenta la información detallada sobre las características más relevantes de las tarjetas MultiMediaCard (MMC) según el análisis de las especificaciones técnicas publicadas por la MultiMediaCard Association (MMCA).

En el tercer capítulo se estudiará el módulo MMC del microcontrolador AT89C51SND1C para conocer el manejo de la generación de la frecuencia del reloj y la sincronización con los controladores de las líneas de comandos y datos. Para una mejor comprensión de este capítulo se recomienda revisar previamente el proyecto de grado relacionado.

En el cuarto capítulo se describirá el diseño e implementación de hardware, el cual está basado en las recomendaciones de los fabricantes de los componentes del prototipo y la investigación de proyectos similares.

En el quinto capítulo se presentará el diseño del código de programa para el prototipo, que permite navegar en el contenido de la tarjeta MMC mediante el ingreso de órdenes. El software utilizado para el desarrollo de código es el µVision 2, desarrollado por la empresa Keil Software.

En el sexto capítulo se presentarán las conclusiones y recomendaciones de toda la investigación realizada en el desarrollo del proyecto.

ÍNDICE GENERAL

CAPÍTULO I

TARJETAS DE ALMACENAMIENTO PARA DATOS MULTIMEDIA

1.1. INTRODUCCIÓN

En este capítulo se analiza los diferentes tipos de tarjetas existentes en el mercado para almacenamiento de datos, las cuales se han convertido en una manera muy práctica de transportar información ya sea imágenes, datos, música etc. Además se puede observar las características principales en lo que respecta a descripción de interfases y protocolos de comunicación para el desarrollo de sistemas que faciliten una manera de manipular la información existente en las tarjetas de almacenamiento masivo.

Hoy en día los datos multimedia son la forma más común de presentar información que emplean una combinación de texto, sonido, imágenes, animación y vídeo. Entre las aplicaciones informáticas multimedia más comunes figuran juegos, programas de aprendizaje y material de referencia.

Las aplicaciones multimedia necesitan de memoria y capacidad de proceso. Por ejemplo, un ordenador multimedia debe poseer la capacidad de ejecutar aplicaciones que necesitan de memoria adicional para ayudar a la CPU a efectuar cálculos y permitir la presentación de imágenes complejas en la pantalla. El ordenador también necesita un disco duro de alta capacidad para almacenar y recuperar la información, así como una unidad de disco compacto para ejecutar aplicaciones en CD-ROM.

Debido a la necesidad de transportar información de una manera segura, rápida y portátil, surgen las tarjetas de almacenamiento de información multimedia. De las cuales, su mayoría se encuentran formadas por una memoria flash, que es un chip de memoria no volátil (esto quiere decir que su contenido permanece aunque al dispositivo se le retire la polarización). En cierto sentido se considera una variante de la memoria EEPROM[1] que puede ser borrada y programada al nivel de byte, mientras que la memoria flash puede ser borrada y reprogramada en unidades de memoria llamadas bloques, cuyo tamaño puede variar desde los 512 bytes hasta los 256 KB. Esto hace que la memoria flash sea muy útil para actualizar la BIOS de un ordenador o computadora, o para almacenar cantidades de información importantes, como una colección de imágenes o de documentos de texto.

1.2. PRINCIPALES MODELOS DE TARJETAS

En el mercado existen distintos tipos de tarjetas de memoria flash que se describirán en esta sección:

- PC Card.
- CompactFlash (CF).
- Memory Stick (MS).
- SmartMedia Card (SM).
- xD-Picture Card.
- Secure Digital (SD).
- Memoria Flash USB.

[1] EEPROM: Electrically Erasable Programmable Read-Only Memory, memoria de lectura solamente borrable y programable eléctricamente.

1.2.1. PC CARD

Figura. 1.1 Tarjeta PC Card

A comienzos de los años 90 el rápido desarrollo del manejo de la computación móvil obligó al desarrollo de pequeñas, livianas y portátiles herramientas para transportar información. Una de estas innovaciones es la tecnología PC Card (Figura 1.1) que son periféricos del tamaño de una tarjeta de crédito y han logrado la estandarización en el mercado brindando beneficios especialmente para aplicaciones industriales.

Las PC Cards se rigen por un estándar internacional controlado por la PCMCIA[2]. En 1991, esta Organización definió una interfase para dispositivos I/O con un conector de 68 pines, inicialmente usado en las tarjetas de memoria manteniendo compatibilidad con los modelos anteriores.

Especificaciones Físicas.

De acuerdo a las especificaciones físicas pueden existir tres tipos de PC Cards como se muestra en la Figura 1.2. Las tarjetas poseen las mismas dimensiones tanto en largo como en ancho y usan el mismo conector de 68 pines, la única

[2] PCMCIA: Asociación Internacional de Tarjetas de Memoria para Computadoras Personales. Fundada en 1989.

diferencia entre este tipo de tarjetas es el espesor, los cuales son 3.3, 5.0, y 10.5 milímetros para el Tipo I, Tipo II y Tipo III respectivamente.

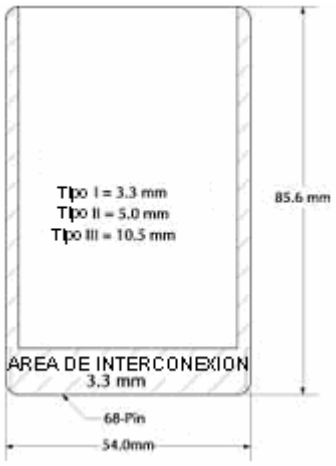


Figura 1.2. Dimensiones Físicas de la PC Card

El host se conecta con la tarjeta PC Card por medio del conector de 68 pines. En la Tabla 1.1 se muestran las dimensiones físicas de la tarjeta PC Card.[3]

Interfase física	68 pines
Largo	85.6 mm
Ancho	54.0 mm
Espesor	Tipo I = 3.3 mm Tipo II = 5.0 mm Tipo III = 10.5 mm

Tabla 1.1. Características Físicas de la Tarjeta PC Card

[3] Estándares publicados por la PCMCIA.

Especificaciones Eléctricas.

De acuerdo a las especificaciones eléctricas y por interfase de comunicación, existen dos tipos de PC Cards:

- **Antiguo estándar PCMCIA:** Operan a 5 (V) y funcionan en un bus ISA de 16 bits.
- **CardBus PCI:** Operan a 3.3 (V) y funcionan en un bus PCI de 32 bits.

Velocidad de Transmisión.

En la Tabla 1.2 se muestran las velocidades de transmisión para tarjetas PC Card PCI. Esta información no es de libre acceso debido a los estándares publicados por la PCMCIA.

Tipo de tarjeta	Valor	Unidad	Aplicación
PC Card PCI	133	Mbps	Zoomed Video
	100		Ethernet

Tabla 1.2. Velocidades de Transmisión para Tarjetas PC Card PCI

Zoomed Video es una conexión entre la PC Card y el sistema, la cual permite a la tarjeta escribir datos de video directamente al controlador VGA de un ordenador, los datos son transferidos sin necesidad de un buffer de almacenamiento ya que son enviados al bus Zoomed Video y no utiliza el bus del sistema.[4] Debido a esto surgieron las especificaciones de CardBus y de Zoomed Video, que han permitido el desarrollo de video MPEG y Ethernet 100 Mbit.

[4] http://www.pcmcia.org/pccard.htm, PC Card

Descripción de Pines.

La distribución de pines para las interfases PC Card de 16-bit y para la CardBus de 32-bit se describe a continuación en la Tabla 1.3:

Pin	16-Bit ISA	32-Bit PCI	Pin	16-Bit ISA	32-Bit PCI
1	GND	GND	35	GND	GND
2	D3	CAD0	36	CD1#	CCD1#
3	D4	CAD1	37	D11	CAD2
4	D5	CAD3	38	D12	CAD4
5	D6	CAD5	39	D13	CAD6
6	D7	CAD7	40	D14	RSRVD
7	CE1#	CCBE0#	41	D15	CAD8
8	A10	CAD9	42	CE2#	CAD10
9	OE#	CAD11	43	VS1#	CVS1
10	A11	CAD12	44	RSRVD	CAD13
11	A9	CAD14	45	RSRVD	CAD15
12	A8	CCBE1#	46	A17	CAD16
13	A13	CPAR	47	A18	RSRVD
14	A14	CPERR#	48	A19	CBLOCK#
15	WE#	CGNT#	49	A20	CSTOP#
16	READY	CINT#	50	A21	CDEVSEL#
17	Vcc	Vcc	51	Vcc	Vcc
18	Vpp1	Vpp1	52	Vpp2	Vpp2
19	A16	CLCC	53	A22	CTRDY
20	A15	CIRDY#	54	A23	CFRAME#
21	A12	CCBE2#	55	A24	CAD17
22	A7	CAD18	56	A25	CAD19
23	A6	CAD20	57	VS2#	CVS2
24	A5	CAD21	58	RESET	CRST#
25	A4	CAD22	59	WAIT#	CSERR#
26	A3	CAD23	60	RSRVD	CREQ#
27	A2	CAD24	61	REG#	CCBE3#
28	A1	CAD25	62	BVD2	CAUDIO
29	A0	CAD26	63	BVD1	CSTSCHG
30	D0	CAD27	64	D8	CAD28
31	D1	CAD29	65	D9	CAD30
32	D2	RSVD	66	D10	CAD31
33	WP	CCLKRUN#	67	CD2#	CCD2#
34	GND	GND	68	GND	GND

Tabla 1.3. Distribución de Pines para las Interfases PC Card

Aplicaciones.

Los dispositivos digitales disminuyen en cuanto a su tamaño y aumentan su capacidad de memoria, por lo que el estándar PC Card seguirá con una tendencia a introducir en el mercado dispositivos con mayor capacidad de almacenamiento y convirtiéndose en una gran solución.

Las tarjetas Tipo I son menos comunes. La mayoría de PC Cards son de Tipo II, como por ejemplo: módems, tarjetas de sonido, interfaces de red, adaptadores CompactFlash y dispositivos de almacenamiento masivo. Las aplicaciones más comunes para PC Cards Tipo III se orientan a trabajar como medios de discos duros portátiles.

A continuación se presentan algunos ejemplos de los dispositivos disponibles en la tecnología PC Card:

- Conversores A/D y otros dispositivos de adquisición de datos.
- Interfase CD-ROM.
- Interfase Teléfono Celular.
- Tarjetas GPS (Sistema de posicionamiento Global).
- Discos duros.
- Tarjetas de módems y de Ethernet.
- Tarjetas de sonido.
- Tarjetas de video conferencia.

1.2.2. COMPACT FLASH (CF)

Figura 1.3. Tarjeta CompactFlash

El concepto de la tecnología para tarjetas CompactFlash es simple: capturar, retener y transportar datos, audio e imágenes. CF es un pequeño dispositivo de almacenamiento masivo como se muestra en la Figura 1.3. Fue introducida por primera vez en 1994 por la corporación SanDisk y fue el primer estándar de memoria Flash en competir con las tarjetas PC Card.

La Asociación CompactFlash (CFA) fue establecida en Octubre de 1995, incluye miembros como las compañías: Canon, Kodak, Hewlett-Packard, Hitachi, Lexar Media, Renesas y SanDisk. CFA es una corporación que promueve la adopción y el manejo de la tecnología CompactFlash en todo el mundo, al ser la encargada de aprobar y publicar el estándar CF. La CFA ha desarrollado las especificaciones CF+ para lograr expandir el concepto CF más allá del almacenamiento de datos e incluir dispositivos de I/O.

La tarjeta CF posee un controlador inteligente que maneja los protocolos de interfase, almacenamiento de datos, control de reloj y código de corrección de error (ECC). Contiene un módulo de memoria flash. Posee un conector de 50 pines que consiste en dos filas de 25 contactos.

Especificaciones Físicas.

Las especificaciones CF disponen que el host se conecta con la tarjeta CF por medio del conector de 50 pines, el cual consiste en dos columnas de 25 contactos a una separación de 1.27 mm. En la Tabla 1.4 se muestran las dimensiones físicas para las tarjetas CF.

LARGO	36.4 ± 0.15 mm	
ANCHO	42.80 ± 0.10 mm	
ESPESOR INCLUYE ETIQUETA	3.3 mm ± 0.10 mm	CF
ESPESOR INCLUYE ETIQUETA	5.0 mm ± 0.10 mm	CF+

Tabla 1.4. Dimensiones Físicas de Tarjetas CF

En la Figura 1.4 se muestran las dimensiones físicas de las tarjetas CF.

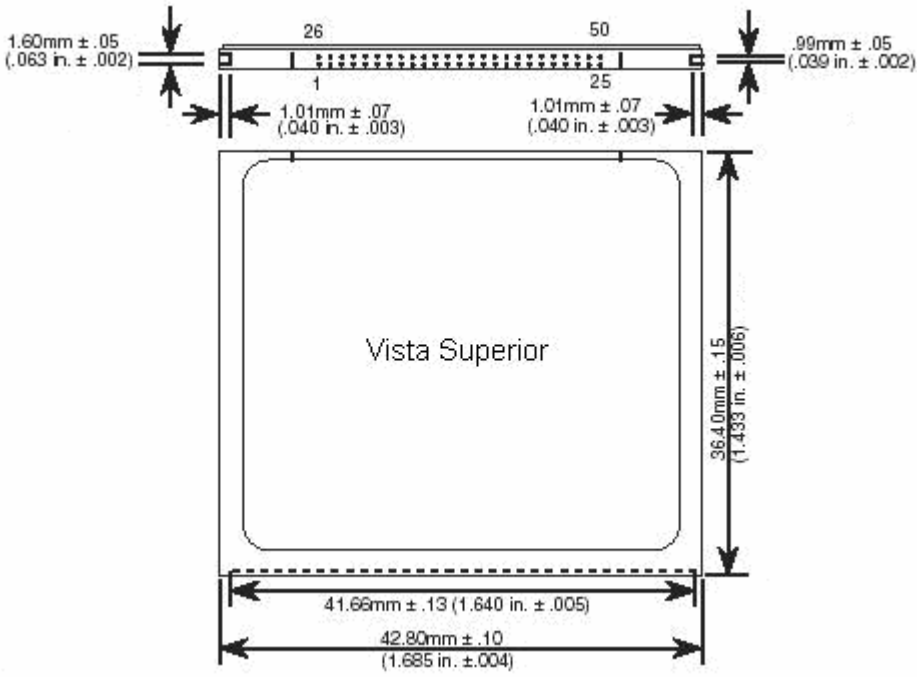


Figura 1.4. Dimensiones Físicas de las Tarjetas CompactFlash

Especificaciones Eléctricas.

Las tarjetas CompactFlash soportan una operación dual a 3.3 y 5 voltios por lo que pueden ser intercambiadas entre estos dos sistemas. Otros tipos de tarjetas CF solo pueden operar a uno de estos voltajes.[5]

Velocidad de Transmisión.

Las especificaciones para las velocidades de transmisión de las tarjetas CF no son de libre acceso debido a que la CFA limita el uso de esta información únicamente a los miembros de esta Asociación.

Descripción de Pines.

La asignación de pines se encuentra especificada en la Tabla 1.5.

[5] http://cfa.org/thecnics/CompactFlash, CompactFlash

CompactFlash (CF)							
Número de Pin	Señal	Tipo de Pin	Tipo I/O	Número de Pin	Señal	Tipo de Pin	Tipo I/O
1	GND		GROUND	26	-CD1	O	GROUND
2	D03	I/O	I1Z,0Z3	27	D11	I/O	I1Z,0Z3
3	D04	I/O	I1Z,0Z3	28	D12	I/O	I1Z,0Z3
4	D05	I/O	I1Z,0Z3	29	D13	I/O	I1Z,0Z3
5	D06	I/O	I1Z,0Z3	30	D14	I/O	I1Z,0Z3
6	D07	I/O	I1Z,0Z3	31	D15	I/O	I1Z,0Z3
7	-CE1	I	I3U	32	-CE2	I	I3U
8	A10	I	I1Z	33	-VS1	O	GROUND
9	-OE	I	I3U	34	-IORD	I	I3U
10	A09	I	I1Z	35	-IOWR	I	I3U
11	A08	I	I1Z	36	-WE	I	I3U
12	A07	I	I1Z	37	READY	O	OT1
13	VCC		POWER	38	VCC		POWER
14	A06	I	I1Z	39	-CSEL	I	I2Z
15	A05	I	I1Z	40	-VS2	O	OPEN
16	A04	I	I1Z	41	RESET	I	I2Z
17	A03	I	I1Z	42	-WAIT	O	OT1
18	A02	I	I1Z	43	-INPACK	O	OT1
19	A01	I	I1Z	44	-REG	I	I3U
20	A00	I	I1Z	45	BVD2	O	OT1
21	D00	I/O	I1Z,0Z3	46	BVD1	O	OT1
22	D01	I/O	I1Z,0Z3	47	D08	I/O	I1Z,0Z3
23	D02	I/O	I1Z,0Z3	48	D09	I/O	I1Z,0Z3
24	WP	O	OT3	49	D10	I/O	I1Z,0Z3
25	-CD2	O	GROUND	50	GND		GROUND

* Señales de activación en bajo poseen el prefijo "-". El tipo de pines son de Entrada, Salida o Entrada/Salida

Tabla 1.5. Descripción de Pines Tarjeta CF

Aplicaciones.

Diversas compañías se encuentran diseñando una nueva tecnología CF para ser usadas en productos como computadoras portátiles, cámaras digitales, PDAs, comunicaciones personales, reproductores de MP3 y dispositivos de monitoreo. Esta tecnología ofrece aplicaciones para lograr expandir el almacenamiento de datos y el desempeño como dispositivos I/O.

1.2.3. MEMORY STICK (MS)

Figura. 1.5. Tarjeta Memory Stick

Memory Stick es una tarjeta de memoria que se muestra en la Figura 1.5. Fue lanzada al mercado por Sony en Octubre de 1998. MS ha sido desarrollada para el uso exclusivo de esta compañía para almacenar, transportar y reproducir contenidos informáticos, tanto imágenes, sonidos y música. Los responsables de Sony pretenden convertir a Memory Stick en el formato universal para todo tipo de aplicaciones multimedia.

Los rasgos más sobresalientes de MS son un tamaño reducido y compacto, construcción robusta, seguridad en los datos para evitar la pérdida, continuidad del producto gracias a la filosofía de Sony, alta velocidad de grabación y reproducción que incorpora funciones de protección contra las copias digitales.

Especificaciones Físicas.

MS tiene dimensiones similares al largo de una batería AA. La forma es diseñada ergonómicamente para ser de fácil manipulación y al igual que otras memorias Flash es fácil de distinguir la parte frontal de la posterior como se observa en la Figura 1.6.[6]

[6] http://www.sony.org/Memory_Stick, Memory Stick.

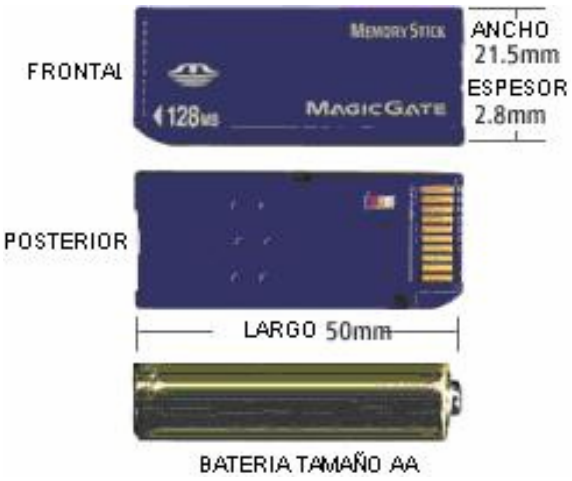


Figura 1.6. Vistas de la Tarjeta MS

MS posee un conector de 10 pines, además de un switch de prevención que permite evitar la eliminación accidental o el mal uso de los datos. MS ha sido diseñada para poseer una alta resistencia a vibraciones debido a la construcción de estado sólido que no posee partes móviles. En la Tabla 1.6 se muestran las dimensiones físicas de la tarjeta Memory Stick.

Dimensiones (milímetros)	50 x 21.5 x 2.8
Dimensiones (pulgadas)	1.97 x 0.85 x 0.11
Tipo de Conector	10 pines

Tabla 1.6. Dimensiones Físicas de la tarjeta MS

Especificaciones Eléctricas.

En la Tabla 1.7 se muestran las especificaciones eléctricas para tarjetas MS, así como también el tipo de memoria y la interfase de comunicación.

IC Tipo de Memoria	Flash Memory
Interfase	Serial
Voltaje de Operación	2.7 - 3.6 Voltios
Consumo de Energía	Promedio : aproximadamente 45mA

Tabla 1.7. Especificaciones Técnicas para las Tarjetas MS

Velocidad de Transmisión.

En la Tabla 1.8 se muestran las velocidades de transmisión para tarjetas MS.

Tasa de Transmisión de datos	Máx. 20MHz
Velocidad de escritura de acceso	Máx. 1.8MB/segundo
Velocidad de lectura de acceso	Máx. 2.5MB/segundo

Tabla 1.8. Velocidades de Transmisión de las Tarjetas MS

Descripción de Pines.

La información acerca de la descripción de pines de las tarjetas MS no es libre acceso debido a que Sony no posee un estándar abierto para la compatibilidad de sus dispositivos. En la Figura 1.7 se observa la vista de la superficie de conexión de la tarjeta MS.

Figura. 1.7. Vista de la Superficie de Conexión de la Tarjeta MS

Aplicaciones.

Las tarjetas MS son compatibles con dispositivos Sony como: DVD, sistemas de audio en automóviles, sistemas de navegación para automóviles, teléfonos celulares, computadoras, PDAs, diccionarios electrónicos, cámaras digitales, televisores, proyectores digitales, unidades de video conferencia, impresoras digitales, robots de entretenimiento y en los periféricos de las computadoras

1.2.4. SMARTMEDIA CARD (SM)

Figura 1.8. SmartMedia Card

SmartMedia es una tarjeta de memoria que se observa en la Figura 1.8. Fue introducida por Toshiba en el año 1995 para competir con las tarjetas CompactFlash y PC Card. La tarjeta SmartMedia fue inicialmente conocida como Solid State Floppy Disk Card y sus siglas en ingles son: SSFDC.

SmartMedia era popular en las cámaras digitales y alcanzó su cresta de ventas en el año 2001, cuando se la utilizaba como medio de almacenamiento en casi la mitad de cámaras digitales que se comercializaban. El formato empezó a presentar problemas con los productos de Fuji y Olimpo, debido a que su tamaño es demasiado grande y por este motivo, las tarjetas SmartMedia no encontraron el apoyo suficiente para convertirse en un estándar.

Toshiba y Samsung todavía construyen tarjetas SmartMedia para ser usadas en dispositivos existentes en el mercado. Una ventaja que permanece por encima de otros formatos es la capacidad de acceder a cualquier tarjeta SmartMedia por un floppy de 3.5", usando un adaptador conocido como FlashPath.

Especificaciones Físicas.

En la Tabla 1.9 se muestran las dimensiones físicas de las tarjetas SM y en la Figura 1.9 se observa dichas dimensiones.

Tamaño	45.0 × 37.0 × 0.76 mm.
Terminal de electrodo plano	22 pines.

Tabla 1.9. Dimensiones Físicas de la tarjeta SM.

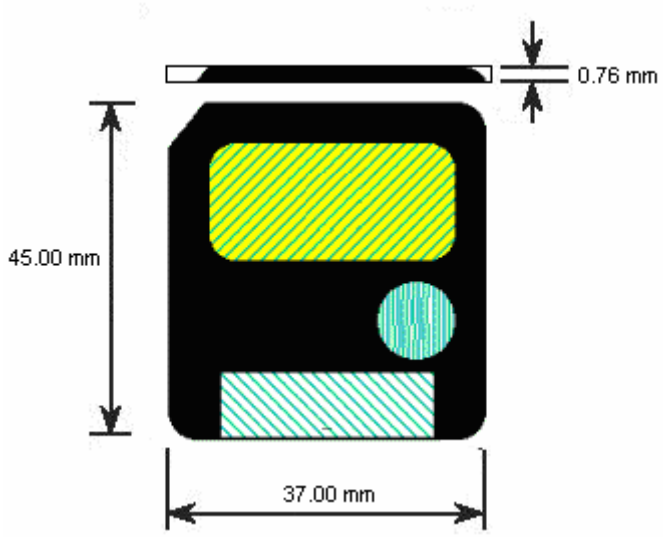


Figura 1.9. Dimensiones Físicas de la Tarjeta SM.

Especificaciones Eléctricas.

Algunas tarjetas SM tienen la capacidad de funcionar en un modo dual de voltaje, mientras que otras solo soportan voltaje de 3.3 (V) o 5 (V).[7]

[7] http://searchstorage.techtarget.com/sDefinition/0,,sid5_gci837127,00.html, SmartMedia.

Velocidad de Transmisión.

En la Tabla 1.10 se muestran las especificaciones de velocidad de transmisión para tarjetas SM.

Interfase I/O	8-bit (16-bit en algunos casos).
Velocidad de transmisión de datos	2MB/s

Tabla 1.10. Velocidades de Transmisión de la Tarjeta SM

Descripción de Pines.

La asignación de pines se encuentra especificada en las Figuras 1.10 y 1.11. La muesca en el lado izquierdo indica que la tarjeta SM opera con 5 (V), mientras que los 3.3V se señalan con la muesca en el lado derecho.

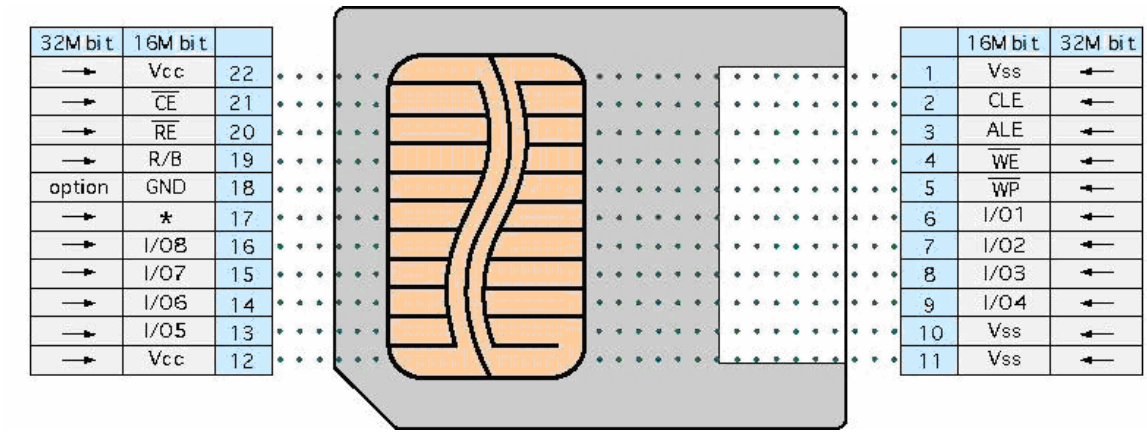


Figura. 1.10. Descripción de Pines para Tarjeta SM de 5V

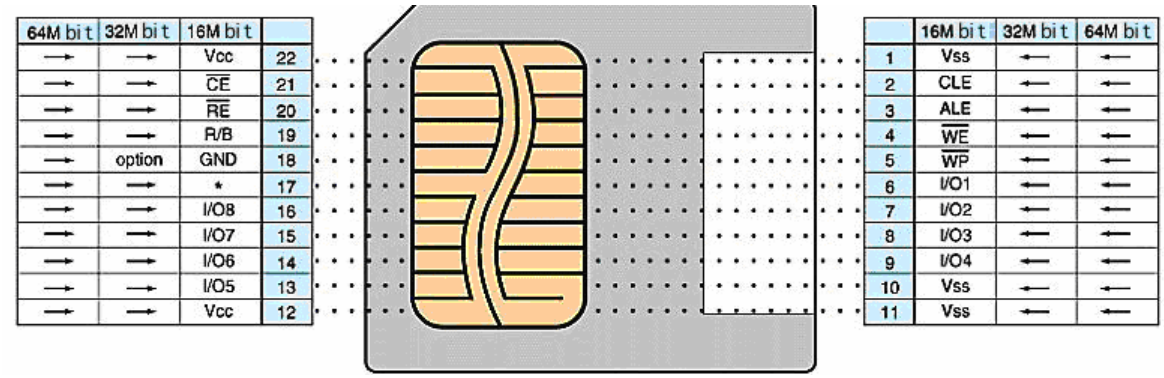


Figura. 1.11. Descripción de Pines para Tarjeta SM de 3.3V

Aplicaciones.

Sus principales aplicaciones se encuentran en el mundo del video y la fotografía, pero también SmartMedia puede usarse en dispositivos portátiles que requieren del almacenamiento de datos tomados en un equipo, esto hace que este siendo utilizada para video juegos, impresoras, y otros. Se emplean en dispositivos de compañías como Fuji, Minolta, Olimpo, Ricoh, Sanyo, JVC, y se las encuentra en las capacidades desde 2MB hasta más de 128MB.

Su tamaño no es una considerable ventaja en relación con otros formatos de menor dimensión, y también, su capacidad es relativamente baja. Podrían ser remplazadas por las tarjetas xD-Picture.

1.2.5. xD-PICTURE CARD

Figura. 1.12. xD-Picture Card

La xD-Picture[8], que se observa en la Figura 1.12, es una tarjeta de memoria flash desarrollada y presentada al mercado en el año 2002 por las marcas Olimpo y Fuji, y fabricada por la Corporación Toshiba. Actualmente las tarjetas xD están disponibles en un rango de capacidad desde 16 MB a 1 GB. La xD es una tarjeta de fácil uso por ser pequeña y liviana, con alta fiabilidad y precio bajo.

Una tarjeta xD de mejor rendimiento salió al mercado en el año 2005 y es conocida como xD Tipo M, y usa arquitectura de Célula Multinivel[9]. La tarjeta xD Tipo M se encuentra disponible en tamaños de hasta 8 GB, aunque las tarjetas tienen problemas de compatibilidad en cámaras digitales antiguas.

Algunas cámaras de Olimpo usan tarjetas CompactFlash además de xD, aunque el apoyo a CF está disminuyendo. La xD compite principalmente con los formatos Secure Digital, CompactFlash y Memory Stick.

[8] xD: Siglas de extreme digital.
[9] xD Tipo M: Siglas de Multi Level Cell.

Especificaciones Físicas.

En la Tabla 1.11 se observan las características físicas de las tarjetas de almacenamiento xD-Picture.[10] En la Figura 1.13 se muestran las dimensiones físicas de las tarjetas xD.

Tamaño	20mm x 25mm x 1.78mm.
Tipo de conector	18 pines

Tabla 1.11. Especificaciones Físicas de la Tarjeta xD-Picture

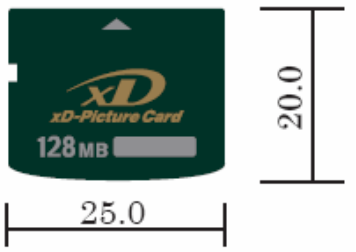


Figura. 1.13. Dimensiones Físicas de las Tarjetas xD (Medidas en mm)

Especificaciones Eléctricas.

Las tarjetas xD operan a 3.3 (V) como voltaje de polarización. No existe mayor información de libre acceso acerca de los parámetros eléctricos para este tipo de tarjetas.

Velocidad de Transmisión.

En la Tabla 1.12 se muestran las especificaciones de la velocidad de transmisión de las tarjetas xD-Picture.

[10] http://fft.fujifilm.com/, xD-Picture.

Velocidad de escritura para las tarjetas de 16 MB y 32 MB	1.3 MB/s
Velocidad de escritura para las tarjetas de 64MB y superiores	3 MB/s
Velocidad de lectura	5 MB/s

Tabla 1.12. Velocidad de Transmisión de la Tarjeta xD-Picture

Descripción de Pines.

La tarjeta xD posee un conector de 18 pines. La información de libre acceso no señala mayores detalles sobre la asignación de pines para esta tarjeta.

Aplicaciones.

El uso de la tarjeta xD para cámaras digitales está creciendo a un paso notable con más de 30% de crecimiento anual. Su principal aplicación se orienta hacia el almacenamiento de imágenes como datos, ya que es accesible para cualquier cámara digital que no necesita tener los derechos de propiedad intelectual sobre las imágenes que puede crear.

Su futuro se plantea como un medio únicamente dedicado al almacenamiento de imágenes, para lo cual Fuji y Olimpo han contribuido en un esfuerzo cooperativo para responder a las necesidades de consumidores que usan cámaras digitales.

1.2.6. TARJETA SECURE DIGITAL (SD)

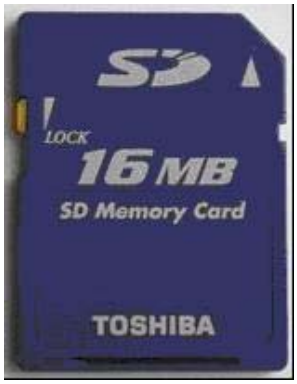


Figura. 1.14. Tarjeta SD

En la Figura 1.14 se muestra una tarjeta SD que fue diseñada para reemplazar a las tarjetas SM, como el dispositivo de memoria dominante en las cámaras digitales. La mayoría de fabricantes de cámaras digitales están tendiendo a usar SD en sus productos, entre las principales marcas figuran: Canon, Nikon, Kodak, Konica y Minolta. Mientras que, tres de las más grandes marcas, siguen usando sus propios formatos en sus cámaras, así Olimpo y Fuji usan la xD, y Sony usa Memory Stick.

El término “Secure” viene desde el origen de la tarjeta SD. Toshiba agregó el hardware de criptografía para aliviar a la industria de la música involucrada con graves problemas de piratería y de lo cual las tarjetas multimedia son muy susceptibles.

La tarjeta SD usa la " tecnología de la revocación " misma tecnología de protección en los DVD. Los contenidos digitales se registran con la propiedad intelectual de la tecnología de protección de IBM, Intel, Matsushita, y Toshiba. La Figura 1.15 muestra la descripción de protección de propiedad intelectual.

Figura. 1.15. Descripción de Protección de Propiedad Intelectual en Tarjetas SD

Especificaciones Físicas.

En la Figura 1.16 se observan las dimensiones físicas para las tarjetas SD.[11] En la Tabla 1.13 se muestran las especificaciones generales de las SD.

Figura. 1.16. Dimensiones Físicas de la Tarjeta SD

Tamaño:	32 mm x 24 mm x 2.1 mm.
Terminal de 9 pines.	
Tiene un interruptor mecánico para proteger los datos en contra de escritura.	
Pueden insertarse en los adaptadores para tarjetas MMC	

Tabla 1.13. Especificaciones Generales de la Tarjeta SD

[11] http://www.sdcard.org/sd_memorycard/index.html, Secure Digital (SD).

Especificaciones Eléctricas.

Las tarjetas SD operan en un rango de voltaje desde 2.7 (V) a 3.6 (V). Debido a esta característica se la considera como un dispositivo de bajo consumo de potencia.

Velocidad de Transmisión.

La tarjeta SD utiliza comunicación SPI[12]. La interfaz está basada en la comunicación serial a través de cuatro cables (reloj, entrada serial, salida serial, selección de chip) la cual es compatible con la mayoría de puertos SPI de los microcontroladores.

Las tarjetas SD poseen las siguientes velocidades de transmisión:

- **SD de baja velocidad:** Soportan tasas de transferencia de 0 a 400 KB/s en modo de transferencia un-bit.
- **SD de alta velocidad:** Soportan tasas de transferencia de 0 a 100 MB/s en el modo de cuatro-bit y 0 a 25 MB/s en el modo un-bit.

Descripción de Pines

La asignación de pines se observa en la Figura 1.17 y se describe en la Tabla 1.14 según las especificaciones recomendadas por la Asociación de Tarjetas SD.

[12] SPI: Siglas de Serial Peripherical Interface.

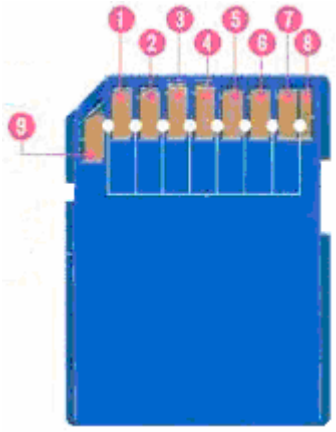


Figura. 1.17. Asignación de Pines de la Tarjeta SD

# de pin	Nombre del terminal
1	CD/DAT3
2	CMD
3	VSS
4	VDD
5	CLK
6	VSS
7	DAT0
8	DAT1
9	DAT2

Tabla 1.14. Descripción de Pines de la Tarjeta SD

MiniSD.

La tarjeta MiniSD fue desarrollada para cubrir las demandas de la industria ante la reducción de tamaño de los teléfonos móviles. A pesar de su tamaño ofrecen todos los beneficios de la tarjeta SD como se muestra en la Tabla 1.15.

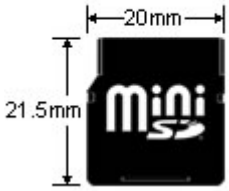
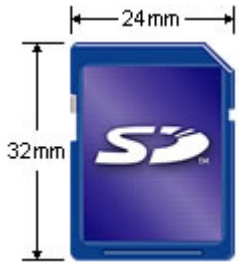
	MiniSD	SD
		
Ancho	20 mm	24 mm
Largo	21.5 mm	32 mm
Espesor	1.4 mm	2.1 mm
Peso (aproximadamente)	1 g	2 g
Rango de voltaje	2.7 - 3.6V	2.7 - 3.6V
Interruptor Write-protect	NO	SI
Protección de terminales	NO	SI
Número de pines	11 pines	9 pines

Tabla 1.15. Análisis Comparativo de las Tarjetas SD

Aplicaciones.

Las tarjetas SD están teniendo un gran auge en los últimos años, debido a que es un dispositivo pequeño y liviano, fácil de usar y tiene una gran capacidad de memoria para almacenar cualquier tipo de datos. Es compatible con casi todos los sistemas electrónicos.

Los dispositivos digitales están disminuyendo en cuanto al tamaño y aumentando la capacidad de memoria. Los fabricantes de SD están seguros que esta tendencia continuará y colocan SD como la gran solución, aunque el tamaño de la SD no es el más pequeño, sin embargo, la gran capacidad es esencial para adaptarse a una variedad de equipos digitales. Las SD están actualmente disponibles hasta en tamaños de 8GB.[13]

[13] Datos de libre acceso obtenidos de la SDA (SD Association).

1.2.7. MEMORIA FLASH USB

Figura. 1.18. Memoria Flash USB

Los avances en la tecnología para acceder a la información obligan a tener la posibilidad de transportar datos en dispositivos sumamente portátiles y esto impulsó la aparición de las memorias flash USB que se observa en la Figura 1.18. Estas memorias son aditamentos que pueden almacenar información necesaria para cualquier aplicación, siendo el único requerimiento una entrada USB, la misma que se encuentra integrada en la mayoría de los equipos nuevos.

Las memorias USB están formadas esencialmente por una memoria flash NAND. Estas memorias no necesitan una fuente de energía ya que funcionan con la corriente que proporciona el puerto USB. Como medida de seguridad, algunas memorias tienen la posibilidad de impedir la escritura mediante un interruptor.

Especificaciones Físicas.

Las memorias USB no tienen un estándar de dimensiones generales, dependen del fabricante y en el mercado se encuentran disponibles en diversos modelos para todos los requerimientos. En la Tabla 1.16 se muestran las características físicas de una memoria flash USB.

Tamaño:	depende del fabricante y la capacidad de memoria.
Terminal de 4 pines según el estándar USB.	

Tabla 1.16. Especificaciones Físicas de la Memoria Flash USB

Especificaciones Eléctricas.

Las memorias flash USB funcionan mediante al puerto USB del dispositivo al cual estén conectadas. El estándar USB estable que para la polarización se requiera de un voltaje de 5 (V) mientras que para las señales de comunicación el voltaje puede variar entre 3.3 (V) a 5 (V).

El conector USB proporciona, nominalmente, 5 V. En la práctica, el voltaje entregado puede caer debajo a 4 V. Un segmento dado del bus se especifica para entregar arriba de 500 mA.

El puerto USB transfiere señales y energía a los periféricos utilizando un cable de 4 hilos apantallado o no apantallado. En la Figura 1.19 se observa un esquema del cable, con dos conductores para alimentación y los otros dos para señal, debiendo estos últimos ser trenzados o no según la velocidad de transmisión.

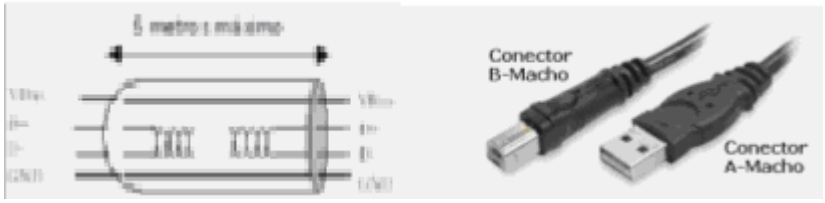


Figura. 1.19. Conector USB

Se transmiten las señales USB por un par trenzado de cables de datos, denominados D+ y D-. Éstos usan comunicación semiduplex para combatir los efectos de ruido electromagnético en las líneas más largas.[14]

Velocidad de Transmisión.

A continuación se presentan las especificaciones de comunicación para memorias flash USB:

- Interfase USB versión 1.1 o 2.0.
- **Memorias USB v2.0:** Alcanzan velocidades de escritura y lectura de hasta 480 Mbit/s teóricos.
- **Memorias con cable de 4 hilos apantallado v1.1:** Permite transmisiones a 12 Mbps tanto para lectura como escritura.
- **Memorias con cable de 4 hilos no apantallado v1.1:** Permite transmisiones a 1.5 Mbps tanto para lectura como escritura.

Descripción de Pines.

La Figura 1.20 se observa los terminales del conector USB. En la Tabla 1.17 se muestra la descripción de dichos terminales.

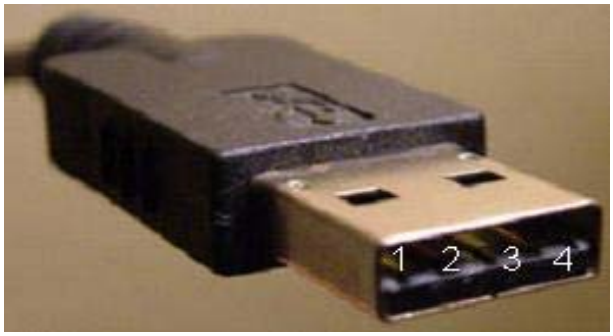


Figura. 1.20. Asignación de Terminales del Conector USB

[14] http://xataka.com/archivos/2005/11/08-memoria-flash-usb-en-forma-de-.php, Flash Memory USB.

Símbolo	Parámetro	Rango de unidad
1 Vcc	Voltaje de alimentación	3.3 a 5 V
2 D+	Dato serial	
3 D-	Dato serial	
4 Vss	Tierra	0

Tabla 1.17. Descripción de terminales

Aplicaciones.

Las memorias flash almacenan la información cuando no hay alimentación, y esto es una característica muy valorada para la múltiple variedad de usos en los que se emplea este tipo de memoria. El pequeño tamaño de las memorias USB es un factor determinante a la hora de escoger un dispositivo portátil, así como la ligereza y versatilidad para todos los usos hacia los que está orientada.

El costo de estas memorias es muy bajo respecto a las tarjetas multimedia, ofrece rendimientos y características muy superiores. Pero aún se enfrentan en diferentes campos con el disco duro que ofrece un costo por MB muy inferior a los que ofrece la memoria flash y, además los discos duros tienen una capacidad superior a la de las memorias flash.

CAPÍTULO II

TARJETA MULTIMEDIA MMC

2.1. INTRODUCCIÓN

Figura 2.1. Tarjeta MultiMediaCard

En la Figura 2.1 se observa una tarjeta MultiMediaCard (MMC) real de la compañía ATP. MMC fue lanzada al mercado en 1998, causando un gran impacto para los dispositivos portátiles destinados a aplicaciones multimedia. El éxito se debe a las múltiples características que ofrece la tarjeta MMC, siendo una de las principales su tamaño drásticamente menor. Fue diseñada para ser usada en reproductores de MP3, PDA, celulares de última generación y cámaras digitales de video.

El objetivo principal de los fabricantes de tarjetas MMC es lograr la regularización del uso de MMC en la industria, tomando como base un

estándar abierto. Con este fin, en Enero del año 1998 fue establecida la Asociación de MultiMediaCard.[15]

En 1999, se desarrolló una nueva tecnología para tarjetas MMC, a la cual se incorporaron funciones para el manejo de sistemas orientados en áreas específicas; así, es el caso de las funciones para protección de derechos de propiedad intelectual, lo que impulsó la aparición de la tarjeta Secure MultiMediaCard (Secure MMC), la cual mantiene la compatibilidad con todos los sistemas MMC y tiene la capacidad de encriptar los datos y protegerlos con una clave. Sus aplicaciones se dirigen hacia la industria de software y música, pero aun no consiguen afianzarse como el medio ideal para la venta debido al costo de la tarjeta.

Los fabricantes de tarjetas MMC están obligados a cumplir las especificaciones técnicas publicadas por la MMCA para las diferentes versiones de las tarjetas MMC. Estas especificaciones son analizadas y posteriormente sometidas a su aprobación por parte de un comité especializado, en el cual todos los fabricantes de tarjetas MMC tienen un representante.

2.2. DESCRIPCIÓN GENERAL

La tarjeta MMC fue diseñada para ser un medio de almacenamiento de datos y cubrir una amplia área de aplicaciones multimedias, como por ejemplo: fotografía, video digital, juguetes electrónicos y reproductores de audio.

La tarjeta MMC posee 7 pines, destinados para comunicación y alimentación. La comunicación está basada en el desarrollo de un sistema de tres líneas principales, si utiliza modo MMC. En el caso de operar por

[15] MMCA: Siglas en inglés de MultiMediaCard Association.

comunicación SPI[16], se usan 4 líneas, la adicional corresponde al CS (Chip Select). Esto permite la compatibilidad con los controladores existentes que de manera general operan con protocolo de comunicación SPI.

Los sistemas MMC proporcionan compatibilidad e integrabilidad entre todas sus tarjetas, sin importar la versión de su especificación, pero difieren en la capacidad de manejar archivos según las limitaciones del controlador (Host).

La versión más reciente para las tarjetas MMC es la v4.1, en la cual se ha aumentado la velocidad de transmisión de datos, a razón de ampliar el bus de datos de una a ocho líneas; de igual manera, esto incrementó el número de pines de siete a trece.

2.3. CARACTERÍSTICAS DE LAS TARJETAS MMC

Las características de las tarjetas MMC son parte de las especificaciones recomendadas por la MMCA, que contempla la posibilidad de operar con dos tipos de tarjetas:

- **MMC de bajo voltaje:** Trabajan en un rango de 1.65 a 1.95 (V). Algunas tarjetas MMC tienen la capacidad de trabajar para bajos y altos voltajes, esto es señalado por el fabricante en las hojas técnicas.
- **MMC de alto voltaje:** Su rango de voltaje es desde 2.7 a 3.6 (V).

[16] SPI: Serial Peripherical Interface.

Los rangos de voltaje para tarjetas MMC se muestran en la Tabla 2.1.

	MMC alto voltaje (V)	MMC bajo voltaje (V)
Señales del protocolo MMC	2.7 – 3.6	1.65 – 1.95
Alimentación		

Tabla 2.1. Rangos de Voltaje de Tarjetas MMC

Otras características de las tarjetas MMC son:

- Permite realizar solo lectura, lectura y escritura, y funciones como tarjeta I/O.
- Reloj variable entre 0 y 20 MHz.
- Contraseña para protección de datos.
- Corrección de errores presentes en memoria.
- Cache de datos opcional para optimizar el rendimiento en el tiempo de acceso.
- Características de protección de escritura integradas (permanente y temporal).
- Mecanismo para borrado de sectores.
- En modo SPI existen restricciones para emplear todas las funciones que dispone la tarjeta MMC.

Para evitar confusiones entre el modo de comunicación MMC y SPI se presenta la Tabla 2.2, donde constan sus principales diferencias.

	Modo MMC	Modo SPI
Interfaz	v3.1 o menor: *3 líneas.* CLK, CMD y DAT. v4.1: *10 líneas.* CLK, CMD y DAT0:7.	*Cualquier versión: 4 líneas* CLK, Data Input DI, Data Output DO y Chip Select CS
Frecuencia	0 – 20 MHz	0 –20 MHz
Acceso a memoria	Bloque simple Multi Bloque Cadena de datos	Bloque simple Multi Bloque
Selección de tarjeta	Por software en la identificación se asigna una dirección relativa a la tarjeta.	Por hardware con la señal CS.
Especiales	Protección en la transferencia de datos para evitar errores.	No disponible.

Tabla 2.2. Comparación entre el Modo MMC y SPI

2.4. CONCEPTO DE TARJETA MMC

El concepto de la tarjeta MMC está basado en la transferencia de datos por medio de un mínimo número de señales de comunicación. Estas señales son:

- **CLK:** Señal que con cada ciclo de reloj transmite un bit, que puede ser en la línea de comandos o en la línea de datos. La frecuencia puede variar entre cero y la frecuencia máxima de reloj del host.
- **CMD:** Esta señal es un canal que trasmite los comandos de una manera bidireccional. Es usado para la identificación de la tarjeta y para comunicar los comandos en una transferencia de datos. La señal CMD tiene dos modos de operación: Open-Drain, para el modo de identificación y el modo Push-Pull, para el modo de transferencia.
- **DAT:** Esta señal es un canal bidireccional para la transmisión de datos. La señal DAT funciona en modo Push-Pull.

Las tarjetas MMC según las funciones que proporcionan a un sistema se clasifican en:

- **Tarjetas de memoria sólo para lectura (ROM):** Estas tarjetas se fabrican con un contenido fijo de datos. Se utilizan típicamente como medios de distribución para software, audio, vídeo, etc.
- **Tarjetas de lectura y escritura (RW):** Estas tarjetas se venden típicamente como medios en blanco (vacíos) y se utilizan para almacenar datos como por ejemplo video, imágenes digitales o audio.
- **Tarjetas I/O:** Estas tarjetas se usan para comunicación, por ejemplo: módems.

Las tarjetas MMC se conectan directamente al bus MMC, pero es recomendado utilizar hardware externo para evitar corrientes parásitas. La Tabla 2.3 define los pines de las tarjetas MMC de versión 3.1 o menores.

Pin	Nombre	Tipo	Descripción
1	RSV	NC	Reservado.
2	CMD	E/S/PP/OD	Línea de comando.
3	Vss1	S	Suministro de tierra.
4	VDD	S	Suministro de voltaje.
5	CLK	E	Reloj.
6	Vss2	S	Suministro de tierra.
7	DAT	E/S/PP	Línea de datos.

* **S**: Suministro de energía; **E**: Entrada; **S**: Salida; **PP**: Push Pull; **OD**: Open Drain; **NC**: Sin conexión.
** La línea DAT para tarjetas de solo lectura solamente es una salida.

Tabla 2.3. Pines de la Tarjeta MMC

2.5. ESTRUCTURA INTERNA DE LA TARJETA MMC

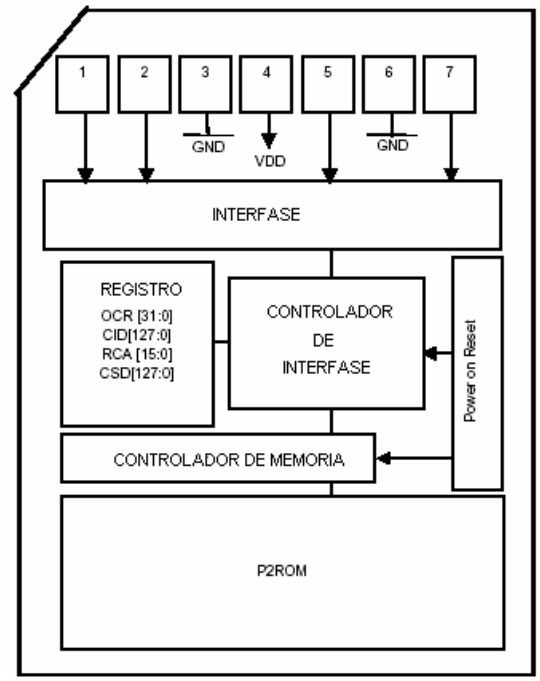


Figura 2.2. Diagrama de Bloques de la Tarjeta MMC

En la figura 2.2 se observa la estructura interna de una tarjeta MMC que está compuesta por diversos componentes electrónicos, entre los cuales se encuentran:

- **Controlador de memoria:** Es un dispositivo que está encargado de administrar los otros componentes de la tarjeta MMC según las instrucciones requeridas por el host. Se comunica con el controlador de interfase para comprobar que el formato de comando enviado es el correcto y de igual manera, enviar una respuesta con el formato adecuado.
- **Controlador de interfase:** Es un dispositivo que recibe la señal de los pines de la tarjeta MMC. Procesa la información analizando su contenido y se comunica con el controlador de memoria.
- **Interfase:** Es un circuito que permite transmitir o recibir las señales enviadas durante la comunicación con la tarjeta MMC.

- **Memoria flash (P2ROM):** Es el tipo de memoria establecida para almacenar la información introducida en la tarjeta MMC.
- **Power On Reset:** Es un circuito que indica si la tarjeta ha sido correctamente polarizada.
- **Registros:** Es un conjunto de registros donde se almacena la información de identificación de la tarjeta. Dentro de la tarjeta MMC se definen cinco registros: OCR, CID, CSD, RCA y DSR. Los registros de las tarjetas MMC se muestran en la Tabla 2.4.

Nombre	Longitud en bits	Descripción
CID	128	**Card Identification Register** Número de identificación de la tarjeta. Cada fabricante tiene su propio número.
RCA	16	**Relative Card Address Register.** Dirección relativa de la tarjeta asignada dinámicamente durante el proceso de identificación de la tarjeta.
DSR	16	**Driver Stage Register.** Configuración de los drivers de salida de la tarjeta.
CSD	128	**Card-Specific Data Register.** Información de como acceder a la tarjeta.
OCR	32	**Operation Conditions Register.** Información sobre las condiciones de operación de la tarjeta.

Tabla 2.4. Descripción de los Registros de la Tarjeta MMC

La información de estos registros es leída en el proceso de identificación de la tarjeta MMC, y en algunos casos puede ser modificada por medio de software.

2.5.1. DESCRIPCIÓN DE REGISTROS

OCR Operation Conditions Register.

OCR es un registro de 32 bits que define el rango de voltaje de operación de la tarjeta, es decir codifica en bits los voltajes de alimentación (VDD y Vss). Este registro incluye un bit de información de estado, llamado R/B (bit 31) que es activado si la tarjeta ha terminado su proceso de encendido. El registro OCR puede ser solo leído y se incluye en todas las tarjetas MMC. El contenido del registro OCR se muestra en la Tabla 2.5.

Bit OCR	Rango de voltaje de alimentación	Bit OCR	Rango de voltaje de alimentación
Bit [6:0]	Reservado	Bit 16	2.8 – 2.9
Bit 7	1.65 – 1.95	Bit 17	2.9 – 3.0
Bit 8	2.0 – 2.1	Bit 18	3.0 – 3.1
Bit 9	2.1 – 2.2	Bit 19	3.1 – 3.2
Bit 10	2.2 – 2.3	Bit 20	3.2 – 3.3
Bit 11	2.3 – 2.4	Bit 21	3.3 – 3.4
Bit 12	2.4 – 2.5	Bit 22	3.4 – 3.5
Bit 13	2.5 – 2.6	Bit 23	3.5 – 3.6
Bit 14	2.6 – 2.7	Bit [30:24]	Reservado
Bit 15	2.7 – 2.8	Bit 31	Bit de estado de encendido*

*: cuando el bit esta en bajo, la tarjeta no termina la rutina de encendido.

Tabla 2.5. Contenido del Registro OCR

Varios bits del registro OCR pueden ser activados, ya que corresponden al rango de voltaje que soporta la tarjeta. Es así que para una tarjeta MMC de alto voltaje se activarán los bits 15 al 23.

CID Card ID.

CID es un registro de 128 bits en el que se almacena la información de identificación de la tarjeta designada por el fabricante y es usado durante el proceso de identificación. El número CID es único para cada tarjeta MMC. En el protocolo de comunicación en modo MMC se usa enviando la dirección relativa asignada en el proceso de identificación para seleccionar la tarjeta de la cual se requiera su registro CID. La Tabla 2.6 muestra el contenido del registro CID:

Nombre	Campo	# de bits	Ubicación en CID
ID manufacturador	MID	8	[127:120]
ID aplicación	OID	16	[119:104]
Nombre del producto	PNM	48	[103:56]
Revisión del producto	PRV	8	[55:48]
Numero serial	PSN	32	[47:16]
Fecha de fabricación	MDT	8	[15:8]
CRC7	CRC	7	[7:1]
No es usado, simpre es "1"	–	1	[0]

Tabla 2.6. Contenido del Registro CID

Los campos son descritos a continuación:

- **MID**: Es un número binario de 8 bits que identifica al fabricante de la tarjeta. Este número es controlado, definido y asignado a un fabricante de tarjetas MMC por la MMCA. Este procedimiento se establece para asegurar la singularidad del registro CID.
- **OID:** Es un número binario de 16 bits que identifica la tarjeta OEM y/o los contenidos de la misma cuando es usada en la distribución ROM de datos multimedia. De la misma manera que el MID, este número es único y asignado a un fabricante por la MMCA.
- **PNM:** Es el nombre del producto en una cadena de 6 caracteres ASCII.
- **PRV:** Es la fecha de revisión del producto que está compuesta de dos dígitos en código binario decimal (BCD) de 4 bits cada uno, representando

el número de la revisión en formato de “n.m”. El “n” son los bits más significativos y “m” son los bits menos significativos. Como un ejemplo, el campo de valor binario para la revisión del producto “6.2” será: 0110 0010.

- **PSN**: Es un entero binario sin signo de 32 bits que representa el número de serie de la tarjeta MMC.
- **MDT**: Es un valor que indica la fecha de fabricación compuesta de dos dígitos hexadecimales de cuatro bits cada uno, representando la fecha en formato “m/y” (month/year, en español, mes/año). La “m” es el campo para el código del mes, empezando con 1 para Enero. La “y” es el código del año, que muestra 0 para el año 1997. Como un ejemplo, el valor binario del campo MDT para la fecha de producción “año 2000 mes abril” será: 0100 0011.
- **CRC:** Son operaciones que sirven para realizar el control de redundancia cíclica para CRC7 de 7 bits.

CSD Card-Specific Data.

El registro Card-Specific Data proporciona la información de cómo acceder al contenido de la tarjeta. El CSD define el formato de dato, tipo de corrección de error, máximo tiempo de acceso a datos, velocidad de transferencia, si el registro DSR puede usarse o no, etc. La parte programable del registro CSD puede cambiarse con el comando CMD27[17]. En la Tabla 2.7 se muestra el contenido del registro CSD.

[17] CMD27: Comando PROGRAM_CSD usado para programar el registro CSD. Recibe argumento cero, tipo de respuesta esperada R1. Ver Anexo 1.

Nombre	Campo	Ancho	Tipo	Bit en CSD
Estructura CSD	CSD_STRUCTURE	2	R	[127:126]
Especificación de versión del sistema	SPEC_VER	4	R	[125:122]
Reservado	–	2	R	[121:120]
Tiempo de acceso para lectura de datos 1	TAAC	8	R	[119:112]
Timer de acceso lectura de datos en ciclos CLK	NSAC	8	R	[111:104]
Máxima velocidad de transferencia	TRAN_SPEED	8	R	[103:96]
Clases de comando de la tarjeta	CCC	12	R	[95:84]
Dimensión máx. bloque de lectura datos	READ_BL_LEN	4	R	[83:80]
Bloques parciales permitidos para leer	READ_BL_PARTIAL	1	R	[79]
Desalineación bloque escritura	WRITE_BLK_MISALIGN	1	R	[78]
Desalineación bloque lectura	READ_BLK_MISALIGN	1	R	[77]
Implemen DSR	DSR_IMP	1	R	[76]
Reservado	–	2	R	[75:74]
Tamaño del dispositivo	C_SIZE	12	R	[73:62]
Corriente máx. lectura @ VDD min	VDD_R_CURR_MIN	3	R	[61:59]
Corriente máx. lectura @ VDD max	VDD_R_CURR_MAX	3	R	[58:56]
Corriente máx. escritura @ VDD min	VDD_W_CURR_MIN	3	R	[55:53]
Corriente máx. escritura @ VDD max	VDD_W_CURR_MAX	3	R	[52:50]
Multiplicador de tamaño del dispositivo	C_SIZE_MULT	3	R	[49:47]
Tamaño borrar sector	SECTOR_SIZE	5	R	[46:42]
Tamaño borrar grupo	ERASE_GRP_SIZE	5	R	[41:37]
Tamaño grupo de datos protegido	WP_GRP_SIZE	5	R	[36:32]
Habilitar grupo de datos protegido	WP_GRP_ENABLE	1	R	[31]
Fabricante ECC default	DEFAULT_ECC	2	R	[30:29]
Factor de velocidad de escritura	R2W_FACTOR	3	R	[28:26]
Dimensión máx. bloque escritura datos	WRITE_BL_LEN	4	R	[25:22]
Bloques parciales permitidos para escribir	WRITE_BL_PARTIAL	1	R	[21]
Reservado	–	5	R	[20:16]
Formato grupo de archivos	FILE_FORMAT_GRP	1	R/W	[15]
Bandera Copy (OTP)	COPY	1	R/W	[14]
Protección permanente escritura	PERM_WRITE_PROTECT	1	R/W	[13]
Protección temporal escritura	TMP_WRITE_PROTECT	1	R/W	[12]
Formato archivos	FILE_FORMAT	2	R/W	[11:10]
Cógido ECC	ECC	2	R/W/E	[9:8]
CRC	CRC	7	R/W/E	[7:1]
No usado, siempre es "1"	–	1	–	[0]

* R = Leíble, W = Escribible una vez, E = Borrable (múltiple escritura)

Tabla 2.7. Contenido del Registro CSD

A continuación se describen los campos del registro CSD más relevantes y los tipos de datos pertinentes:

- **CSD_STRUCTURE:** Es un número que describe la versión de la estructura CSD de la tarjeta MMC.

Estructura CSD	Versión estructura CSD	Especificación de versión del sistema
0	CSD version No. 1.0	Version 1.0 – 1.2
1	CSD version No. 1.1	Version 1.4 – 2.2
2	CSD version No. 1.2	Version 3.1
3	Reservado	Reservado

Tabla 2.8. Definición CSD_STRUCTURE

- **SPEC_VER:** Es un número que muestra las especificaciones técnicas de la versión que soporta la tarjeta MMC.

SPEC_VER	Versión para especificación del Sistema
0	Version 1.0 – 1.2
1	Version 1.4
2	Version 2.0 – 2.2
3	Version 3.1
4–15	Reservado

Tabla 2.9. Definición SPEC_VER

- **TRAN_SPEED:** Define el valor de la máxima velocidad a la cual se puede transmitir los datos.

TRAN_SPEED	Descripción
2:0	Velocidad transferencia Unidad: 000 = 100Kbit/s 001 = 1Mbit/s 010 = 10Mbit/s 011 = 10Mbit/s Otros = Reservado
6:3	Tiempo Valor: 0 = Reservado, 1 = 1.0, 2 = 1.2, 3 = 1.3, 4 = 1.5, 5 = 2.0, 6 = 2.5 7 = 3.0, 8 = 3.5, 9 = 4.0, A = 4.5, B = 5.0, C = 5.5, D = 6.0, E = 7.0, F = 8.0
7	Reservado

Tabla 2.10. Definición TRAN_SPEED

- **CCC:** Los comandos de las tarjetas MMC están divididos en subconjuntos (Command classes). Un valor de ‘1' en un bit CCC significa que el correspondiente Command Class es soportado por la tarjeta.

Bit	Descripción	Bit	Descripción	Bit	Descripción
0	Class 0	4	Class 4	8	Class 8
1	Class 1	5	Class 5	9	Class 9
2	Class 2	6	Class 6	10	Class 10
3	Class 3	7	Class 7	11	Class 11

Tabla 2.11. Descripción CCC

- **READ_BL_LEN:** Es un valor que muestra la longitud de bloque de datos que soporta la tarjeta MMC y está codificado como se muestra en la Tabla 2.12 para el rango de 1 a 2048 bytes.

Valor Bit	Descripción	Valor Bit	Descripción	Valor Bit	Descripción
0	2^0 = 1 byte	4	2^4 = 16 bytes	8	2^8 = 256 bytes
1	2^1 = 2 bytes	5	2^5 = 32 bytes	9	2^9 = 512 bytes
2	2^2 = 4 bytes	6	2^6 = 64 bytes	10	2^{10} = 1024 bytes
3	2^3 = 8 bytes	7	2^7 = 128 bytes	11	2^{11} = 2048 bytes
15:12	Resevado				

Tabla 2.12. Longitud de Bloques READ_BL_LEN

- **DSR_IMP:** Define si el registro DSR está integrado en la tarjeta MMC y si permite su manejo. Así, un valor DSR_IMP = 0 indica que no hay ninguna aplicación de DSR; mientras que, si DSR_IMP = 1 muestra que si se dispone de DSR.

- **C_SIZE:** Es un parámetro usado para calcular la capacidad de la memoria de la tarjeta MMC mediante los valores de bits C_SIZE_MULT y READ_BL_LEN. Los cálculos que recomiendan los fabricantes se muestran en la Tabla 2.13.

Capacidad de memoria = BLOCKNR * BLOCK_LEN	
Dónde:	
BLOCKNR = (C_SIZE+1) * MULT	
MULT = 2^C_SIZE_MULT+2	(C_SIZE_MULT <8)
BLOCK_LEN = 2^READ_BL_LEN	(READ_BL_LEN <12)

Tabla 2.13. Cálculos para la Capacidad de la Memoria Basados en C_SIZE

- **C_SIZE_MULT:** Es un parámetro usado para codificar un factor MULT para calcular el tamaño de memoria del dispositivo. El factor MULT se define como MULT = 2^C_SIZE_MULT + 2.

Valor Bit	Descripción	Valor Bit	Descripción
0	$2^4 = 4$	4	$2^6 = 64$
1	$2^3 = 8$	5	$2^7 = 128$
2	$2^4 = 16$	6	$2^8 = 256$
3	$2^5 = 32$	7	$2^9 = 512$

Tabla 2.14. Descripción de C_SIZE_MULT

- **R2W_FACTOR:** Define el tiempo típico de acceso al bloque de datos como un múltiplo del tiempo de acceso para lectura.

Valor R2W_FACTOR	Tiempo de acceso multiple de lectura
0	1
1	2
2	4
3	8
4	16
5	32
6, 7	Reservado

Tabla 2.15. Definición R2W_FACTOR

- **WRITE_BL_LEN:** Es un valor que establece la longitud de bloque para el proceso de escritura.

- **FILE_FORMAT_GRP:** Indica el formato del grupo de archivos seleccionados. Este campo es solo para leer (ROM). El uso de este campo se muestra en la descripción de FILE_FORMAT.

- **PERM_WRITE_PROTECT:** Protege permanentemente la información de la tarjeta en contra de cualquier proceso de escritura o borrado (todos los comandos de escritura y lectura para la tarjeta son permanentemente deshabilitados). El valor predefinido es cero; es decir, permanentemente no

protegida en contra de escritura. Se puede modificar el valor de este bit en el caso que requiera que sus datos sean permanentemente protegidos y podrá cambiar su contenido desactivando el valor de PERM_WRITE_PROTECT. En el caso que la aplicación sea orientada a la protección de propiedad intelectual no se podrá modificar el valor de este bit y solo podrá realizar la lectura de la tarjeta.

- **TMP_WRITE_PROTECT:** Protege temporalmente la información de la tarjeta en contra de cualquier proceso de escritura o borrado. Este bit puede ser activado o desactivado. El valor predefinido es ‘0 ', es decir no protegido en contra de escritura.

- **FILE_FORMAT:** Indica el formato de los archivos en la tarjeta. Este campo es solo para leer (ROM).

FILE_FORMAT_GRP	FILE_FORMAT	Tipo
0	0	Disco duro como sistema de archivos con tabla de partición.
0	1	DOS FAT (como floppy) solo sector Boot (sin tabla de partición).
0	2	Formato universal de archivo.
0	3	Otros / desconocido.
1	0, 1, 2, 3	Reservado

Tabla 2.16. Definición de FILE_FOMAT

- **CRC:** Los campos CRC llevan la suma del chequeo del contenido de CSD, el cual debe ser recalculado por el host para su comprobación (Ver Capítulo 2 Protección de errores).

RCA Relative Card Address.

RCA es un registro de 16 bits que almacena la información de la dirección asignada por el host durante el proceso de identificación. Esta dirección es usada para realizar la comunicación host-card. El valor predefinido del registro RCA es el 0x0001h. El valor de 0x0000h se reserva para desactivar todas las

tarjetas al estado Stand-By con el comando CMD7[18]. Al terminar el uso de la tarjeta MMC este valor se pierde.

DSR Driver Stage Register.

DSR es un registro de 16 bits que puede ser usado para mejorar el desempeño del bus MMC para operaciones largas. Esto depende de parámetros como la longitud del bus, tasa de transmisión y el número de tarjetas. Es un registro opcional y no es soportado por todas las tarjetas MMC.

2.6. DESCRIPCIÓN FÍSICA DE LAS TARJETAS MMC

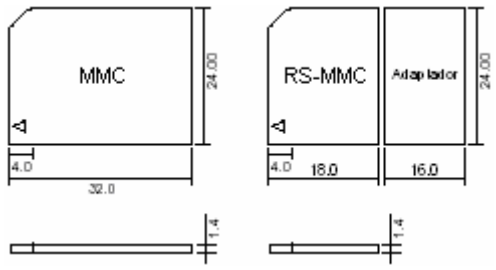


Figura 2.3. Dimensiones Físicas de las Tarjetas MMC

Las tarjetas MMC han sido desarrolladas en dos formas físicas como se muestra en la Figura 2.3. Las tarjetas prestan las mismas características eléctricas y de comunicación pero difieren en cuanto a su tamaño y a la capacidad de almacenamiento de datos.

[18] CMD7: Comando SELECT/DESELECT_CARD usado para cambiar el estado de la tarjeta de STAND-BY a transferencia de datos. Recibe argumento RCA, tipo de respuesta esperada R1. Ver Anexo 1.

El tamaño normal MMC tiene las siguientes dimensiones físicas: 24 mm x 32 mm x 1.4 mm. Es el formato más común y su mercado sigue siendo el más grande debido a la expansión que le han dado sus fabricantes.

El tamaño reducido RS-MMC (RS: Reduce Size) tiene las dimensiones de 24 mm x 18 mm x 1.4 mm, y está siendo muy utilizada en celulares debido a la reducción de su volumen en cerca de un 50%. Actualmente su capacidad de memoria está llegando a los 512MB. Para usar las tarjetas RS-MMC en un socket para tamaño normal se usa un adaptador especial.

2.7 CONCEPTO DE BUS MMC

El bus MultiMediaCard (bus MMC) ha sido diseñado para funcionar con la tarjeta como una memoria de estado sólido para el almacenamiento masivo o para desempeñarse como dispositivo I/O.

El bus MMC permite implementar sistemas de bajo costo con una alta tasa de transferencia de datos, a través de un solo bus maestro que permite trabajar con varias tarjetas con la asignación de una dirección relativa a cada una de ellas. Esto se realiza en el proceso de identificación. La comunicación en el bus MMC es bidireccional en las líneas de transmisión y puede ser llevada a cabo en el modo MMC o SPI.

2.7.1 LÍNEAS DEL BUS

La arquitectura del bus MMC permite que se conecten varias tarjetas MMC al mismo sistema de líneas. Ninguna tarjeta tiene una conexión individual al host o a otro dispositivo, lo que reduce los costos de conexión del sistema MMC.

Las líneas del bus se clasifican en tres grupos:

- **Suministro de alimentación:** VDD, VSS1 y VSS2, usadas para la alimentación de energía de las tarjetas (verificar en hojas técnicas).
- **Transferencia de datos:** CMD y DAT, usadas para la comunicación bidireccional.
- **Reloj:** CLK, usada para sincronizar la transferencia de datos a través del bus.

2.8. PROTOCOLO DEL BUS MMC

Después de encender el sistema, el host debe identificar las tarjetas MMC mediante un proceso de intercambio de mensajes basado en el protocolo de bus MMC. Cada mensaje es representado por una de las siguientes señales:

- **Comando (Command):** Es una señal que envía los comandos de una operación. Es transmitida serialmente desde el host a la tarjeta por la línea CMD.
- **Respuesta (Response):** Es una señal de respuesta que es enviada desde una tarjeta direccionada al host como una respuesta a los comandos recibidos anteriormente. Es transmitida serialmente por la línea CMD.
- **Datos (Data):** Son un paquete información que son transmitidos serialmente por la línea DAT desde la tarjeta al host o viceversa.

El direccionamiento de la tarjeta MMC lo realiza el microcontrolador (Host) en una sesión de dirección durante la fase de identificación a través del bus MMC, para todas las tarjetas conectadas.

Las tarjetas MMC son identificadas por su número CID. Este método requiere que cada tarjeta tenga un único número CID y con el fin de cumplir

este requisito se lee los 24 bits (MID y OID) que son definidos por la MMCA. Cada constructor de tarjetas solicita un número MID, que es único, y opcionalmente un OID.

Las operaciones de direccionamiento siempre contienen señales de comandos y respuestas. Además, algunas operaciones tienen una señal de datos, otras transfieren su información directamente dentro de una estructura de comando y respuesta. En caso de que se presente una operación sin datos los bits no serán transmitidos en ninguna de las líneas. Se definen dos tipos de transmisión de datos que se muestran a continuación:

- **Comandos secuenciales (Sequential commands):** Estos comandos inician un flujo continuo de datos y se termina cuando un comando de parada sigue en la línea CMD. De este modo se reduce al mínimo los comandos de cabecera.
- **Comandos orientados a bloques (Block-oriented commands):** Envían un bloque de datos seguido por bits CRC. Permiten la transferencia de datos en bloque simple o múltiple. Una transmisión de bloque múltiple se termina cuando un comando de parada sigue en la línea CMD, de manera similar a la operación en flujo.

Las Figuras 2.4 y 2.5 muestran los diferentes tipos de operaciones, en estas figuras las señales en gris corresponden a la información enviada por el host y las señales en blanco son los datos enviados por la tarjeta MMC.

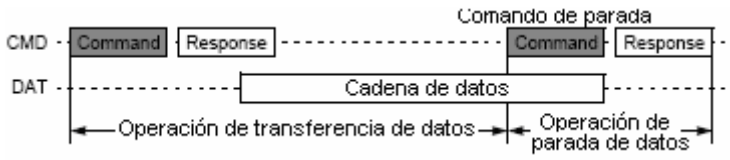


Figura 2.4. Operación Secuencial de Lectura

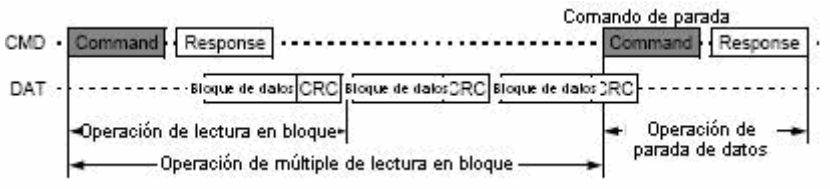


Figura 2.5. Operación de Lectura por Múltiple Bloque

2.8.1. FORMATO DE LA SEÑAL DE COMANDO (Command Token Format)

Los comandos tienen un formato de 48 bits. Para su transmisión son precedidos por un bit START (un nivel bajo en la línea CMD). A continuación se envía un bit de transmisión (un nivel alto en la línea CMD) y se sigue con los datos del comando a enviar (contenido) que están formados por un índice de 6 bits y un argumento de 32 bits. Luego se envían 7 bits de comprobación de CRC. El fin del envío del paquete de datos se señala con un bit END (un nivel alto en la línea CMD). La Figura 2.6 muestra el formato de una señal de comando.

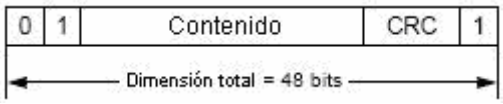


Figura 2.6. Formato de la Señal de Comando

La Tabla 2.17 presenta una descripción del formato general para la señal de comando.

Posición Bit	47	46	45:40	39:8	7:1	0
Ancho (Bits)	1	1	6	32	7	1
Valor	'0'	'1'	-	-	-	'1'
Descripción	Bit START	Bit de Transmisión	Indice de Comando	Argumento	CRC7	Bit End

Tabla 2.17. Descripción del Formato de la Señal de Comando

2.8.2. FORMATO DE LA SEÑAL DE RESPUESTA (Response Token Format)

Hay cinco tipos de señales de respuesta (R1 al R5) que tienen un código que se envía en un formato de 48 o 136 bits. Las respuestas tipo R1, R4 y R5 tienen un formato similar y su longitud es la misma, mientras que R2 y R3 tiene formatos independientes.

El formato es similar a las señales de comandos, con la diferencia que el bit de transmisión es un nivel bajo en la línea CMD para la comunicación tarjeta-host. La Figura 2.7 muestra los formatos de señales de respuesta:

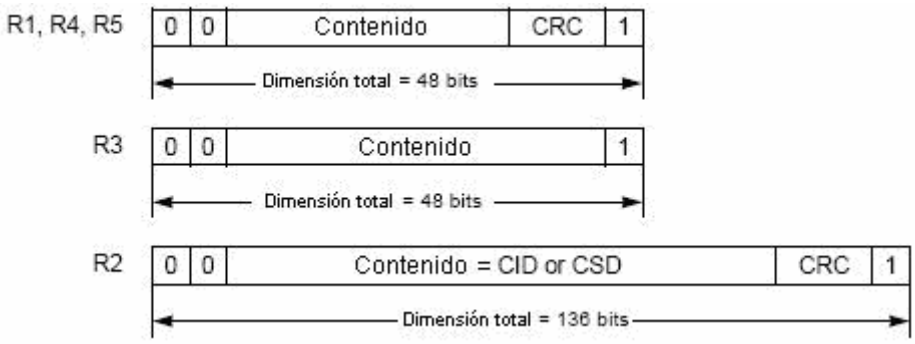


Figura 2.7. Formatos de las Señales de Respuesta

A continuación, se describe cada uno de los formatos de las señales de respuesta en las Tablas 2.18 a la 2.22.

- **Respuesta 1:** Respuesta normal. R1.

Posición Bit	47	46	45:40	39:8	7:1	0
Ancho (bits)	1	1	6	32	7	1
Valor	'0'	'0'	-	-	-	'1'
Descripción	Bit START	Bit de Transmisión	Indice de Comando	Estado de tarjeta	CRC7	Bit END

Tabla 2.18. Descripción del Formato de la Señal de Respuesta R1

- **Respuesta 2:** Registros CID y CSD. R2.

Posición Bit	135	134	[133:128]	[127:1]	0
Ancho (bits)	1	1	6	32	1
Valor	'0'	'0'	'111111'	-	'1'
Descripción	Bit START	Bit de Transmisión	Reservado	Argumento	Bit END

Tabla 2.19. Descripción del Formato de la Señal de Respuesta R2

- **Respuesta 3:** Registro OCR. R3.

Posición Bit	47	46	[45:40]	[39:8]	[7:1]	0
Ancho (bits)	1	1	6	32	7	1
Valor	'0'	'0'	'111111'	-	'1111111'	'1'
Descripción	Bit START	Bit de Transmisión	Reservado	Registro OCR	Reservado	Bit End

Tabla 2.20. Descripción del Formato de la Señal de Respuesta R3

- **Respuesta 4:** I/O. R4.

Posición Bit	47	46	[45:40]	[39:8]	[7:1]	0
Ancho (bits)	1	1	6	32	7	1
Valor	'0'	'0'	'100111'	-	-	'1'
Descripción	Bit START	Bit de Transmisión	Indice de comando	Argumento	CRC7	Bit END

Tabla 2.21. Descripción del Formato de la Señal de Respuesta R4

- **Respuesta 5:** R5.

Posición Bit	47	46	[45:40]	[39:8]	[7:1]	0
Ancho (bits)	1	1	6	32	7	1
Valor	'0'	'0'	'101000'	-	-	'1'
Descripción	Bit START	Bit de Transmisión	Indice de comando	Argumento	CRC7	Bit END

Tabla 2.22. Descripción del Formato de la Señal de Respuesta R5

2.8.3. FORMATO DE PAQUETES DE DATOS (Data Packet Format)

Hay dos tipos de paquetes de datos:

- **Flujo (Stream):** Cadena de datos que tiene una longitud indeterminada.
- **Bloque (Block)**: La información se transmite en paquetes de datos con una longitud fija que es configurable.

Cada dato es precedido por el bit Start (0) y sucedido por el bit End (1) como se observa en la Figura 2.8. Debido a que no hay un fin determinado en el flujo de datos, estos no incluyen el registro de protección CRC. El algoritmo de

protección CRC para los paquetes de datos en bloque es un polinomio CCITT[19] de 16 bits.

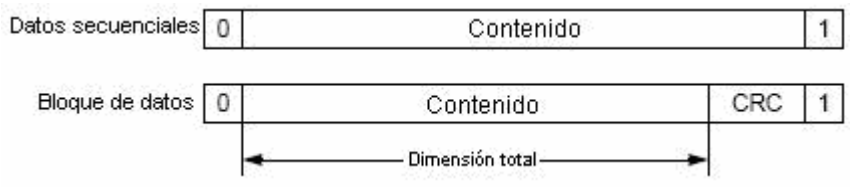


Figura 2.8. Formato de Señal de Datos

2.8.4. CONTROL DEL RELOJ

La señal de reloj del bus MMC puede ser usada por el Host para colocar a las tarjetas en modo de ahorro de energía, o para el control de flujo de datos en el bus. El host puede disminuir la frecuencia del reloj o apagarlo.

Existen restricciones que el host debe seguir, las cuales se describen a continuación:

- La frecuencia del bus puede ser cambiada en cualquier momento, pero existen restricciones para frecuencia máxima en los modos de identificación y transferencia de datos.
- El reloj debe estar activado para que exista transferencia de señales.
- Después de la última operación del bus MMC, el host debe enviar 8 ciclos de reloj a la tarjeta MMC para completar las operaciones antes de desconectarse. A continuación se exponen algunas recomendaciones de los fabricantes:
    - Un comando sin respuesta: 8 ciclos después del bit End (Host).
    - Un comando con respuesta: 8 ciclos después del bit End (tarjeta).

[19] CCITT: Son las siglas de Comité Consultivo Internacional Telegráfico y Telefónico (Consultative Committee for International Telegraphy and Telephony), antiguo nombre del comité de normalización de las telecomunicaciones dentro de la UIT ahora conocido como UIT-T.

- Operación lectura de datos: 8 ciclos después del bit End del último bloque.
- Operación escritura de datos: 8 ciclos después de la señal de status CRC.

2.9. PROTECCIÓN DE ERRORES.

El CRC (Cyclic Redundancy Code) ayuda a proteger los comandos, respuestas y transferencia de datos de la tarjeta MMC contra posibles fallos en los cálculos de CRC y ruido introducido en las líneas de transmisión que altere el valor de la trama de información.

Un CRC se genera para cada comando e inspecciona cada respuesta en las líneas de transmisión. Para los bloques de datos, un CRC es generado para descubrir los defectos en la trama de información. El host puede incluir códigos de corrección de errores, mientras que para dispositivos libres de errores este rasgo no se requiere.

Si una tarjeta MMC requiere una corrección de error externa (por medios externos fuera de la tarjeta), entonces un algoritmo de ECC[20] tiene que ser llevado a cabo en el controlador MMC.

[20] ECC: Códigos de correción de errores (Error-Correcting Codes) permiten reconstruir un mensaje que ha sufrido errores en la transmisión. Estos códigos están basados en principios matemáticos.

2.10. OTROS TIPOS DE TARJETAS MMC.

RS-MMC (REDUCE SIZE MMC).

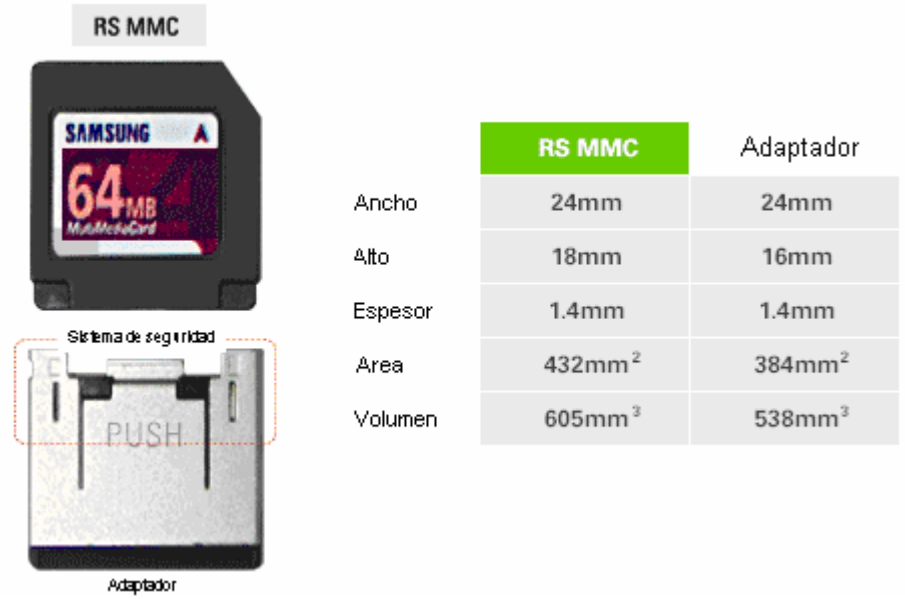
Figura 2.9. RS-MMC

En la Figura 2.9 se muestra una tarjeta RS-MMC de la compañía SanDisk. La tarjeta RS-MMC fue introducida en noviembre del año 2002 por la Asociación de MultiMediaCard (MMCA). Es aproximadamente la mitad del tamaño de la tarjeta MMC normal. RS-MMC ofrece todos los beneficios de MMC pero es casi 50% menor en su volumen. Como una extensión a la norma de MMC existente, RS-MMC permite a diseñadores de equipos electrónicos, elaborar dispositivos considerablemente más pequeños.

El crecimiento acelerado de la próxima generación de dispositivos móviles que requieren una gran capacidad de almacenamiento, mantendrá una alta demanda en el mercado de las tarjetas RS-MMC.

RS-MMC proporciona la versatilidad para integrar las funciones multimedias más ricas en los dispositivos, por eso se convierte en la principal aliada para la próxima revolución celular. Con la ventaja de las normas abiertas de RS-MMC, los fabricantes del dispositivo la pueden adoptar fácilmente. Los fabricantes más grandes de celulares como Nokia, Motorola, Samsung y Siemens apoyan

MMC y adoptarán RS-MMC en sus dispositivos futuros. Este nuevo mercado representa una oportunidad significante para RS-MMC. La Figura 2.10 muestra la vista y las dimensiones físicas de la tarjeta RS-MMC.



	RS MMC	Adaptador
Ancho	24mm	24mm
Alto	18mm	16mm
Espesor	1.4mm	1.4mm
Area	432mm^2	384mm^2
Volumen	605mm^3	538mm^3

Figura 2.10. Vista y Dimensiones de la Tarjeta RS-MMC

SECURE MMC.

La especificación para la tecnología de la tarjeta Secure MMC es compatible con la tarjeta existente MultiMediaCard. Sirve como protección para el almacenamiento seguro de datos y recuperación de información digital para las transacciones electrónicas móviles, independiente del dispositivo, servidor o red.

Su aplicación principal está en SDMI[21], que es un foro de más de 180 compañías y organizaciones que representan la tecnología de la información, consumidores electrónicos, tecnología de seguridad, industria magnetofónica

[21] SDMI: Siglas de Secure Digital Music Initiative.

mundial e Internet involucrados en proveer protección para la información multimedia digital.

2.10. APLICACIONES Y FUTURO EN EL MERCADO

El futuro en el mercado es muy alentador para las tarjetas MMC, ya que las altas capacidades (disponibles desde 16 MB a más de 2GB), alta velocidad de transmisión, bajo consumo de energía; las hace muy útiles para dispositivos portátiles.

La MMCA mantendrá el estándar abierto sin ninguna cuota por la licencia, pero existen pagos debido a la membresía anual y compra de especificaciones para cada versión. La tarjeta MMC se muestra como la mejor opción para los más de 100 miembros que representan a los principales fabricantes de controladores para tarjetas MMC, conectores, accesorios, etc.

Las aplicaciones de tarjetas MMC están orientadas a crear una compatibilidad con cámaras digitales, reproducción de archivos multimedia, teléfonos móviles, PDAs, y otros dispositivos que ofrecen sockets de MMC o SD. Así como también en comercialización de música por SDMI.

CAPÍTULO III

MÓDULO MMC DEL MICROCONTROLADOR ATMEL AT89C51SND1C

3.1. INTRODUCCIÓN

El microcontrolador AT89C51SND1C posee un núcleo (CORE) correspondiente al clásico microprocesador 8051 de la Intel. Sin embargo, a pesar de tener muchas similitudes con este y otros microcontroladores, sus módulos avanzados, lo convierten en un controlador moderno capaz de manejar las últimas tecnologías como por ejemplo MP3, USB, MMC y otros.

Para mayor información sobre el funcionamiento del microcontrolador AT89C51SND1C se deberá consultar el proyecto de grado titulado “Desarrollo de un Sistema Reproductor de Archivos MPEG I/II Layer 3 (MP3) Basado en el Microcontrolador ATMEL AT89C51SND1C” que constituye la primera fase de los proyectos relacionados con este microcontrolador.

En este capítulo se analizará el módulo MMC del microcontrolador ATMEL AT89C51SND1C y poniendo énfasis en los conceptos empleados en este proyecto. Este módulo funciona en base al estándar de comunicación MMC, que posee dos líneas de comunicación bidireccionales y una línea para la señal de reloj, las cuales se pueden observar en la Figura 3.1

Pin	Tipo	Descripción	Función alternativa
MCLK	S	Salida del Reloj MMC Reloj para transferir datos y comandos sincronizadamente	-
MCMD	E/S	Línea de comandos MMC Canal bidireccional de comandos que es usado para la inicialización de la tarjeta y transferencia de comandos. Para evitar cualquier corriente parásita, la entrada MCMD debe estar polarizada a \Vcc o \Vss.	-
MDAT	E/S	Línea de datos MMC Canal bidireccional para transferencia de datos. Para evitar cualquier corriente parásita, la entrada MDAT debe estar polarizada a \Vcc o \Vss.	-

Figura. 3.1. Señales de Comunicación del Módulo MMC

Dichas líneas del módulo MMC del microcontrolador son utilizadas para la comunicación con la tarjeta MMC y poseen la siguiente estructura interna que se describe en la Figura 3.2.

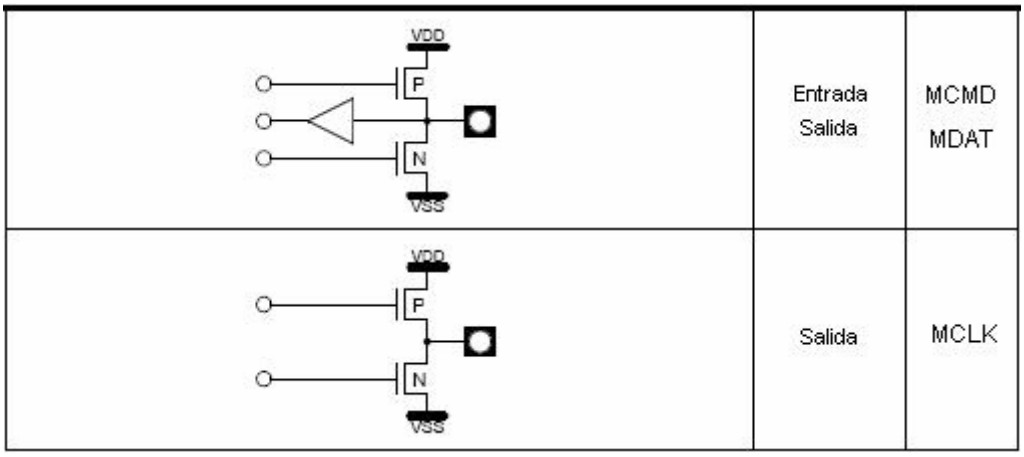


Figura. 3.2. Estructura Interna de Pines de Comunicaciones Modo MMC

De la figura anterior se observa que tanto la línea de comandos (MCMD) y la de datos (MDAT) son canales bidireccionales, la salida es tipo CMOS y la entrada posee un buffer; mientras que la línea de reloj (MCLK) es solo de salida y es de vital importancia para la sincronización que debe existir con los dos canales mencionados anteriormente.

3.2. DESCRIPCIÓN

La interfase del controlador MMC está vinculada con el núcleo C51, a través de los registros de funciones especiales que se muestran en la Tabla 3.1. Además en la Tabla 3.2 se indican las funciones de los bits de dichos registros.

Registros de control MMC:	MMCON0, MMCON1 y MMCON2
Registro de estado MMC:	MMSTA
Registro de interrupción MMC:	MMINT
Registro para máscara de interrupción MMC:	MMMSK
Registro de comandos MMC:	MMCMD
Registro de datos MMC:	MMDAT
Registro de reloj MMC:	MMCLK

Tabla. 3.1. Registros de Funciones Especiales

Mnemónico	Dir	Nombre registro	7	6	5	4	3	2	1	0
MMCON0	E4h	Control 0 MMC	DRPTR	DTPTR	CRPTR	CTPTR	MBLOCK	DFMT	RFMT	CRCDIS
MMCON1	E5h	Control 1 MMC	BLEN3	BLEN2	BLEN1	BLEN0	DATDIR	DATEN	RESPEN	CMDEN
MMCON2	E6h	Control 2 MMC	MMCEN	DCR	CCR	-	-	DATD1	DATD0	FLOWC
MMSTA	DEh	Control y estado MMC	-	-	CBUSY	CRC16S	DATFS	CRC7S	RESPFS	CFLCK
MMINT	E7h	Interrupción MMC	MCBI	EORI	EOCI	EOFI	F2FI	F1FI	F2EI	F1EI
MMMSK	DFh	Máscara de interrupción MMC	MCBM	EORM	EOCM	EOFM	F2FM	F1FM	F2EM	F1EM
MMCMD	DDh	Comandos MMC	MC7	MC6	MC5	MC4	MC3	MC2	MC1	MC0
MMDAT	DCh	Datos MMC	MD7	MD6	MD5	MD4	MD3	MD2	MD1	MD0
MMCLK	EDh	Divisor de reloj MMC	MMCD7	MMCD6	MMCD5	MMCD4	MMCD3	MMCD2	MMCD1	MMCD0

Tabla. 3.2. Descripción de los Bits de los Registros de Funciones Especiales del Módulo MMC

En la Figura 3.3 se observa que el controlador MMC se encuentra dividido en cuatro bloques:

- **Generador del reloj:** Es la etapa que administra la señal MCLK (siglas de MMC CLK) hacia la tarjeta.
- **Controlador de línea de comandos:** Está encargado de controlar el flujo de datos en la línea MCMD (siglas de MMC CMD) hacia o desde la tarjeta MMC.
- **Controlador de línea de datos:** Es el bloque que maneja el tráfico de datos en la línea MDAT (siglas de MMC DAT) hacia o desde la tarjeta MMC.

- **Controlador de interrupción**: Es el bloque que administra las fuentes que pueden generar una interrupción.

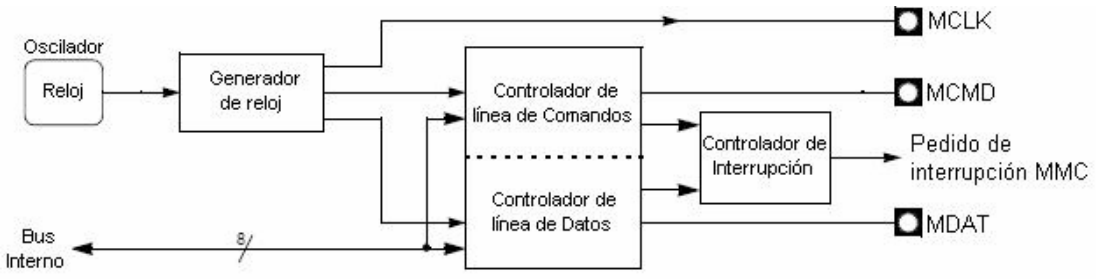


Figura. 3.3. Diagrama de Bloques del Controlador MMC

3.3. GENERADOR DE RELOJ

El reloj MMC se genera por la división del reloj del oscilador (Fosc) emitido por el bloque del controlador de reloj. El factor de división viene dado por los bits MMCD7:0 del registro MMCLK; así, un valor de 0x00 detiene el reloj MMC. Los bits MMCD7:0 pueden ser usados para aumentar o reducir dinámicamente el reloj MMC. La Figura 3.4 muestra el generador del reloj MMC y la fórmula para el cálculo del reloj MMCLK.

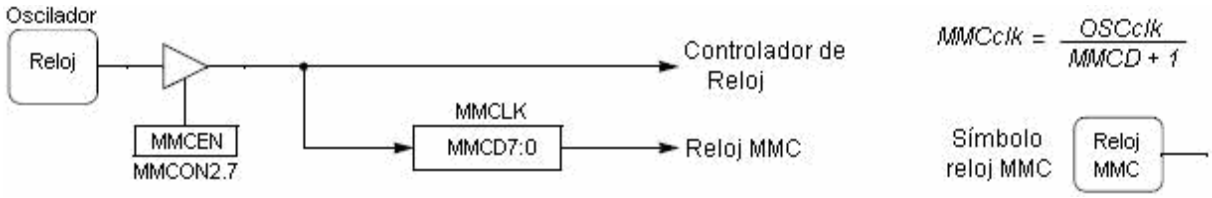


Figura. 3.4. Generador, Símbolo y Fórmula para el Cálculo del Reloj MMC

En cuanto el bit MMCEN en el registro MMCON2 es activado, la generación del Reloj MMC es habilitada. Esta señal es enviada a los controladores de las líneas

de comandos y de datos. En la Figura 3.5 se observa el diagrama de flujo para la configuración del reloj del Controlador MMC.

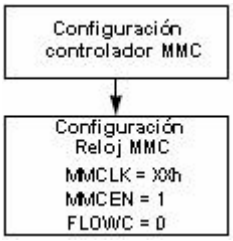


Figura. 3.5. Diagrama de Flujo para Configuración del Reloj del Controlador MMC

3.4. CONTROLADOR DE LA LÍNEA DE COMANDOS

El controlador de línea de comandos está dividido en 2 canales como se observa en la Figura 3.6. El canal transmisor de comandos maneja la transmisión de comandos hacia la tarjeta MMC a través de la línea MCMD; y el canal receptor de comandos maneja la recepción de la respuesta desde la tarjeta MMC a través de la línea MCMD.

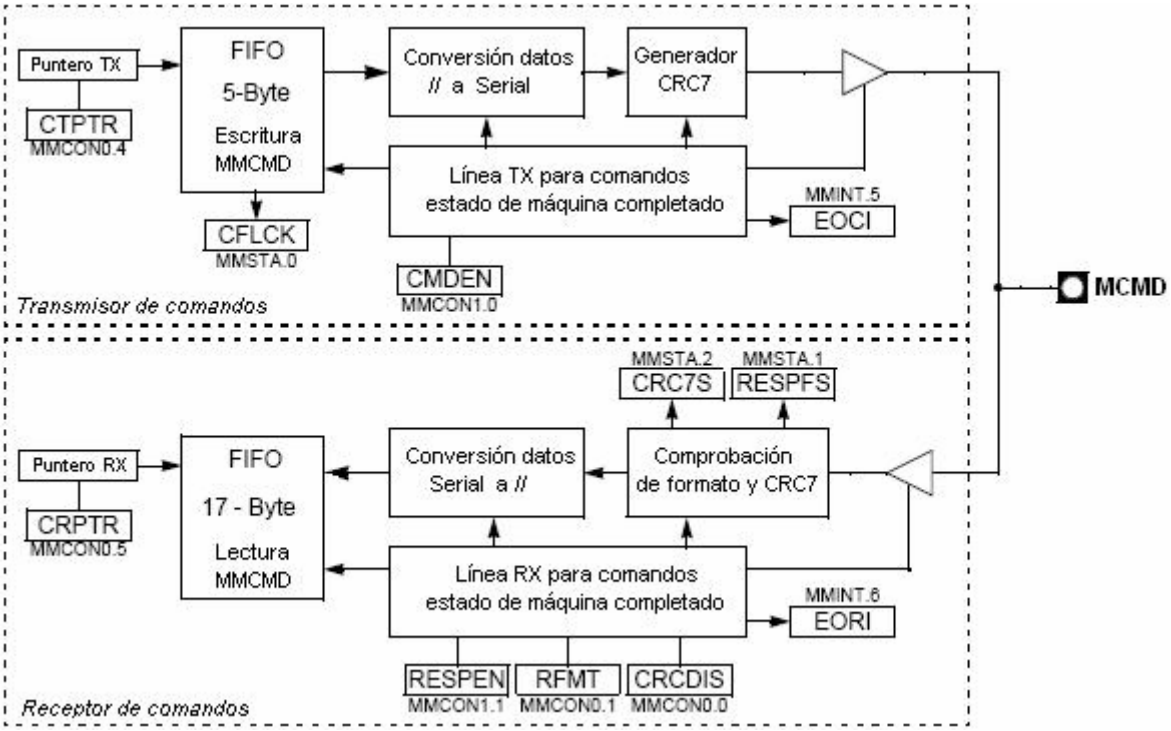


Figura. 3.6. Diagrama de Bloques del Controlador de Línea de Comandos

3.4.1. TRANSMISOR DE COMANDOS

Para enviar un comando a la tarjeta, se debe cargar en el comando de transmisión FIFO, el índice del comando (6 bits) y el argumento (4 Bytes), usando el registro MMCMD. Antes de empezar la transmisión se debe activar y desactivar el bit CMDEN en el registro MMCON1. Además se debe configurar primero:

- **RESPEN (MMCON1):** Es un bit que indica si se espera o no una respuesta.
- **RFMT (MMCON0):** Es un bit que indica el tamaño de la respuesta esperada.
- **CRCDIS (MMCON0):** Es un bit que indica si el CRC7 incluido en la respuesta será computado o no. Para evitar el error de CRC, CRCDIS puede ser puesto en “1” para respuestas que no incluyan CRC7.

En la Figura 3.7 se observa el diagrama de flujo para la transmisión de comandos.

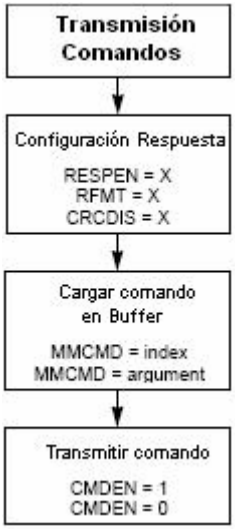


Figura. 3.7. Diagrama de Flujo para la Transmisión de Comandos.

En cuanto la transmisión de comandos es habilitada activando y desactivando el bit CMDEN que se encuentra en el registro MMCON1, la bandera CFLCK en el registro MMSTA se habilita automáticamente, indicando que la escritura en la FIFO está asegurada. Este mecanismo se lleva a cabo para evitar el desbordamiento del comando.

El fin de la transmisión de comandos es notificado por el microcontrolador mediante la activación de la bandera EOCI en el registro MMINT. Esta bandera puede generar una solicitud de interrupción MMC. La finalización de la transmisión de comandos también desactiva la bandera CFLCK.

Se puede abortar la carga de comandos habilitando y deshabilitando el bit CTPTR en el registro MMCON0, el cual desactiva el puntero de escritura a la FIFO de transmisión.

3.4.2. RECEPTOR DE COMANDOS

Cuando la tarjeta MMC ha recibido un comando, envía una respuesta la cual es señalada mediante la activación de la bandera EORI en el registro MMINT. Esta bandera puede generar una solicitud de interrupción MMC. Cuando esta bandera es activada, otras 2 banderas, en el registro MMSTA: RESPFS y CRC7S, dan un estado según la respuesta recibida:

- **RESPFS (MMSTA):** Es un bit que indica si el formato de respuesta es correcto o no, el tamaño es el esperado (48 bits o 136 bits) y si el “End bit” o bit de finalización valido ha sido recibido.
- **CRC7S (MMSTA):** Es un bit que indica si el cómputo de CRC7 es correcto o no.

Estas banderas son desactivadas cuando un comando es enviado a la tarjeta MMC y se actualizan cuando la respuesta ha sido recibida. Se puede abortar la lectura de la respuesta al habilitar y deshabilitar el bit CRPTR en el registro MMCON0, el cual desactiva el puntero de lectura de la FIFO de recepción.

De acuerdo a las especificaciones para tarjetas MMC, el proceso de espera (delay) entre un comando y una respuesta (parámetro conocido como N_{CR}) no puede exceder a 64 períodos de reloj MMC. Para evitar cualquier mal funcionamiento del controlador MMC cuando la tarjeta no envía su respuesta (por ejemplo, cuando se haya retirado la tarjeta del bus) se debe emplear un período de tiempo muerto para salir de tal situación. En el caso de caer en este tiempo muerto, se debe desactivar el controlador de comando y el estado interno de la misma con solo activar y desactivar el bit CCR en el registro MMCON2.

3.5. CONTROLADOR DE LA LÍNEA DE DATOS

El controlador de la línea de datos posee una FIFO de 16 Bytes, la cual es usada tanto por el canal de transmisión de datos como por el canal de recepción de datos, como se observa en la Figura 3.8.

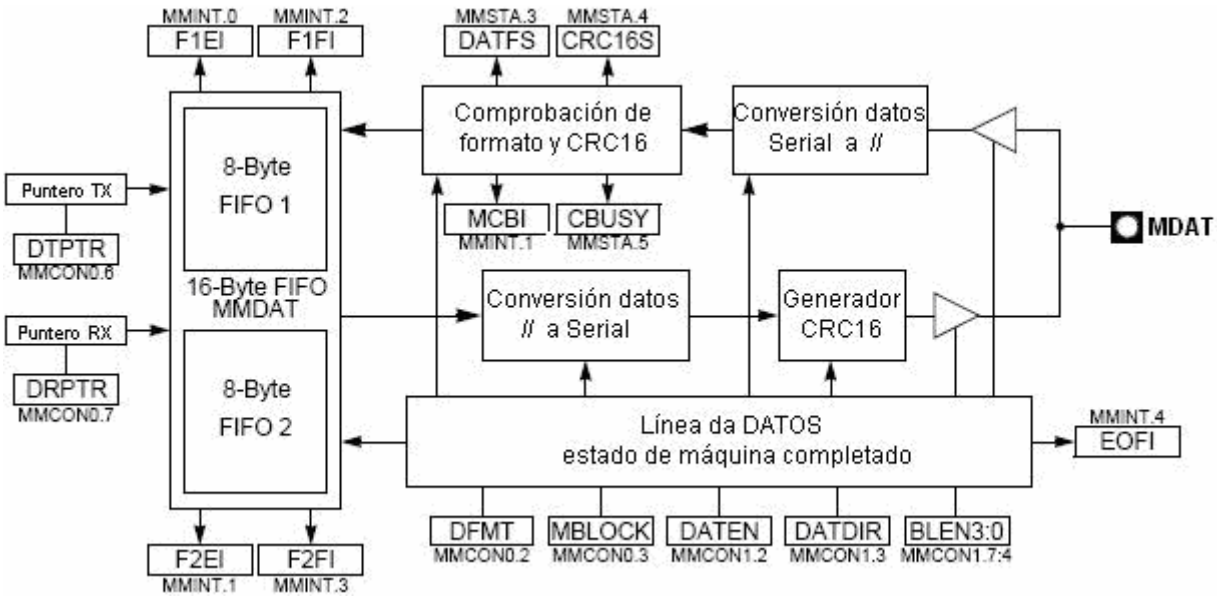


Figura. 3.8. Diagrama de Bloques del Controlador de Línea de Datos

3.5.1. IMPLEMENTACIÓN FIFO

La FIFO de 16 Bytes se basa en el manejo de una FIFO dual de 8 Bytes, 2 punteros y cuatro banderas; las cuales indican si el estado de cada FIFO es lleno o vacío.

Los punteros no son accesibles pero pueden desactivarse en cualquier momento mediante la habilitación y deshabilitación de los bits DRPTR y DTPTR del registro MMCON0. El desactivar los punteros es el equivalente a terminar el proceso de escritura o lectura de los datos.

Las banderas F1EI y F2EI que se encuentran en el registro MMINT son activadas automáticamente por el microcontrolador para señalar que la FIFO1 y la FIFO2 están vacías, respectivamente. Las banderas F1FI y F2FI que se encuentran en el registro MMINT son activadas automáticamente por el microcontrolador para señalar que la FIFO1 y la FIFO2 están llenas, respectivamente. Estas banderas pueden generar una solicitud de interrupción MMC.

3.5.2. CONFIGURACIÓN DE DATOS

Antes de enviar o recibir cualquier dato, el controlador de la línea de datos debe ser configurado según los diferentes tipos de transferencia de datos que se desee utilizar. Esto se lleva a cabo usando los siguientes bits:

- **DFMT (MMCON0):** Bit de formato de datos que al estar desactivado habilita el formato de cadena de datos (data stream format), mientras que activando el bit DFMT se habilita el formato de datos en bloque (data block format).
- **MBLOCK (MMCON0):** En el formato de datos en bloque, se debe configurar si desea el modo bloque simple, desactivando el bit MBLOCK o multi-bloque, activando el bit MBLOCK.
- **BLEN3:0 (MMCON1):** La longitud del bloque debe ser configurada usando los bits BLEN3:0 tal como se muestra en la Tabla 3.3.

BLEN3:0	**Longitud del bloque (Byte)**
BLEN = 0000 a 1011	Longitud = 2^{BLEN}: 1 a 2048
> 1011	Reservado: no programar para BLEN3:0 > 1011

Tabla. 3.3. Programación de la Longitud del Bloque

La Figura 3.9 resume los diagramas de flujo de configuración para los diferentes tipos de transferencia de datos.

Figura. 3.9. Diagramas de Flujo para la Configuración del Controlador de Datos

3.5.3. TRANSMISIÓN DE DATOS

3.5.3.1. CONFIGURACIÓN

Para transmitir datos a la tarjeta MMC se debe configurar primero el modo de transmisión en el controlador de datos, activando el bit DATDIR del registro MMCON1. La Figura 3.10 muestra el diagrama de flujo para la transmisión de datos en cadena para los modos de monitoreo (polling) e interrupción; mientras que la Figura 3.11 muestra el diagrama de flujo para la transmisión en bloque de datos en los modos de monitoreo e interrupción. Estos diagramas asumen que la longitud del bloque es mayor que 16 Bytes.

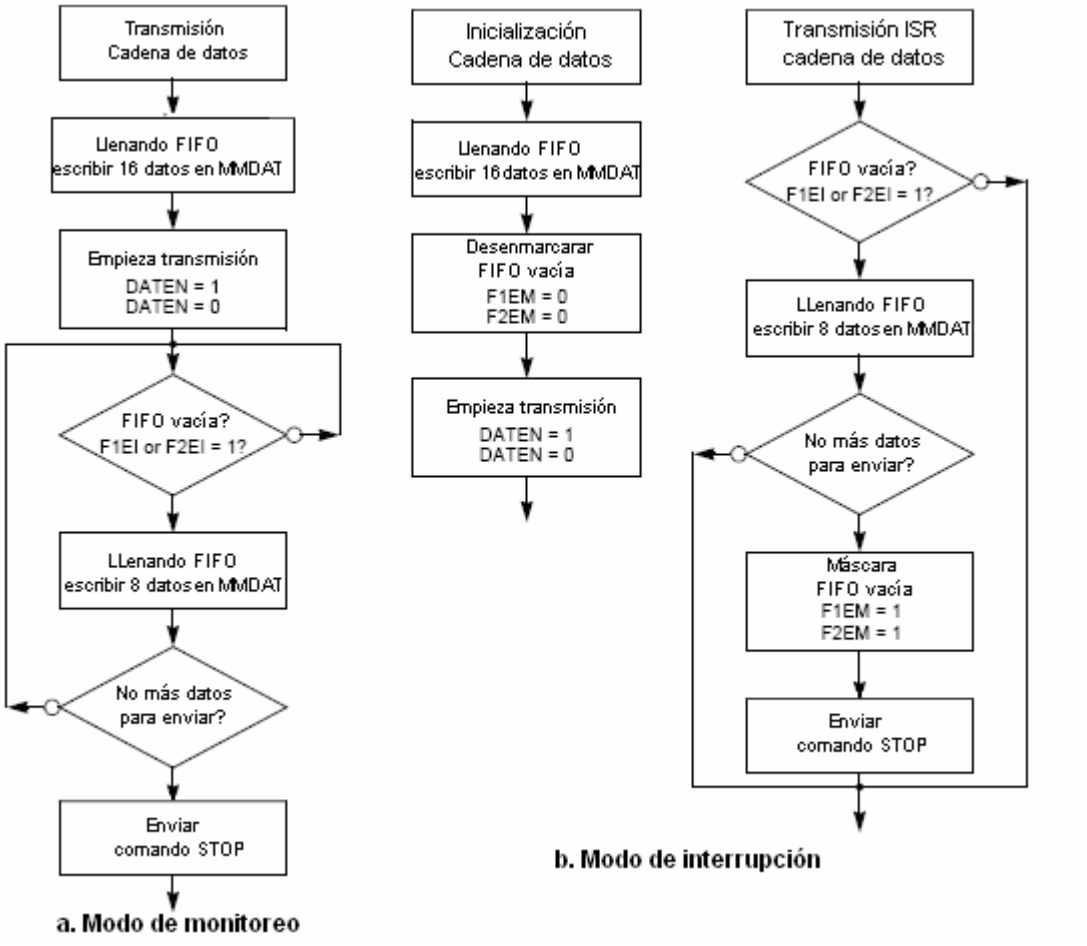


Figura. 3.10. Diagrama de Flujo para la Transmisión de Datos en Modo de Monitoreo e Interrupción

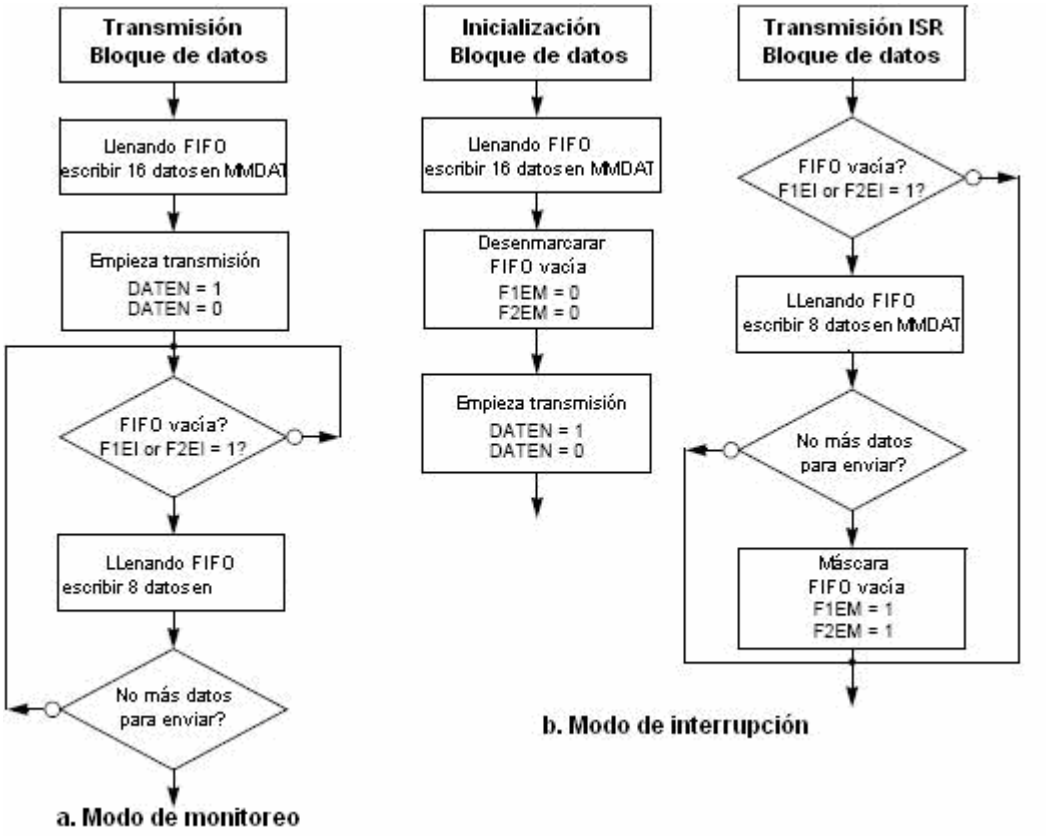


Figura. 3.11. Diagrama Flujo para la Transmisión de Datos en Modo Bloque

3.5.3.2. CARGA DE DATOS

Los datos son cargados en la FIFO escribiendo en el registro MMDAT. El número de datos cargados puede variar desde 1 a 16 Bytes. Entonces, si es necesario (enviar más de 16 Bytes) se debe esperar a que una de las FIFO este vacía (banderas F1EI o F2EI activadas) antes de cargar los siguientes 8 Bytes, ya que cada FIFO tiene un tamaño de 8 Bytes.

3.5.3.3. TRANSMISIÓN DE DATOS

La transmisión se habilita activando y desactivando el bit DATEN del registro MMCON1. Los datos son transmitidos inmediatamente, si la respuesta ha sido recibida; o son retrasados después de la recepción de la respuesta y la comprobación de estado es correcta. En ambos casos, la transmisión se retrasa si la tarjeta MMC envía un estado ocupado (estado busy) en la línea de datos; y permanece en retraso, hasta que se llegue al fin de esta condición.

De acuerdo a las especificaciones para tarjetas MMC, la transferencia de datos desde el host a la tarjeta MMC no puede empezar hasta que hayan transcurrido 2 períodos de reloj MMC (parámetro conocido como N_{WR}). Para direccionar todos los tipos de tarjeta, este retraso puede ser programado en pasos de 2 periodos de reloj MMC, usando los bits DATD1:0 del registro MMCON2 desde 3 períodos de reloj MMC cuando se desactivan estos bits, a 9 períodos de reloj MMC cuando los bits de DATD1:0 son activados.

3.5.3.4. FIN DE LA TRANSMISIÓN

El fin de la transmisión de una trama de datos (bloque o cadena) es señalada por la activación de la bandera EOFI en el registro MMINT. Esta bandera puede generar una solicitud de interrupción MMC.

En el modo cadena de datos, la bandera EOFI es activada después de la recepción del bit de finalización. Esto asume que previamente se ha enviado el comando de parada “STOP” a la tarjeta MMC, la cual es la única manera de detener la transferencia de datos en este modo.

En el modo bloque de datos, la bandera EOFI es activada después de la recepción del estado CRC. Otras 2 banderas en el registro de MMSTA: DATFS y CRC16S, informan el estado del envío de la trama.

- **DATFS (MMSTA):** Es un bit que indica si el formato del estado de la petición CRC es correcta o no.
- **CRC16S (MMSTA):** es un bit que indica si la tarjeta ha encontrado correcto o no el CRC16 del bloque.

3.5.3.5. ESTADO OCUPADO (BUSY)

La tarjeta MMC usa una petición Busy durante la operación de escritura de un bloque. Este “Busy State” o estado ocupado es informado a través de la bandera CBUSY en el registro MMSTA y por la bandera MCBI del registro MMINT que es activada cada vez que CBUSY cambia de estado, cuando la tarjeta MMC entra o sale de un estado ocupado “Busy State”. Esta bandera puede generar una solicitud de interrupción MMC.

3.5.4. RECEPTOR DE DATOS

3.5.4.1. CONFIGURACIÓN

Para recibir datos desde la tarjeta MMC, se debe configurar al controlador de datos en modo de recepción, desactivando el bit DATDIR del registro MMCON1. La Figura 3.12 muestra el diagrama de flujo para la recepción de datos en modo cadena, en los modos de monitoreo (polling) e interrupción; mientras que la Figura 3.13 resume el diagrama de flujo para la recepción de datos en modo bloque, para los modos de monitoreo e interrupción. Estos diagramas asumen que la longitud de bloque es mayor que 16 Bytes.

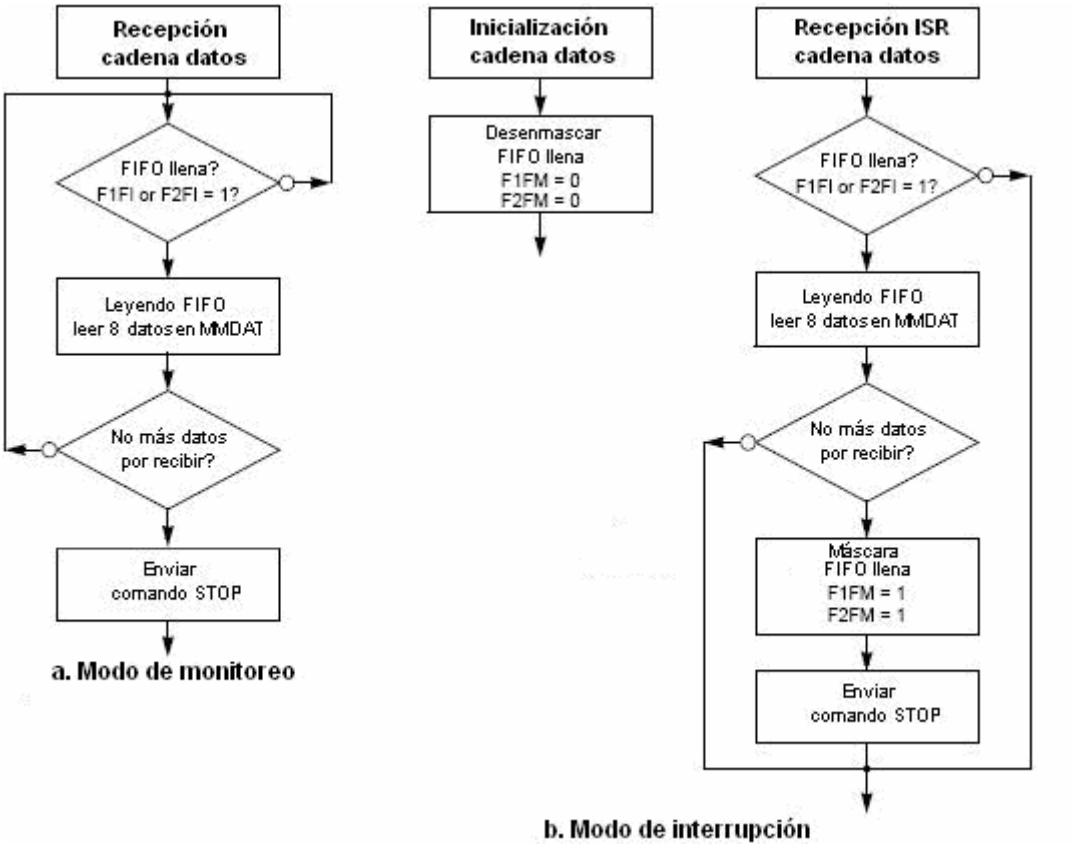


Figura 3.12. Diagrama de Flujo para la Recepción de Datos en Modos de Monitoreo e Interrupción

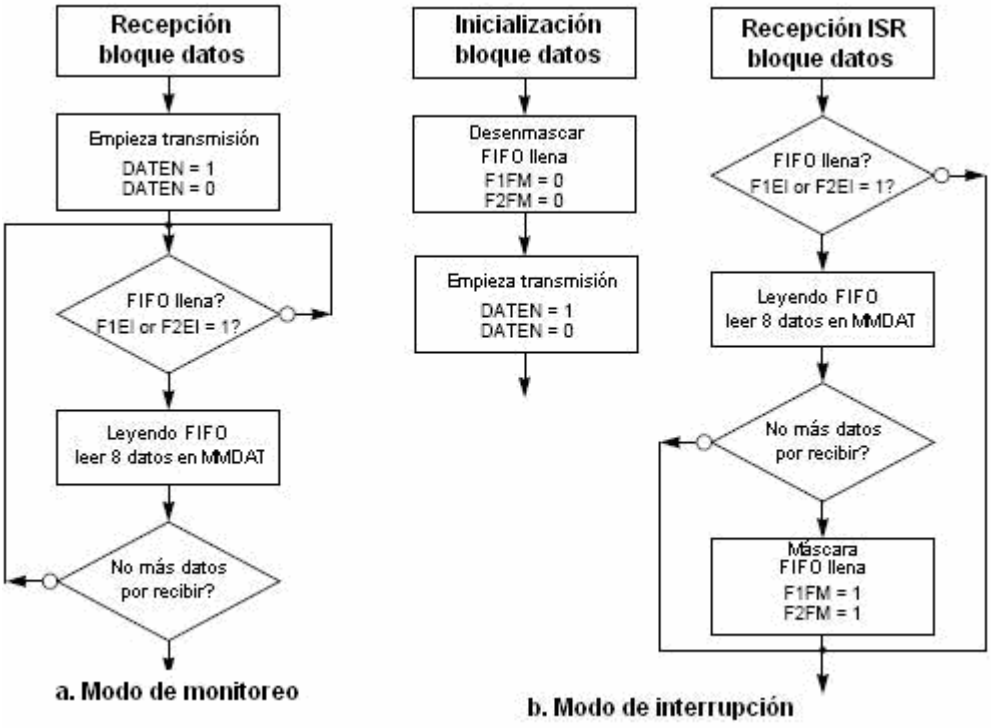


Figura 3.13. Diagrama de Flujo para la Recepción de Datos en Modo Bloque

3.5.4.2. RECEPCIÓN DE DATOS

El fin de la recepción de una trama de datos (bloque o cadena) es señalado por la por la activación de la bandera EOFI en el registro MMINT. Esta bandera puede generar una solicitud de interrupción MMC. Cuando esta bandera es activada, otras 2 banderas en el registro MMSTA, DATFS y CRC16S indican el estado de la trama recibida. En el caso de datos en cadena, CRC16S no tiene ningún significado y permanece desactivado.

De acuerdo a las especificaciones MMC, la transmisión de datos desde la tarjeta MMC empieza después de un retraso en el tiempo de acceso (parámetro conocido como N_{AC}) empezando desde el bit de finalización del comando de lectura. Para evitar cualquier problema o bloqueo del controlador MMC cuando la tarjeta no envía los datos, se debe ingresar a un período de tiempo muerto. En el caso de que el tiempo se exceda y se entre en modo sin operación, se puede restablecer el sistema activando y desactivando el bit DCR en el registro MMCON2. Este tiempo muerto puede deshabilitarse después de recibir 8 datos (bandera F1FI activada) o después del fin de la trama de recepción (bandera EOFI activada) en caso de que la longitud del bloque sea menor de 8 datos (1, 2 o 4).

3.5.4.3. LECTURA DE DATOS

Los datos son leídos desde la FIFO a través del registro MMDAT. Cada vez que una FIFO se llene (F1FI o F2FI se activan) se debe vaciar la FIFO leyendo 8 nuevos datos.

3.6. CONTROL DE FLUJO

Para permitir una alta velocidad en la transmisión sin tener problemas con la frecuencia del oscilador del CPU, el bit FLOWC en MMCON2 permite control en el flujo de los datos tanto para la transmisión como para la recepción.

Durante la transmisión, activando el bit FLOWC se produce los siguientes efectos:

- **MMCLK detenido:** Ambas FIFOs se vacíen: F1EI y F2EI activadas.
- **MMCLK reiniciado:** Una de las FIFOs se llena: F1EI o F2EI desactivadas.

Durante la recepción, activando el bit FLOWC se produce los siguientes efectos:

- **MMCLK detenido:** Ambas FIFOs se llenan: F1FI y F2FI activadas.
- **MMCLK reiniciado:** Una de las FIFOs se vacía: F1FI o F2FI desactivadas.

Tan pronto como el reloj es detenido, el bus MMC se detiene también, y permanece en este estado hasta que el reloj sea restablecido mediante la lectura o escritura de datos en MMDAT.

3.7. INTERRUPCIÓN

3.7.1. DESCRIPCIÓN

Como se puede observar en la Figura 3.14, el controlador MMC implementa ocho fuentes de interrupciones correspondientes a las banderas MCBI, EORI, EOCI, EOFI, F2FI, F1FI, F2EI, F1EI presentes en el registro MMINT. Todas estas fuentes son enmascarables por separado, usando los bits de máscara MCBM, EORM, EOCM, EOFM, F2FM, F1FM, F2EM y F1EM del registro MMMSK respectivamente.

Una petición de interrupción es generada cada vez que una bandera no enmascarable es activada y además es activado el controlador MMC global de interrupción (EMMC en el registro IEN1).

Cuando se realiza la lectura del registro MMINT, automáticamente se borra el valor de las banderas de interrupciones, esto implica que el contenido del registro debe ser almacenado y probar la bandera de interrupción para estar seguro de no olvidar ninguna interrupción.

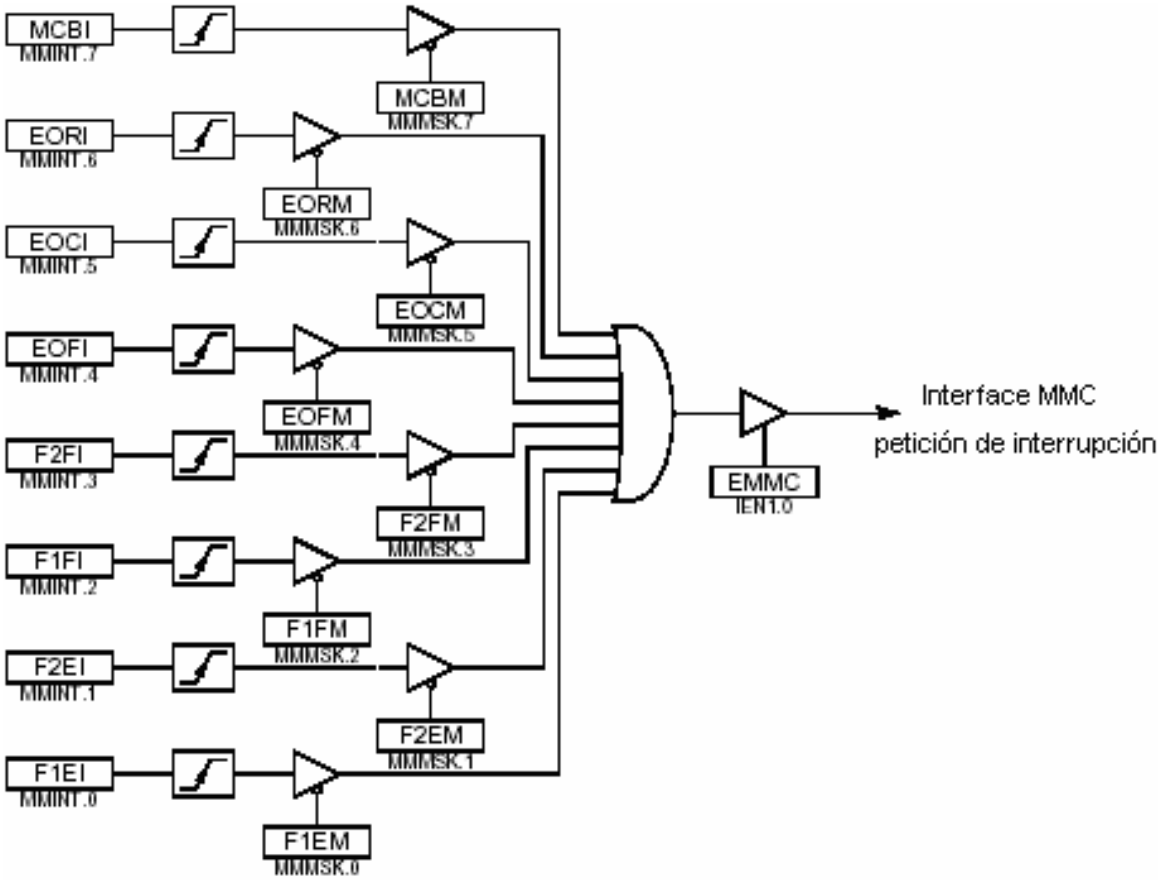


Figura. 3.14. Sistema de Interrupción del Controlador MMC

3.8. REGISTROS

A continuación se describen los registros de funciones especiales del módulo MMC del microcontrolador AT89C51SND1C y las condiciones de uso para los bits de dichos registros:

7	6	5	4	3	2	1	0
DRPTR	DTPTR	CRPTR	CTPTR	MBLOCK	DFMT	RFMT	CRCDIS

BIT	MNEMONICO	DESCRIPCION
7	DRPTR	Bit de reseteo del puntero de recepción de datos Setear para resetear el puntero de lectura de la FIFO de datos Resetear para liberar el puntero de lectura de la FIFO de datos
6	DTPTR	Bit de reseteo del puntero de transmisión de datos Setear para resetear el puntero de escritura de la FIFO de datos. Resetear para liberar el puntero de escritura de la FIFO de datos.
5	CRPTR	Bit de reseteo del puntero del comando de recepción Setear para resetear el puntero de lectura del comando de recepción FIFO. Resetear para liberar el puntero de lectura del comando de recepción FIFO.
4	CTPTR	Bit de reseteo del puntero del comando de transmisión Setear para resetear el puntero de escritura del comando de transmisión FIFO. Resetear para liberar el puntero de escritura del comando de transmisión FIFO.
3	MBLOCK	Bit de activación de bloque múltiple Setear para seleccionar un formato de datos de multiples bloques. Resetear para seleccionar un formato de datos un solo bloque.
2	DFMT	Bit de formato de datos Setear para seleccionar un formato de datos en bloque(data block format) Resetear para seleccionar un formato de datos en cadena.(stream data format)
1	RFMT	Bit de formato de respuesta Setear para seleccionar un formato de respuesta de 48-bits. Resetear para seleccionar un formato de respuesta de 136-bits.
0	CRCDIS	Bit para desactivar CRC7 Setear para deshabilitar el cómputo CRC7 cuando se reciba una respuesta. Resetear para habilitar el cómputo CRC7 cuando se reciba una respuesta.

VALOR DE RESET =0000 0000b

Tabla 3.4. Registro MMCON0

MMCON0 (S:E4h) – Registro de Control 0 MMC

7	6	5	4	3	2	1	0
BLEN3	BLEN2	BLEN1	BLEN0	DATDIR	DATEN	RESPEN	CMDEN

BIT	MNEMONICO	DESCRIPCION
7 - 4	BLEN3:0	Bits de tamaño de bloque Referirse a la Tabla 3.2 para descripción de bits No se debe programar valores mayores a 1011b
3	DATDIR	Bit de dirección de datos Poner en uno para seleccionar transferencia de datos desde el host hacia la tarjeta (modo de escritura). Poner en cero para seleccionar transferencia de datos desde la tarjeta hacia el host (modo de lectura).
2	DATEN	Bit de habilitación para transmisión de datos Setear y resetear para activar la transmisión de datos inmediatamente o después de que se haya recibido una respuesta
1	RESPEN	Bit de habilitación de respuesta Setear y resetear para habilitar la recepción de una respuesta seguido de la transmisión de un comando
0	CMDEN	Bit de habilitación de un comando de transmisión Seteary resetear para habilitar la transmisión del comando FIFO a la tarjeta.

VALOR DE RESET =0000 0000b

Tabla 3.5. Registro MMCON1

MMCON1 (S:E5h) –Registro de Control 1 MMC

7	6	5	4	3	2	1	0
MMCEN	DCR	CCR	-	-	DATD1	DATD0	FLOWC

BIT	MNEMONICO	DESCRIPCION
7	MMCEN	Bit de activación del reloj MMC Setear para habilitar el reloj MCLK y para activar el controlador MMC. Setear para desactivar el reloj MCLK y para detener el controlador MMC.
6	DCR	Bit para resetear el controlador de datos Setear y resetear este bit para resetear el controlador de la línea de datos en el caso de que una transferencia sea abortada.
5	CCR	Bit para resetear el controlador de comandos Setear y resetear este bit para resetear el controlador de la línea de comandos en el caso de que una transferencia sea abortada.
4-3	---	Reservado El valor leído en este bit es siempre cero.No setear este bit.
2-1	DATD1:0	Bits de retardo para transmisión de datos Este bit es usado para retardar la transmisión de datos después de una respuesta de 3 períodos de reloj MMC (todos los bits en "0") para 9 períodos de reloj MMC (todos los bits en "1") en pasos de 2 períodos de reloj MMC
0	FLOWC	Bit de control de Flujo MMC Setear para habilitar el control de flujo durante la transmisión de datos. Resetear para deshabilitar el control de flujo durante la transmisión de datos.

VALOR DE RESET =0000 0000b

Tabla 3.6. Registro MMCON2

MMCON2 (S:E6h) – Registro de Control 2 MMC

7	6	5	4	3	2	1	0
-	-	CBUSY	CRC16S	DATFS	CRC7S	RESPFS	CFLCK

BIT	MNEMONICO	DESCRIPCION
7 - 6	-	Reservado El valor leído en este bit es siempre cero.No setear este bit.
5	CBUSY	Bandera Busy de la tarjeta Seteado por hardware cuando la tarjeta envía un estado busy en la línea de datos. Reseteado por hardware cuando la tarjeta no envía mas un estado busy en la línea de datos.
4	CRC16S	Bit de estado CRC16 Modo de transmisión Seteado por hardware cuando una petición de respuesta reporta un bueno CRC. Reseteado por hardware cuando una petición de respuesta reporta un mal CRC. Modo de recepción Seteado por hardware cuando el CRC16 recibido en el bloque de datos es correcto. Reseteado por hardware cuando el CRC16 recibido en el bloque de datos es incorrecto.
3	DATFS	Bit de estado del formato de los datos Modo de transmisión Seteado por hardware cuando el formato de la petición de respuesta es correcta. Reseteado por hardware cuando el formato de la petición de respuesta no es correcta. Modo de recepción Seteado por hardware cuando el formato de la trama es correcta. Reseteado por hardware cuando el formato de la trama no es correcta.
2	CRC7S	Bit de estado CRC7 Seteado por hardware cuando el CRC7 computado en una respuesta es correcta. Reseteado por hardware cuando el CRC7 computado en una respuesta no es correcta. Este bit no es importante cuando se setea CRCDIS.
1	RESPFS	Bit de estado del formato de respuesta Seteado por hardware cuando el formato de una respuesta es correcta. Reteado por hardware cuando el formato de una respuesta no es correcta.
0	CFLCK	Bit de bloqueo del comando FIFO Seteado por hardware como señal al usuario para no escribir en el comando de transmisión FIFO: estado ocupado "busy state". Reeteado por hardware como señal al usuario que el comando de transmisión FIFO se puede utilizar: estado idle "idle state".

VALOR DE RESET =0000 0000b

Tabla 3.7. Registro MMSTA

MMSTA (S:DEh Solo de lectura) – Registro de Control y Estado MMC

7	6	5	4	3	2	1	0
MCBI	EORI	EOCI	EOFI	F2FI	F1FI	F2EI	F1EI

BIT	MNEMONICO	DESCRIPCION
7	MCBI	Bandera de interrupción cuando la MMC esta ocupada Seteado por hardware cuando la tarjeta entra o sale de un estado busy (cuando esta señal de ocupado es válida o no es válida en la línea de datos). Se resetea cuando se lee MMINT.
6	EORI	Bandera de interrupción para el fin de una respuesta Seteado por hardware al final de la recepción de una respuesta Se resetea cuando se lee MMINT.
5	EOCI	Bandera de interrupción para el fin de un comando Seteado por hardware al final de la transmisión de un comando Se resetea cuando se lee MMINT.
4	EOFI	Bandera de interrupción para el final de una trama Seteado por hardware al final de la transmisión de una trama (en cadena o bloque) Se resetea cuando se lee MMINT.
3	F2FI	Bandera de interrupción cuando la FIFO 2 se llena Seteado por hardware cuando la segunda FIFO se encuentra llena Reseteado por hardware cuando la segunda FIFO se encuentra vacía.
2	F1FI	Bandera de interrupción cuando la FIFO 1 se llena Seteado por hardware cuando la primera FIFO se encuentra llena Reseteado por hardware cuando la primera FIFO se encuentra vacía.
1	F2EI	Bandera de interrupción cuando la FIFO 2 se vacía Seteado por hardware cuando la segunda FIFO se encuentra vacía. Reseteado por hardware cuando la segunda FIFO se encuentra llena.
0	F1EI	Bandera de interrupción cuando la FIFO 1 se vacía Seteado por hardware cuando la primera FIFO se encuentra vacía. Reseteado por hardware cuando la primera FIFO se encuentra llena.

VALOR DE RESET= 0000 0011b

Tabla 3.8. Registro MMINT

MMINT (S:E7h Solo lectura) –Registro de Interrupción MMC

7	6	5	4	3	2	1	0
MCBM	EORM	EOCM	EOFM	F2FM	F1FM	F2EM	F1EM

BIT	MNEMONICO	DESCRIPCION
7	MCBM	Bit de interrupción enmascarable busy para tarjeta MMC Seteado para prevenir la bandera MCBI de generar una interrupción MMC. Resetear para permitir que la bandera MCBI genere una interrupción MMC.
6	EORM	Bit de interrupcción enmascarable para el fin de una respuesta Seteado para prevenir que la bandera EORI de generar una interrupción MMC Resetear para permitir que la bandera EORI genere una interrupción MMC.
5	EOCM	Bit de interrupcción enmascarable para el fin de un comando Seteado para prevenir que la bandera EOCI de generar una interrupción MMC Resetear para permitir que la bandera EOCI genere una interrupción MMC.
4	EOFM	Bit de interrupcción enmascarable para el fin de una trama Seteado para prevenir que la bandera EOFI de generar una interrupción MMC Resetear para permitir que la bandera EOFI genere una interrupción MMC.
3	F2FM	Bit de interrupcción enmascarable cuando la FIFO 2 este llena Seteado para prevenir que la bandera F2FI de generar una interrupción MMC Resetear para permitir que la bandera F2FI genere una interrupción MMC.
2	F1FM	Bit de interrupcción enmascarable cuando la FIFO 1 este llena Seteado para prevenir que la bandera F1FI de generar una interrupción MMC Resetear para permitir que la bandera F1FI genere una interrupción MMC.
1	F2EM	Bit de interrupcción enmascarable cuando la FIFO 2 este vacía Seteado para prevenir que la bandera F2EI de generar una interrupción MMC Resetear para permitir que la bandera F2EI genere una interrupción MMC.
0	F1EM	Bit de interrupcción enmascarable cuando la FIFO 1 este vacía Seteado para prevenir que la bandera F1EI de generar una interrupción MMC Resetear para permitir que la bandera F1EI genere una interrupción MMC.

VALOR DE RESET = 1111 1111b

Tabla 3.9. Registro MMMSK

MMMSK (S:DFh) – Registro Enmascarable de Interrupción MMC

7	6	5	4	3	2	1	0
MC7	MC6	MC5	MC4	MC3	MC2	MC1	MC0

BIT	MNEMONICO	DESCRIPCION
7 - 0	MC7:0	Byte de recepción de comando MMC Registro de Salida (lectura) de la respuesta FIFO. Byte de transmisión de comando MMC Registro de Entrada (escritura) del comando FIFO.

VALOR DE RESET = 1111 1111b

Tabla 3.10. Registro MMCMD

MMCMD (S:DDh) – Registro de Comando MMC

7	6	5	4	3	2	1	0
MD7	MD6	MD5	MD4	MD3	MD2	MD1	MD0

BIT	MNEMONICO	DESCRIPCION
7 - 0	MD7:0	Byte de dato MMC Registro de entrada (escritura) o salida (lectura) de los datos de la FIFO.

VALOR DE RESET = 1111 1111b

Tabla 3.11. Registro MMDAT

MMDAT (S:DCh) – Registro de Datos MMC

7	6	5	4	3	2	1	0
MMCD7	MMCD6	MMCD5	MMCD4	MMCD3	MMCD2	MMCD1	MMCD0

BIT	MNEMONICO	DESCRIPCION
7 - 0	MMCD7:0	Divisor de Reloj MMC Divisor de 8 bits para generaciones Reloj MMC

VALOR DE RESET =0000 0000b

Tabla 3.12. Registro MMCLK

MMCLK (S:EDh) – Registro de Divisor de Reloj MMC

CAPÍTULO IV

DISEÑO E IMPLEMENTACIÓN DE HARDWARE

4.1. INTRODUCCIÓN

La miniaturización de componentes electrónicos y el desarrollo de las técnicas de programación en línea (ISP) permiten una alta productividad, versatilidad y flexibilidad del proyecto planteado. El mismo que está basado en el desarrollo y análisis de un sistema que permita la lectura y escritura en las tarjetas MMC, teniendo como fuente de consulta los documentos técnicos de los fabricantes, hojas técnicas de datos y aplicaciones similares.

La investigación en esta etapa del proyecto está destinada a presentar todos los factores considerados en el diseño del hardware para la aplicación, donde se toman en cuenta los parámetros recomendados en los documentos técnicos recopilados de libre acceso que son publicados por los fabricantes de tarjetas MMC y microcontroladores con módulos MMC.

La solución general es planteada después de un proceso de optimización para reducir la circuitería necesaria sin descuidar los criterios de eficiencia y rendimiento del sistema a implementarse. El diseño final del hardware está basado en el microcontrolador AT89C51SND1, que es el centro de la aplicación, y al cual se conectan los periféricos necesarios para cumplir con todos los requerimientos planteados para el desempeño total del sistema lector y escritor de

tarjetas MMC. Este microcontrolador tiene módulos orientados a aplicaciones multimedia, como por ejemplo: MP3, MMC, USB, IDE, otros.

La implementación de este sistema ha sido elaborada en base al producto generado por el proyecto de grado titulado “Desarrollo de un Sistema Reproductor de Archivos MPEG I/II Layer 3 (MP3) Basado en el Microcontrolador ATMEL AT89C51SND1C”. Juntos pueden ser considerados como parte de un proyecto de mayor alcance para la reproducción y almacenamiento de archivos MP3.

La tarjeta MMC es una aliada para el desarrollo de sistemas de manejo de archivos de bajo costo, ya que necesita de muy poca circuitería externa y sus especificaciones son sencillas para la comprensión de diseñadores que empiezan en el manejo de estas tarjetas.

El estudio realizado contempla la necesidad de utilizar periféricos como medios para realizar diferentes tareas, así para la visualización es indispensable el uso de un LCD, ya que su manejo es bastante sencillo y común; mientras que, para el ingreso de ordenes e información, la mejor opción es implementar un teclado, en el cual se optimicen la cantidad de botones que disponga pero sin disminuir la capacidad del sistema operativo de brindar la selección de diferentes tareas que pueden ser ejecutadas.

4.2. DESCRIPCIÓN GENERAL DEL PROYECTO

El diseño de hardware está enfocado al desarrollo de un sistema que permita la lectura y escritura en tarjetas MMC, cumpliendo los requerimientos de que sea sencillo pero óptimo en el funcionamiento. Debido a que este proyecto se puede considerar parte de un proyecto más grande no se lo ha implementado en PCB[22]

[22] PCB: Printed Circuit Board.

sino en Protoboard, sin embargo cumple con el objetivo de este proyecto. La Figura 4.1 muestra los periféricos utilizados.

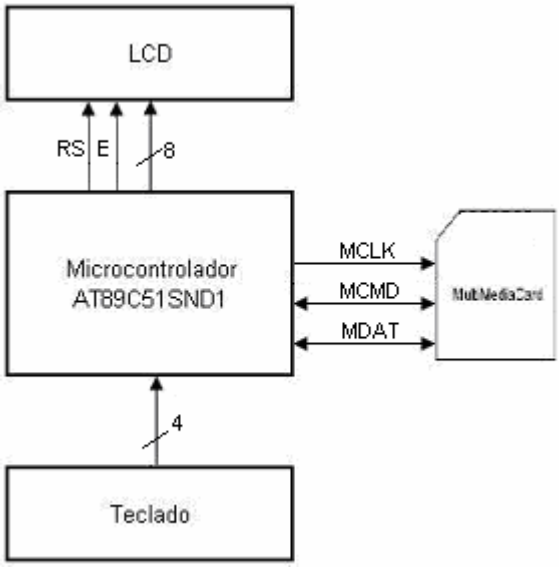


Figura 4.1. Diagrama de Bloques General

En la Figura 4.1 se observa que el microcontrolador ATMEL AT89C51SND1 es considerado como el cerebro de la aplicación, ya que alrededor de este se conectan el periférico de visualización, que permite mostrar los datos; y el bloque de teclado, con el cual se ingresan las órdenes al microcontrolador. El bloque MultiMediaCard es de mayor importancia dentro del objetivo de este proyecto en cuanto a los otros periféricos; pero en comparación con el AT89C51SND1 será considerado como un esclavo del microcontrolador, al igual que el resto de módulos incluidos en el propio chip.

4.2.1. MICROCONTROLADOR AT89C51SND1

El microcontrolador AT89C51SND1 de la compañía ATMEL ha sido desarrollado con la capacidad de ser compatible con la mayoría de dispositivos de almacenamiento masivo que existen en la actualidad; así que posee un módulo

interno para manejo de tarjetas MMC, y otros dispositivos de amplio uso en el mercado. Las características principales del microcontrolador AT89C51SND1 son las siguientes:

- Decodificador de MPEG I/II capa 3.
- Configuración programable para interfaz con un DAC, en los formatos PCM e I2S.
- Frecuencia máxima igual a 20 MHz.
- Memoria RAM interna de 2304 Bytes.
- Memoria para Código de programa de 64KB.
- Memoria Flash para Boot[23] de 4KB vía ISP.
- USB versión 1.1.
- Compatibilidad con interfaz SPI en modo maestro y esclavo.
- Interfaz IDE / ATAPI.
- Interfaz de software con tarjetas SmartMedia.
- Full duplex UART.
- Dos canales de 10 bits para ADC.
- Condiciones de operación:
    - 3 (V) ± 10% a 25mA a un temperatura ambiente de 25 ºC.
    - Rango de temperatura entre -40 ºC a +85 ºC.
- Encapsulados:
    - TQFP80.
    - BGA81.
    - PLCC84, utilizado en el proyecto.

El microcontrolador AT89C51SND1, está basado en la Arquitectura Intel con núcleo C51, al cual se le han incorporado diversos módulos que cumplen con las aplicaciones más relevantes para el estudio y manejo de microcontroladores.

[23] Boot: Arranque Bootloader.

El módulo MMC del microcontrolador presenta una restricción para la aplicación en todas las tarjetas MMC, debido a que solo puede manejar tarjetas con formato de archivos FAT16 por la longitud del bloque de datos que soporta (máximo 512 Bytes), lo cual limita el manejo de archivos; pero no impide realizar el proceso de identificación de las tarjetas MMC porque solo se realiza en base al intercambio de señales de comandos y reconocimiento de registros MMC.

El microcontrolador AT89C51SND1 necesita de poco hardware externo para su total y óptimo desempeño; la Figura 4.2 muestra las señales internas y externas que recibe el microcontrolador y las que envía para el funcionamiento del módulo MMC.

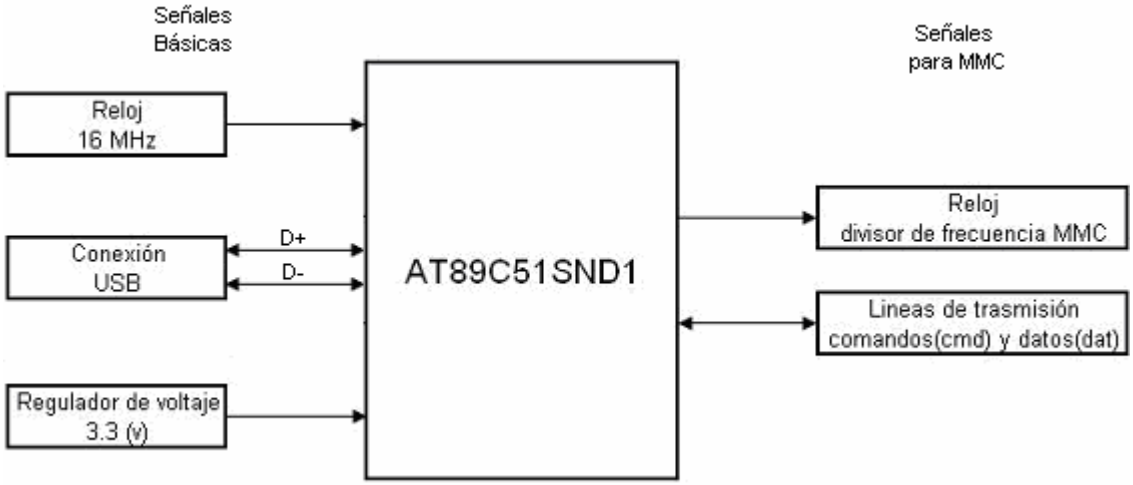


Figura 4.2. Señales Internas y Externas del Microcontrolador

Las señales que constan a la izquierda del microcontrolador AT89C51SND1 son básicas e indispensables para cualquier aplicación; mientras que las señales de la derecha, son las específicas para la comunicación con tarjetas MMC a través del módulo MMC propio del microcontrolador.

4.2.1.1. RELOJ

El controlador del reloj del microcontrolador AT89C51SND1 está basado en un oscilador on-chip y un oscilador on-chip de Phase Lock Loop (PLL). El reloj PLL es empleado para generar otras frecuencias para periféricos y otros módulos, como el MP3 y DAC. La conexión está basada en un cristal que provee un ciclo de pulsos al microcontrolador AT89C51SND1, también el reloj es usado para sincronizar los periféricos conectados para cualquier aplicación.

Consideraciones para el diseño.

El cristal puede tener un valor de hasta 20 MHz que soporta el microcontrolador AT89C51SND1; para esta aplicación se designa el valor de 16MHz por consideraciones realizadas en los tiempos de acceso a la tarjeta MMC en relación con la frecuencia máxima planteada para el reloj MMC que es de 8MHz (8000kHz). Así queda definido el valor de FOSC igual a 16.000KHz para los cálculos de sincronización posteriores.

En este caso, no es necesario sincronizar el reloj PLL debido a que la frecuencia del reloj MMC es calculada en relación directa con la frecuencia de FOSC. La Figura 4.3 muestra la conexión recomendada por ATMEL para la conexión del reloj con un cristal.

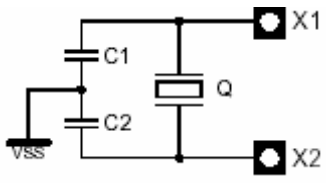


Figura 4.3. Diagrama de Conexión Recomendado por ATMEL para el Reloj

Esta conexión se la dirige a los pines X1 (27) y X2 (29) del microcontrolador AT89C51SND1, donde X1 es la entrada normal para el reloj y X2 es la entrada que mejora el rendimiento del núcleo C51, debido a que este microcontrolador puede trabajar con 6 ciclos de reloj por tiempo de máquina en lugar de 12. Esto se lo realiza habilitando el bit X2 en el registro CKCON.

Los valores de los capacitores son una recomendación de las hojas técnicas de ATMEL y la Tabla 4.1 muestra las consideraciones que se deben tener en cuenta para designar sus valores, así como también las características del cristal, en lo que respecta a su frecuencia y valores de resistencia y capacitancia.

Símbolo	Parámetro	Mínimo	Típico	Máximo	Unidad
CX1	Capacitor X1 a Vss		10		pF
CX2	Capacitor X2 a Vss		10		pF
Ct	Carga total capacitores X1 - X2		5		pF
DL	Nivel de potencia			50	uW
F	Frecuencia del cristal			20	MHz
Rcristal	Resistencia del cristal			40	Ω
Ccristal	Capacitancia del cristal			6	pF

Tabla 4.1. Características para el Cristal y Capacitores de la Etapa del Reloj

La conexión recomendada para la implementación del circuito completo del reloj se muestra en la Figura 4.4.

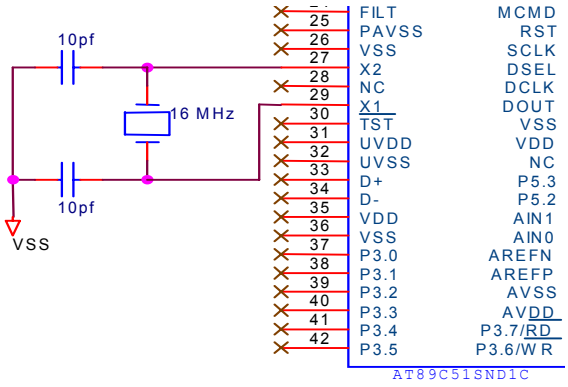


Figura 4.4. Diagrama de Conexión del Reloj

4.2.1.2. CONEXIÓN USB

El microcontrolador AT89C51SND1 implementa un módulo para dispositivos con interfaz USB, que es muy usado en la actualidad. El módulo USB permite trabajar con una alta tasa de transmisión de datos y principalmente es empleado en aplicaciones orientadas a:

- Dispositivos de almacenamiento masivo USB.
- Dispositivos HMI.
- Aplicaciones de Firmware.

La conexión USB cobra gran importancia, al ser el medio que sirve de interfaz entre el prototipo y una PC. A través de éste se carga en el microcontrolador el código de programa (Firmware) correspondiente a la aplicación. El programa que realiza esta tarea es de la compañía ATMEL y su nombre es FLIP (Flexible In-System Programmer) versión 2.4.4.

ATMEL FLIP v2.4.4 permite establecer la comunicación para la programación en línea (ISP de sus siglas en inglés In-System Programming), y posee características como:

- Manejo de interfaces CAN, RS-232, LPC y USB.
- Reconoce todos los microcontroladores de ATMEL.
- Permite la edición de archivos hexadecimales.
- Permite borrar, leer, grabar y verificar los datos en los microcontroladores.
- Permite conocer el tamaño del código del programa y la capacidad máxima de memoria para código.
- Permite conocer si existe offset y modificarlo.
- Permite administrar niveles de seguridad para acceder al código del programa.

En la Figura 4.5 se muestra la pantalla principal del programa ATMEL FLIP versión 2.4.4.

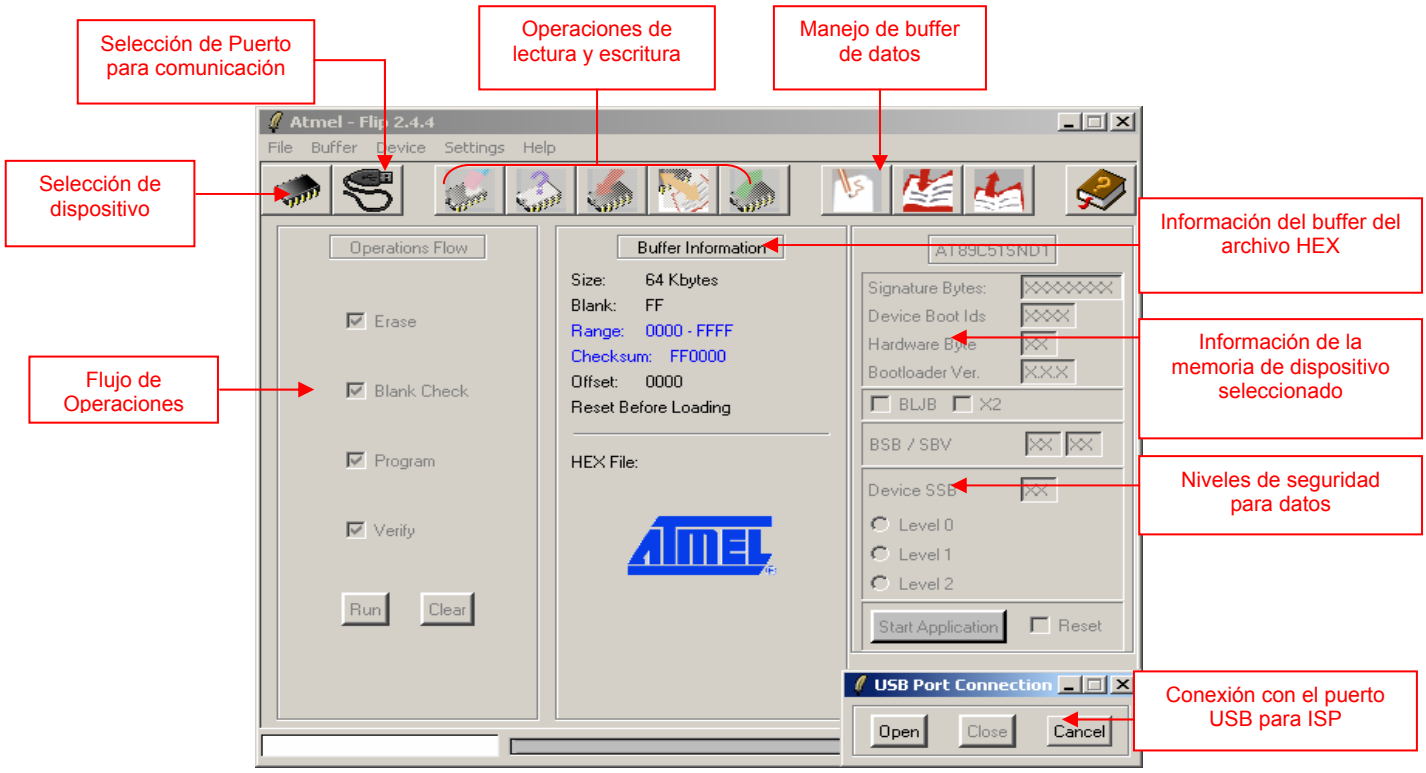


Figura 4.5. ATMEL FLIP v2.4.4

Consideraciones de diseño para la etapa USB.

El diseño se realizó en base al manejo estándar de USB, donde se recomienda diseñar el circuito con resistencias de 27Ω en los canales D+ y D-. Estas señales corresponden a los pines 33 y 34 respectivamente del microcontrolador AT89C51SND1.

Es importante señalar que el interfaz USB posee su propia polarización a 5 (V) por lo que solo se debe realizar la conexión a tierra del bus USB para evitar que los 5 (V) dañen a los otros periféricos o al bus de datos del microcontrolador

AT89C51SND1 que funcionan a 3.3 (V). En la Figura 4.6 se muestra el diagrama de conexión del interfaz USB en el microcontrolador AT89C51SND1.

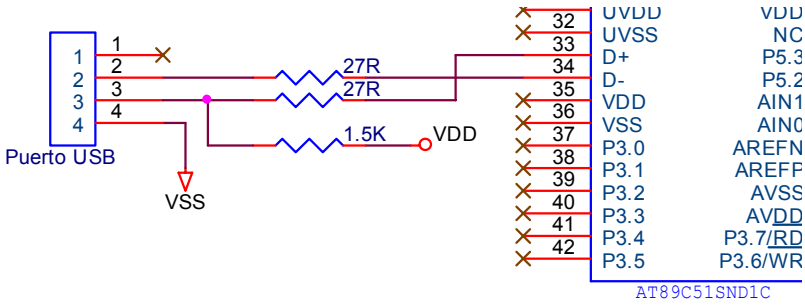


Figura 4.6. Diagrama de Conexión USB

Consideraciones de diseño para la etapa ISP.

La conexión para habilitar la programación en línea ISP recomendada por ATMEL se muestra en la Figura 4.7 y emplea un pulsador normalmente abierto.

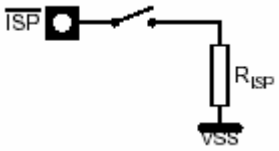


Figura 4.7. Conexión ISP Recomendada por ATMEL

La señal para realizar la programación en línea se activa cuando recibe una señal igual a cero “0” en el pin ISP (13), por lo que la conexión mostrada en la Figura 4.7 mantiene siempre deshabilitado este pin hasta el momento en el que se desee programar en línea. También es necesario encender el sistema y al mismo tiempo presionar el pulsador ISP por un tiempo aproximado de dos segundos; así se ejecuta la rutina ISP que funciona como un Bootloader y es grabado en la fábrica dentro del chip.

La resistencia con la cual trabaja ISP sirve para realizar un Pull Down y para su diseño se recurrió a la Tabla 4.2 obtenida de las hojas técnicas de ATMEL para el microcontrolador.

Símbolo	Parámetro	Mínimo	Típico	Máximo	Unidad
RISP	Resistencia ISP de Pull Down		2.2		KΩ

Tabla 4.2. Características ISP

4.2.1.3. REGULADOR DE VOLTAJE

El regulador de voltaje es una etapa indispensable para asegurar que todos los dispositivos sean polarizados de una manera correcta y evitar daños permanentes en ellos. Todos los dispositivos involucrados en este proyecto, a excepción del LCD, trabajan con un voltaje de 3.3 (V), tanto en lo referente al voltaje de alimentación como al voltaje en los buses de datos y líneas de transmisión.

El regulador de voltaje se lo realizó con el circuito integrado LM2937, que puede trabajar con un voltaje positivo de hasta 5 (V) y suministrar una corriente mayor a los 500mA. El encapsulado del LM2937 es plástico, posee tres pines y una placa para conectar un disipador de calor. Para el caso de este proyecto no es necesario diseñar los cálculos para implementar un disipador de calor porque la potencia total disipada por los componentes es muy baja.

Consideraciones para el diseño.

La conexión recomendada por el fabricante se muestra en la Figura 4.8.

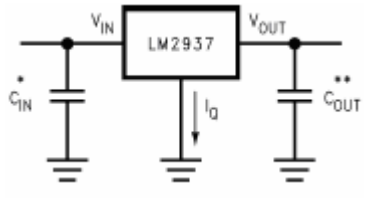


Figura 4.8. Conexión Recomendada para el Regulador de Voltaje

Las consideraciones de diseño están basadas en las hojas técnicas del fabricante, National Semiconductors, donde los dos capacitores empleados tienen el mismo valor y corresponden a las conexiones a tierra. La Tabla 4.3 muestra las consideraciones recomendadas:

Símbolo	Parámetro	Valor	Unidad
Cin	Capacitor de entrada a tierra	10	uF
Cout	Capacitor de salida a tierra	10	uF

Tabla 4.3. Características del regulador de voltaje

El regulador de voltaje soporta un voltaje máximo de 5 (V) en su alimentación. En caso de excederse este valor, el regulador podría dañarse de manera permanente. Es necesario verificar el voltaje en la entrada del regulador, y de igual manera comprobar que la salida este cumpliendo las especificaciones de voltaje de 3.3 (V) para evitar el mal funcionamiento en los componentes del sistema. Para un mejor desempeño del integrado es recomendado trabajar en condiciones de temperatura de -40 a +125 ºC.

4.2.2. MULTIMEDIA CARD

La conexión para la comunicación con la tarjeta MMC a través del módulo MMC del microcontrolador AT89C51SND1 está basada en los estándares planteados por la MMCA. Los diferentes fabricantes de tarjetas MMC publican documentos relacionados al contenido y manejo de sus tarjetas, de los cuales se ha recopilado importante información para diseñar esta etapa del sistema. Luego de analizar esta información se encontraron criterios comunes para la conexión de la tarjeta MMC.

Consideraciones para el diseño.

Para el manejo de las tarjetas MMC se puede trabajar con 7 líneas sin importar la versión de la MMC. Las tres líneas básicas con las cuales se puede realizar la comunicación con una tarjeta MMC son: CLK, CMD y DAT; pero no se debe descuidar la polarización a través de las tres líneas (VDD, VSS1 y VSS2). El voltaje de polarización es de máximo 3.3 (V).

Las líneas de comunicación CMD y DAT, requieren que se les conecte resistencias de Pull Up para evitar datos falsos que afecten al correcto desempeño del sistema. La conexión recomendada por la MMCA se muestra en la Figura 4.9.

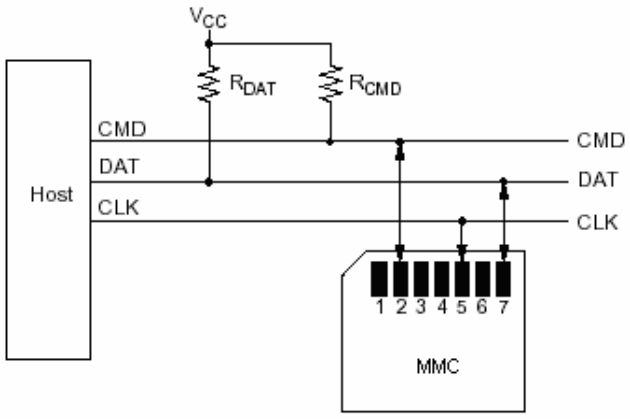


Figura 4.9. Conexión Recomendada por MMCA para las Líneas de Transmisión

Los valores recomendados por los fabricantes para estas resistencias se muestran en la Tabla 4.4.

Símbolo	Parámetro	Mínimo	Máximo	Unidad
RCMD	Resistencia en la línea CMD	4.7	100	KΩ
RDAT	Resistencia en la línea DAT	50	100	KΩ

Tabla 4.4. Parámetros de Resistencias de Pull Up en las Líneas de Transmisión

Se utilizó valores de resistencias recomendadas por ATMEL para la línea de comandos (pin 2 CMD MMC) con un valor de 10 KΩ y para la línea de datos (pin 7 DAT MMC) con el valor de 100 KΩ.

La línea de reloj CLK se conecta de manera directa entre el microcontrolador AT89C51SND1 y el pin 5 (CLK MMC) de la tarjeta MMC. Esta línea lleva los pulsos de reloj necesarios para la sincronización entre el microcontrolador y la tarjeta MMC.

La frecuencia de reloj MMC puede ser ajustada entre 0 y la frecuencia máxima del reloj del microcontrolador AT89C51SND1C (FOSC = 16 MHz), pero hay que tener en cuenta las recomendaciones presentadas por la MMCA para el cálculo máximo de la frecuencia del reloj según el número de tarjetas que disponga el sistema. Esto se muestra en la Tabla 4.5.

Parámetro	Símbolo	Mínimo	Máximo	Unidad	Observación
Frecuencia en modo de transferencia.	FMMC	0	20	MHz	hasta 10 tarjetas MMC
Frecuencia en modo de transferencia.	FMMC	0	5	MHz	hasta 30 tarjetas MMC
Frecuencia en el modo de identificación	FOD	0	400	KHz	-

Tabla 4.5. Frecuencias para los Modos MMC

En todos los circuitos integrados, en los cuales por su encapsulado se puede producir ruido, es recomendado implementar un capacitor entre las líneas de alimentación (VDD y VSS); así los fabricantes de tarjetas MMC recomiendan diseñar el circuito con un capacitor de 0.1uF que no debe estar a una distancia mayor de 13mm entre los pines 4 y 3; VDD y VSS respectivamente. La Figura 4.10 muestra esta conexión.

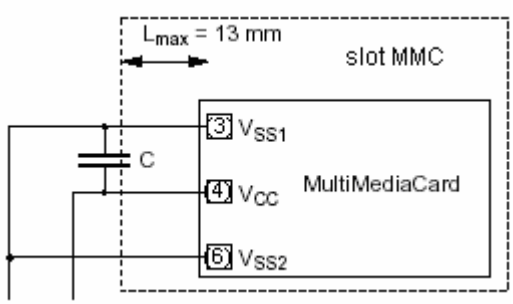


Figura 4.10. Diagrama de Conexión Recomendado por la MMCA para Disminución de Ruido

En el caso de necesitar que más tarjetas MMC estén conectadas al bus MMC, todas deberán tener su respectiva polarización. La línea CLK deberá estar conectada directamente desde el puerto del microcontrolador AT89C51SND1 hacia el pin 5 (CLK) de cada una de las tarjetas MMC. Mientras que, las líneas CMD y DAT necesitan resistencias de Pull Up (ver Figura 4.9) que podrán tener los mismos valores presentado en la Tabla 4.4.

Dependiendo del número de tarjetas conectadas habrá que tener en consideración la capacitancia y la inductancia generadas por estas, para así diseñar las correcciones y evitar fallos en el sistema. La MMCA establece que las condiciones en el bus MMC responden a los valores aproximados que se muestran en la Tabla 4.6.

Parámetro	Condición	Valor	Unidad
Capacitancia	FMMC ≤ 5 MHz (30 tarjetas MMC)	250	pF
	FMMC ≤ 20 MHz (10 tarjetas MMC)	100	pF
Inductancia	F ≤ 20 MHz	16	nH
Capacitancia por tarjeta MMC	-	7	pF

Tabla 4.6. Características del Bus MMC

Los diferentes sockets que existen en el mercado para tarjetas MMC, también permiten utilizar tarjetas SD. En el caso de tarjetas MMC hay que identificar los pines necesarios para la comunicación en modo MMC, ya que de utilizarse comunicación SPI es necesario un pin adicional para realizar la selección del integrado (CS).

En la Figura 4.11 se muestra el diseño completo para la comunicación con la tarjeta MMC teniendo como base todos los criterios anteriormente explicados.

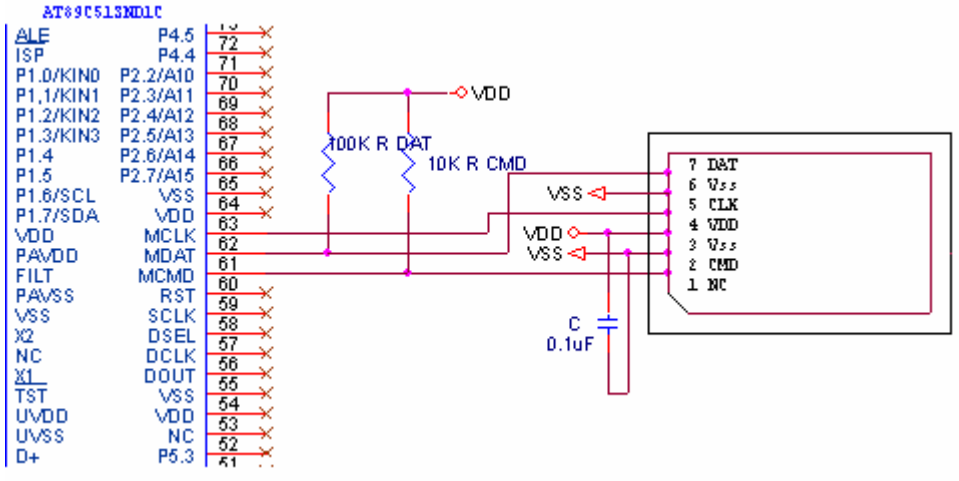


Figura 4.11. Diagrama de Conexión MultiMediaCard

4.2.3. TECLADO

El teclado es el periférico utilizado para la introducción de órdenes en el microcontrolador. Es fundamental en esta aplicación, ya que permite navegar en el contenido de la tarjeta, según las alternativas disponibles en el sistema operativo del microcontrolador.

La utilización de un teclado permite ingresar las órdenes al microcontrolador AT89C51SND1 y realizar las tareas seleccionadas. El teclado está compuesto por 4 botones que son asignados al módulo específico del microcontrolador para el manejo de teclado.

El módulo para teclado del microcontrolador AT89C51SND1 permite implementar una matriz de 4 x n y leerlo mediante interrupciones o monitoreo (polling). El valor de los pines asignados para teclado se puede obtener a través del registro KBSTA. En este sistema no se usa interrupciones para acceder al teclado.

Consideraciones para el diseño.

El microcontrolador AT89C51SND1 en su módulo para teclado implementa una interfase que permite conectar un teclado matricial de 4 x n. Está basado en 4 entradas programables y con capacidad de funcionar con interrupción. Puede detectar el flanco alto o bajo según su configuración.

Para el manejo del sistema de lectura y escritura en tarjetas MMC, se decidió implementar un teclado de 4 x 1, y así disponer de 4 botones, los cuales son suficientes para la manipulación de las órdenes al microcontrolador. Las funciones planteadas para cada botón se muestran en la Figura 4.12.

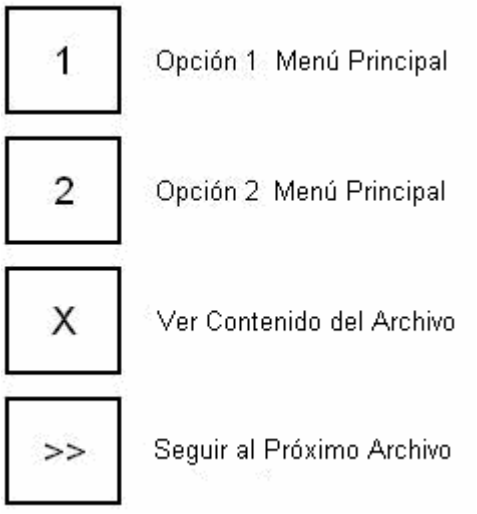


Figura 4.12. Descripción de Funciones del Teclado

Las funciones del teclado son las siguientes:

- **Opcion1 Menú Principal:** El prototipo desarrollado permite conocer el contenido de la tarjeta MMC desplegando los nombres de los archivos según las extensiones reconocidas. El nombre del archivo permanecerá en la pantalla hasta que se presione la tecla “Seguir al Próximo Archivo” o la tecla “Ver Contenido del Archivo”.
- **Opcion2 Menú Principal:** Ejecuta una función que se encarga del borrado físico de la tarjeta sin modificar el formato del sistema de archivos que se encuentra en el MBR[24].
- **Ver Contenido del Archivo:** Permite mostrar el contenido de todo el archivo seleccionado desde su cabecera hasta los “End Bits”. El contenido es presentado en formato ASCII.
- **Seguir al Próximo Archivo:** Realiza la búsqueda del siguiente archivo que se encuentre en la memoria de la tarjeta MMC y muestra el nombre del archivo.

[24] MBR: Master Boot Root.

La conexión del teclado se la implementó utilizando pulsadores normalmente abiertos que permitan la generación de un pulso de nivel ascendente para que el microcontrolador AT89C51SND1 pueda leer la orden ingresada. De igual manera, es necesario configurar la detección del nivel de flanco en alto mediante la activación de los bits KINL3:0 en el registro KBCON. Los valores de las resistencias para el teclado recomendados por fabricantes de tarjetas MMC son los mismos y tienen un valor de 220 Ω.

La Figura 4.13 muestra la conexión completa del teclado al microcontrolador AT89C51SND1.

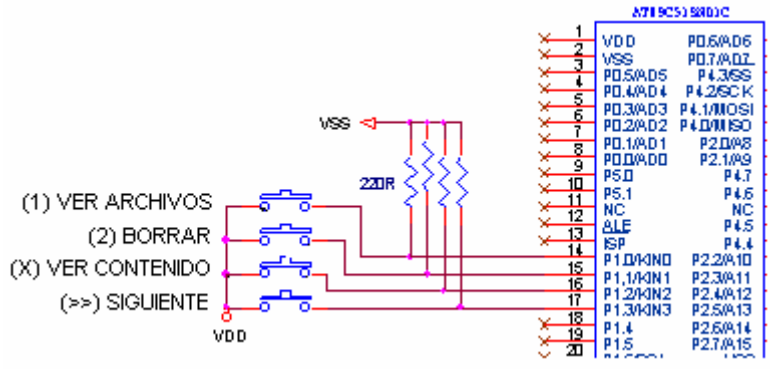


Figura 4.13. Diagrama de Conexión del Teclado

Los botones del teclado están conectados a los pines 14, 15, 16 y 17, que funcionan como puerto 1 o entradas para teclado.

4.2.4. LCD

El LCD es un dispositivo de uso común que permite visualizar información, que para esta aplicación son las opciones del Menú y el contenido de los archivos. El proyecto utiliza un LCD de dos filas por dieciséis caracteres. Esto es práctico para la aplicación ya que se destina la primera línea para la presentación del nombre del archivo encontrado, que se muestra en formato 8.3 los cuales corresponden a ocho caracteres de nombre del archivo, el carácter punto ".", y los tres últimos son designados por la extensión del archivo. La segunda línea está destinada para mostrar la aplicación principal a la cual está orientada el archivo según el análisis de su extensión.

Consideraciones para el diseño.

El LCD utiliza un controlador Hitachi HD44780, que se ha convertido en un estándar en la industria, y cuyas especificaciones son imitadas por la mayoría de los fabricantes.

El control de las principales líneas del LCD se muestra en la Tabla 4.7.

LCD		AT89C51SND1	Descripción
Dato	Pin		
DB0:7	LCD.7:14	P4.0:7	Bus de datos de 8 bits conectados al Puerto 4 del microcontrolador AT89C51SND1.
R/W	LCD.5	-	Señal que indica la lectura o escritura en los registros del LCD.
RS	LCD.4	P1.7	Señal que indica si la información presente en el bus es un comando o dato.
E	LCD.6	P1.6	Señal para selección del LCD.
LED A	LCD.15	-	Sin Conexión
LED K	LCD.16	-	Sin Conexión

Tabla 4.7. Control de las Líneas del LCD con el Microcontrolador AT89C51SND1

En la Tabla 4.8 se muestran las consideraciones recomendadas para la polarización del LCD.

LCD		Descripción
Datos	**Pin**	
VSS	LCD.1	Polarización que debe ser conectada a la tierra general del sistema.
VCC	LCD.2	Polarización a 5(V).

Tabla 4.8. Polarización del LCD

La Figura 4.14 muestra las conexiones necesarias para el manejo del LCD con el microcontrolador AT89C51SND1.

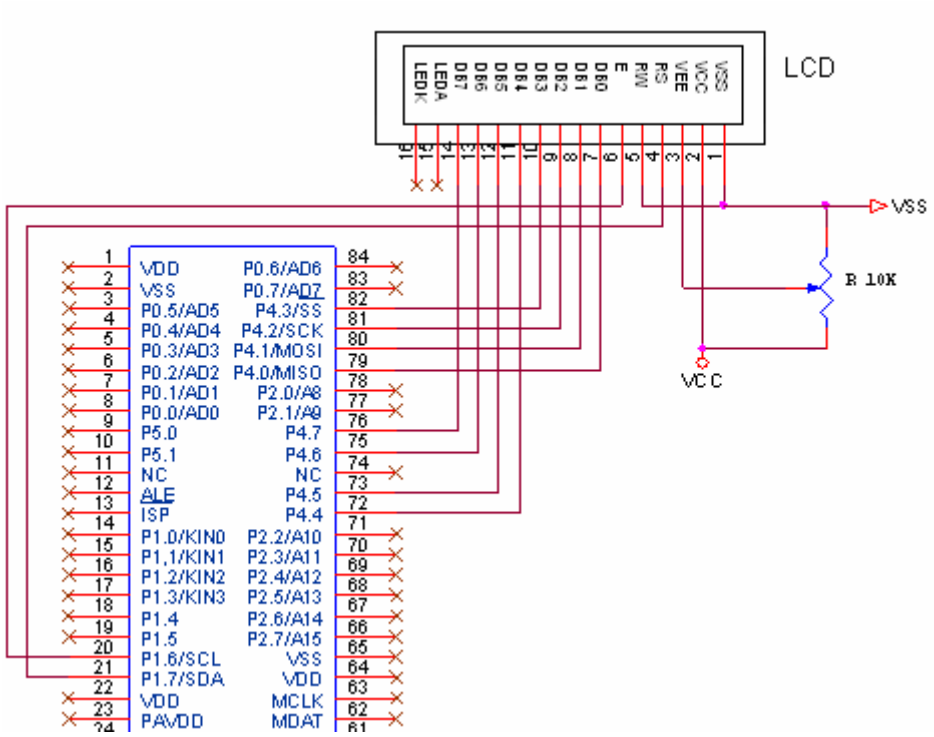


Figura 4.14. Diagrama de Conexión del LCD

CAPÍTULO V

DISEÑO DE SOFTWARE

5.1 INTRODUCCIÓN

En la actualidad existen diversos programas destinados a la creación de código para microcontroladores, en los cuales el entorno orientado a lenguaje C es el más utilizado por su facilidad en la realización de diversas aplicaciones. El presente capítulo muestra la solución propuesta para alcanzar el funcionamiento del sistema de lectura y escritura en tarjetas MMC para lo cual se mantuvo una comunicación extensa con el Servicio de Soporte de ATMEL, sin lo cual no hubiese sido posible su realización ya que no existe información de libre acceso respecto a librerías, drivers, etc.

El software utilizado para el desarrollo del código del programa y la compilación del mismo, es el µVision 2, desarrollado por la empresa Keil Software, el cual puede ser utilizado por cualquier microcontrolador que posea un núcleo con arquitectura 8051. Para mayor información acerca del manejo de µVision 2, se recomienda basarse en el proyecto de grado titulado “Desarrollo de un Sistema Reproductor de Archivos MPEG I/II Layer 3 (MP3) Basado en el Microcontrolador ATMEL AT89C51SND1C”.

El entorno que ofrece µVision 2 para la programación en lenguaje C, permite realizar el código de programa para la lectura y escritura de tarjetas MMC con el microcontrolador AT89C51SND1. La ayuda de µVision 2 permite desarrollar el código de programa con eficiencia gracias a su amplia variedad de funciones predefinidas.

El diseño de software se basó en estructurar un programa a partir de diagramas de flujo obtenidos de los documentos de libre acceso, que facilitan cumplir con todas las etapas que el proyecto demandaba. Mediante la utilización de librerías específicas solicitadas a ATMEL para el manejo de tarjetas MMC, se logró reducir el código de programa. Adicionalmente, se puede emplear programas destinados a brindar una orientación sobre la manipulación de datos en dispositivos de almacenamiento masivo, es el caso del programa FlexHEX Editor 2.0, el cual permite conocer el contenido de la tarjeta MMC.

5.2. DESARROLLO DE SOFTWARE

El desarrollo de software es la etapa en la cual se realiza el sistema operativo que permitirá manipular los datos de la tarjeta MMC. En el desarrollo de software es necesario diseñar pequeños programas por separado, los cuales cumplan con todos los requerimientos impuestos, para así alcanzar la total funcionabilidad del sistema.

El programa es extenso en cuanto a su desarrollo, debido a la múltiple cantidad de operaciones que son necesarias para el funcionamiento del sistema, por tal motivo se plantea la solución desde la parte más general, para posteriormente analizar las funciones necesarias para la aplicación deseada.

5.2.1. FUNCIONAMIENTO GENERAL DEL SISTEMA

El lenguaje C permite crear funciones las cuales sirvan para optimizar el tamaño de código empleado; así se establece el diagrama general del funcionamiento del sistema, en el cual las funciones están almacenadas en librerías según su aplicación, mientras que en la función principal “main()”, solo se accede a las funciones de inicialización de periféricos y presentación de menú. La Figura 5.1 muestra el diagrama de flujo para el funcionamiento general del sistema.

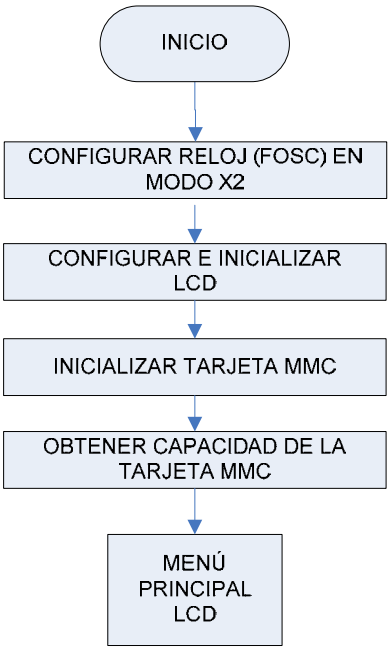


Figura. 5.1. Diagrama de Flujo General

En el diagrama mostrado en la Figura 5.1 se puede observar que existen funciones secundarias que ejecutan las instrucciones para configurar el reloj (Fosc) en modo X2, el periférico de visualización (LCD) y después se procede con la identificación de la tarjeta MMC. Una vez realizada la identificación de la tarjeta MMC se obtiene el tamaño de la capacidad de memoria de la tarjeta MMC que se

esté utilizando. El código de la función “main()” se muestra a continuación, con una descripción de las funciones secundarias:

```
void main()                         //Función principal
{
        EnableX2();                 //Modo X2 ->Fcpu = Fper = Fosc
        lcd_init();                 //Inicialización LCD
        inicializa();               //Inicialización MMC
        capacidad();                //Obtiene la capacidad de la tarjeta

        while (1)
        {
                …                   //Impresión del Menú principal
        }
}
```

Función “EnableX2()”.

Algunos microcontroladores cuya arquitectura interna posee el núcleo C51 requieren trabajar con 12 períodos de reloj por ciclo de máquina, el microcontrolador AT89C51SND1C puede trabajar con 6 períodos de reloj por ciclo de máquina, por tal motivo se debe habilitar el bit X2 en el registro CKCON. El código diseñado se muestra a continuación, así como también la Tabla 5.1 que presenta la descripción del bit empleado.

```
void EnableX2 (void)            //Función para habilitar modo X2.
{
   CKCON=CKCON | 0x01;          //Configuración de registro CKCON para habilitar modo X2.
}
```

Registro CKCON								
Bit	**Nemónico**	**Descripción**						
-	-	-	-	-	-	-	**X2**	
0	X2	Deshabilitar (0) para seleccionar 12 períodos de reloj por ciclo de máquina. (Modo estándar: Fcpu = Fosc / 2). Habilitar (1) para seleccionar 6 períodos de reloj por ciclo de máquina (Modo X2: Fcpu = Fosc).						

Tabla 5.1. Descripción del Bit X2

Función “lcd_init()”.

Realiza las instrucciones necesarias para configurar el LCD como dispositivo de visualización. Esto se lo realiza con el envío de instrucciones al microcontrolador HD44780, las cuales se refieren al formato de visualización en el cual se presentaran los datos en el LCD. A continuación se muestra el código de programa obtenido del proyecto de grado mencionado anteriormente y la Tabla 5.2 que describe las operaciones que se realizan.

```
void lcd_init (void)                    //Función para inicializar LCD
{
        LCD_RS = 0;
        lcd_write (0x38);
        lcd_write (0x04);
        lcd_write (0x0F);
        lcd_write (0x01);
        lcd_write (0x02);
}
```

Instrucción	Posibilidades	Aplicación
Tipo de bus de datos para la comunicación.	Bus de 8 bits.	Bus de 8 bits
	Multiplexado de 4bits.	
Modo de operación para visualización (# de líneas).	1 línea	2 líneas
	2 líneas	
Tamaño de caracteres	5 x 7	5 x 7
	5 x 10	

Tabla 5.2. Instrucciones para Configuración del LCD

Función “inicializa()”.

Será descrita en la sección que explique el modo de identificación para tarjetas MMC. Pero es necesario indicar que su ejecución debe ser antes de la función “capacidad ()”.

Función “capacidad()”.

Evalúa el valor de la variable “mmc_mem_size” la cual retorna el tamaño de la memoria de la tarjeta MMC. El dato obtenido en esta variable es comparado con los valores definidos en la librería “mmc.h”, todos ellos para tarjetas MMC basados en un tamaño de bloque de 512 Bytes.

```
/* Tamaño de la Tarjeta  MMC basada en 512 Bytes de tamaño de bloque */
#define MMC_SIZE_32MB      ((Uint32)62720 - 1)     //Tamaño para una tarjera de 32MB
#define MMC_SIZE_64MB      ((Uint32)125440 - 1)    //Tamaño para una tarjera de 64MB
#define MMC_SIZE_128MB     ((Uint32)250880 - 1)    //Tamaño para una tarjera de 128MB
#define MMC_SIZE_256MB     ((Uint32)501760 - 1)    //Tamaño para una tarjera de 256MB
```

En el código siguiente se observa la comparación del valor almacenado en la variable “mmc_mem_size” la cual se calcula en la función “bit mmc_init_cds()”.

```
void capacidad()                          //Función para leer la capacidad de la tarjeta
{
        lcd_clear();                      //Borra LCD
        lcd_goto(0x00);                   //Cursor en posición 0
        lcd_puts("   MultimediaCard");    //Imprime mensaje
        lcd_goto(0x40);
        lcd_puts(" MMC ");

        //imprime el tamaño de la tarjeta según sea el caso
        if(mmc_mem_size == MMC_SIZE_32MB)      lcd_puts("32 MB");
        if(mmc_mem_size == MMC_SIZE_64MB)      lcd_puts("64 MB");
        if(mmc_mem_size == MMC_SIZE_128MB)     lcd_puts("128 MB");
        if(mmc_mem_size == MMC_SIZE_256MB)     lcd_puts("256 MB");
}
```

La función “bit mmc_init_csd()” desarrollada por ATMEL está basada en el envío del comando CMD9[25], que espera una respuesta de 136 bits de longitud, que corresponde a todo el contenido del registro CSD. Una vez recibido el registro se enmascaran los bits para el valor de C_SIZE y se procede a realizar los

[25] CMD9: Comando SEND_CSD usado para solicitar el registro CSD a la tarjeta direccionada por su RCA. Recibe argumento RCA, tipo de respuesta esperada R2. Ver Anexo 1.

cálculos necesarios para obtener el tamaño de la memoria en la tarjeta MMC en MBytes (ver Tabla 2.13. Cálculos para la Capacidad de la Memoria Basados en C_SIZE). A continuación se describe los bits utilizados en estos cálculos:

- **C_SIZE [CSD73:62]:** Posee un tamaño de 12 bits que son solo de lectura, en los cuales se encuentran almacenado el tamaño del dispositivo.
- **READ_BL_LEN:** Tiene un valor de 9 que genera una longitud de tamaño de bloque igual a: $2^9 = 512$ Bytes. Su valor máximo es 11 que corresponde a una longitud de bloque de 2048 Bytes.
- **MULT:** Es un factor codificado del parámetro C_SIZE_MULT [CSD49:47] que posee un tamaño de 3 bits (ver Tabla 2.14. Descripción de C_SIZE_MULT).

El CMD9 es un comando básico (clase 0) que solicita el CSD propio de la tarjeta MMC por medio de la línea de comandos, se lo envía previo al llamado de la función “bit mmc_init_csd()”, el código necesario es el siguiente:

```
mmc_send_cmd (MMC_SEND_CSD, MMC_RCA_1, MMC_RESP_R2);
```

Se observa que para el envío del comando CMD9 se necesita de un índice el cual representa el valor del comando enviado, un argumento que representa la dirección asignada por el host a la tarjeta en la inicialización y el formato de respuesta esperado. La dirección relativa está definida en la librería “mmc.h” y contienen el siguiente valor:

```
#define MMC_RCA_1          ((Uint32)0x0001FFFF)  //Dirección asignada por el host a la tarjeta
```

Una vez enviado el comando y verificado el formato de la respuesta esperada, se procede a llamar a la función “bit mmc_init_csd()”, de la cual se describe su código a continuación.

```
bit mmc_init_csd (void)
{
	Byte  tmp;
	ACC = Mmc_rd_cmd();
	ACC = Mmc_rd_cmd();		//CSD[0]
	ACC = Mmc_rd_cmd();
	ACC = Mmc_rd_cmd();
	ACC = Mmc_rd_cmd();
	ACC = Mmc_rd_cmd();

	//Chequeo de longitud de bloque
	if ((Mmc_rd_cmd() & CSD_MSK_RBL) != CSD_BLEN_512)
	{
	//Reseteo los punteros
		MMCON0 |= (MSK_CRPTR | MSK_CTPTR);
		MMCON0 &= ~(MSK_CRPTR | MSK_CTPTR);
		return KO;
	}
	//Inicio controlador
	else
	{
		Mmc_set_block_size(CSD_BLEN_512);
	}

	/* Cálculo del tamaño de la tarjeta por números de bloques (block_size = 512) */
	/* Leer C_SIZE */
	((Byte*)&mmc_mem_size)[0] = 0;
	((Byte*)&mmc_mem_size)[1] = 0;
	((Byte*)&mmc_mem_size)[2] = Mmc_rd_cmd() & CSD_MSK_CSH;
	((Byte*)&mmc_mem_size)[3] = Mmc_rd_cmd();
	mmc_mem_size = mmc_mem_size << 2;
	((Byte*)&mmc_mem_size)[3] |= ((Mmc_rd_cmd() & CSD_MSK_CSL) >> 6);

	/* Leer C_SIZE_MULT */
	tmp = Mmc_rd_cmd() << 1;
	tmp = (tmp | (Mmc_rd_cmd() >> 7)) & CSD_MSK_CSM;
	mmc_mem_size = ((mmc_mem_size + 1) << (tmp + 2));
	mmc_mem_size--;

	//Modo de transferencia single block
	MMCON0 |= MSK_DFMT;
	MMCON0 &= ~MSK_MBLOCK;

	//Empezar control de flujo
	MMCON2 |= MSK_FLOWC;

	//Resetear punteros
	MMCON0 |= (MSK_CRPTR | MSK_CTPTR);
	MMCON0 &= ~(MSK_CRPTR | MSK_CTPTR);

	return OK;
}
```

Los registros empleados en esta función se muestran en la Tabla 5.3.

Registro MMCON0							
Bit	**Nemónico**	**Descripción**					
-	-	**CRPTR**	**CTPTR**	-	**DFMT**	-	-
5	CRPTR	*Bit de reseteo del puntero de recepción de comandos.* Deshabilitar (0) para poner en circulación el puntero de lectura del comando recibido en la FIFO. Habilitar (1) para vaciar el puntero de lectura del comando recibido en la FIFO.					
4	CTPTR	*Bit de reseteo del puntero de transmisión de comandos.* Deshabilitar (0) para poner en circulación el puntero de lectura del comando transmitido en la FIFO. Habilitar (1) para vaciar el puntero de lectura del comando transmitido en la FIFO.					
2	DFMT	*Bit de formato de datos.* Deshabilitar (0) para seleccionar formato de datos en cadena. Habilitar (1) para seleccionar formato de datos orientado a bloque.					
Registro MMCON1							
Bit	**Nemónico**	**Descripción**					
BLEN3	**BLEN2**	**BLEN1**	**BLEN0**	-	-	-	-
7-4	BLEN3:0	*Bits de longitud de bloque.* Se permite cualquier valor menor a 1010b.					
Registro MMCON2							
Bit	**Nemónico**	**Descripción**					
-	-	-	-	-	-	-	**FLOWC**
0	FLOWC	*Bits de control de flujo MMC.* Deshabilitar (0) el control de flujo durante una transferencia de datos. Habilitar (1) el control de flujo durante una transferencia de datos.					

Tabla 5.3. Descripción de Bits Empleados en la Función "capacidad()"

Ejemplo del cálculo para la capacidad de una tarjeta MMC.

Partiendo de la Tabla 5.4 donde se observan los valores característicos de las memorias Hitachi disponibles en el mercado se procede a calcular la capacidad de una tarjeta MMC Hitachi de 32 MBytes modelo HB28E032MM2.

MODELO	C_SIZE	C_SIZE_MULT	READ_BLK_LEN	CAPACIDAD DE LA TARJETA
HB28E016MM2	0x7A7	2	9	16 MByte
HB28D032MM2	0x7A7	3	9	32 MByte
HB28D064MM2	0x7A7	4	9	64 MByte
HB28B128MM2	0x7A7	5	9	128 MByte

Tabla 5.4. Descripción de Valores para Tarjetas MMC HITACHI

Cálculos:

$$MULT = 2^{C\_SIZE\_MULT+2}$$
$$MULT = 2^{3+2} = 32$$

$$BLOCKNR = (C\_SIZE + 1)\ x\ MULT$$
$$BLOCKNR = (0x7A7 + 1)\ x\ 32 = 62720$$

$$BLOCK\_LEN = 2^{READ\_BL\_LEN}$$
$$BLOCK\_LEN = 2^{9} = 512$$

$$\text{Capacidad de la tarjeta MMC} = BLOCKNR\ x\ BLOCK\_LEN$$
$$\text{Capacidad de la tarjeta MMC} = 62720\ x\ 512 = 32 Mbytes$$

5.2.2. CONFIGURACIÓN DEL RELOJ (MMCLK)

La configuración del reloj MMC constituye una parte de primordial importancia para el sistema. Permite sincronizar las operaciones para una correcta identificación de la tarjeta MMC y para el modo de lectura y escritura, permite establecer una comunicación eficiente y de alta velocidad para la transmisión de datos.

Para el cálculo del reloj se plantea la Ecuación 5.1, que es tomada de los documentos técnicos del fabricante ATMEL.

$$MMCLK = \frac{FOSC}{MMCD7:0+1}$$

Ecuación 5.1. Cálculo del Reloj MMC

Según la Ecuación 5.1 el valor del reloj MMC puede variar dependiendo de los bits MMCD7:0 en el registro MMCLK, los cuales se describen en la Tabla 5.5.

Registro MMCLK							
Bit	**Nemónico**	**Descripción**					
MMCD7	**MMCD6**	**MMCD5**	**MMCD4**	**MMCD3**	**MMCD2**	**MMCD1**	**MMCD0**
7 -0	MMCD7:0	*Divisor para el reloj MMC.* Divisor de 8 bits que genera la frecuencia del reloj MMC.					

Tabla 5.5. Descripción de Bits MMCD7:0

La Tabla 5.6 presenta los posibles valores de frecuencia que se puede generar para el reloj MMC.

FOSC (KHz)	**MMCLK (KHz)**	**MMCD7:0**	
		decimal	**Binario**
16000	**400**	**39**	**100111**
	800	20	10011
	2000	7	111
	4000	3	11
	8000	**1**	**1**

Tabla 5.5. Valores de Frecuencia según MMCD7:0

El valor de MCMD7:0 = 39 permite configurar el reloj MMC para el modo de identificación de la tarjeta MMC. Por recomendación de la MMCA el reloj no puede exceder los 400 KHz. Los otros valores de MMCD7:0 que generan otras frecuencias, pueden ser adoptados en el modo de transferencia, ya que el reloj MMC puede variar entre 0 y 20MHz, pero con la limitación de que para este sistema la mayor frecuencia podrá ser de 8MHz, debido a que la frecuencia del cristal utilizado por el sistema (FOSC) es de 16MHz.

El código para la configuración del reloj se encuentra en la librería “mmc.h” y se lo generó de la siguiente manera:

```
#define FOSC                16000               // Frecuencia de Oscilación (KHz)
#define MMC_CLK_INIT        ((FOSC / 400) - 1)  // 400 KHz para inicialización
#define MMC_CLK_TRANS       ((FOSC / 8000) - 1) // 8 MHz modo de transferencia
```

La función “Mmc_set_clock(clock)” recibe el valor de la frecuencia que se desea generar a través de las variables MMC_CLK_INIT y MMC_CLK_TRANS. Así, la definición de la función se la realiza de la siguiente manera:

```
#define Mmc_set_clock(clock)    (MMCLK = clock);
```

Para la configuración del reloj MMC se cumple la condición de que al empezar el modo de identificación, se genera a una frecuencia de 400KHz. Para el modo de transferencia se genera el reloj con un valor de MMCD7:0 = 1, para obtener la mayor frecuencia posible (8MHz). El diagrama de flujo para la configuración del reloj se muestra en la Figura 5.2.

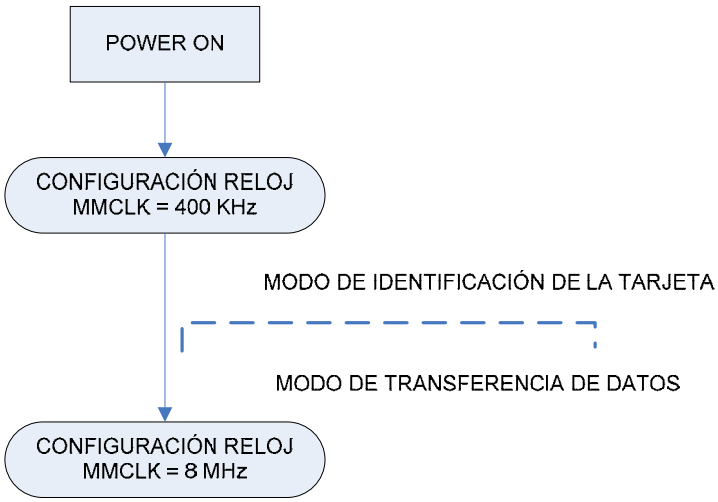


Figura. 5.2. Diagrama de Flujo para la Configuración del Reloj MMC (MMCLK)

5.2.3. IDENTIFICACIÓN DE LA TARJETA MMC

La MMCA dicta las especificaciones técnicas para las tarjetas MMC, lo que ha permitido estandarizar los requerimientos para los sistemas que trabajen con MMC. En consecuencia, la etapa de identificación requirió una amplia investigación basada en la documentación de libre acceso, en lo que se refiere a configuración de reloj, tiempos de acceso y formato de las señales de comandos básicos que son un estándar para todas las tarjetas MMC.

El sistema puede reconocer cualquier tarjeta MMC sin importar su versión, pero con la limitación de que el microcontrolador AT89C51SND1C solo soporta hasta sistema de archivos FAT16 con una capacidad menor a 256MB. Esta limitación en la capacidad de la memoria en las tarjetas se impone debido a que FAT16 solo puede manejar hasta sectores de longitud de bloque de 512 Bytes.

El software desarrollado para la identificación de la tarjeta está realizado en base a la comunicación en modo MMC por medio de un constante intercambio de

comandos y verificando los formatos de las respuesta esperadas del microcontrolador de la tarjeta MMC.

Para el proceso de identificación, la MMCA recomienda un diagrama de flujo básico, el cual fue desarrollado y en su práctica cumplió con todos los requisitos planteados para el sistema MMC. El diagrama de flujo desarrollado se muestra en la Figura 5.3.

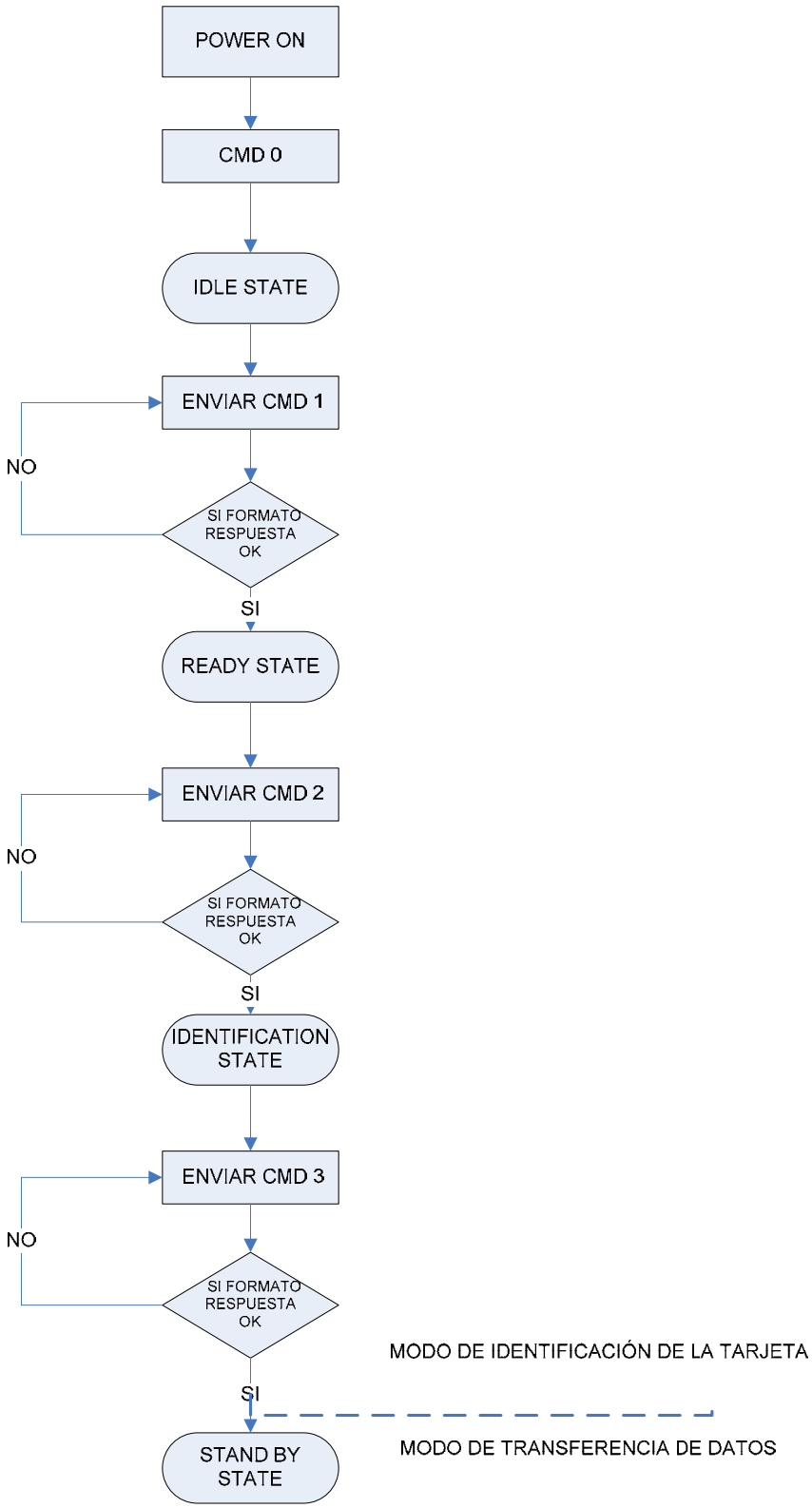


Figura .5.3. Diagrama de Flujo para la Identificación de Tarjetas MMC

En el proceso de identificación para tarjetas MMC es necesario enviar los comandos descritos en la Tabla 5.6, en la cual consta el formato que se debe asignar a cada comando para su correcto envío y posterior comprobación.

Comando	Índice	Argumento	Formato de respuesta	Descripción
GO_IDLE_STATE	CMD0	Sin argumento	-	Envía a todas las tarjetas al estado Idle
SEND_OP_COND	CMD1	[31:0] OCR sin estado busy	R3	Verifica las condiciones de operación de la tarjeta
ALL_SEND_CID	CMD2	Sin argumento	R2	Petición para que las tarjetas en estado Ready envíen su número CID
SET_RELATIVE_ADDRESS	CMD3	RCA1	R1	Asigna una dirección relativa a la tarjeta

Tabla 5.6. Formato de Comandos para el Modo de Identificación

Para el envío de cada comando, es necesario emplear la función “mmc_send_cmd (Byte index, Uint32 argument, Byte response)”. Esta función está almacenada en la librería “mmc_drv.c”, y tiene como objetivo enviar cualquier comando a la tarjeta MMC desde un microcontrolador ATMEL, para tal efecto recibe los parámetros correspondientes al formato del comando a enviar, en lo que respecta al índice, argumento y tipo de respuesta esperada.

El formato de envío de un comando tiene una longitud de 48 bits, por lo que el microcontrolador AT89C51SND1, se encarga de configurar el bit de inicio o “Start bit”, bit de transmisión y el bit de finalización o “End bit”; pero es necesario añadir

la configuración del comando para su respuesta. Así, el código desarrollado por ATMEL para esta función es el siguiente:

```
void mmc_send_cmd (Byte index, Uint32 argument, Byte response)
{
        MMCMD = index;
        MMCMD = ((Byte*)&argument)[0];
        MMCMD = ((Byte*)&argument)[1];
        MMCMD = ((Byte*)&argument)[2];
        MMCMD = ((Byte*)&argument)[3];

        switch (response)
        {
        case MMC_RESP_R1:
        case MMC_RESP_R4:
        case MMC_RESP_R5:
        {
                MMCON0 |= MSK_RFMT;                     /* Respuesta de  48 bits */
                MMCON0 &= ~MSK_CRCDIS;                  /* Respuesta con CRC7 */
                MMCON1 |= (MSK_CMDEN|MSK_RESPEN);
                MMCON1 &= ~(MSK_CMDEN|MSK_RESPEN);
                                                        /*Envía comando con respuesta*/
                break;
        }
        case MMC_RESP_R2:
        {
                MMCON0 &= ~(MSK_RFMT | MSK_CRCDIS);  //Respuesta de 136 bits con CRC7
                MMCON1 |= (MSK_CMDEN|MSK_RESPEN);
                MMCON1 &= ~(MSK_CMDEN|MSK_RESPEN); //Envía comando con respuesta
                break;
        }
        case MMC_RESP_R3:
        {
                MMCON0 |= (MSK_RFMT | MSK_CRCDIS);     //Respuesta con 48 bits sin CRC7
                MMCON1 |= (MSK_CMDEN|MSK_RESPEN);
                MMCON1 &= ~(MSK_CMDEN|MSK_RESPEN); //Envía comando con respuesta
                break;
        }
        case MMC_NO_RESP:
        default:
        {
                MMCON1 |= MSK_CMDEN;
                MMCON1 &= ~MSK_CMDEN;                   /* Envía comando sin respuesta */
                break;
        }
        }
}
```

Para manipular el formato de la respuesta, es necesario configurar los bits, que se muestran en la Tabla 5.7, según el formato de respuesta que se espera por el microcontrolador AT89C51SND1.

Registro MMCON0							
Bit	**Nemónico**	**Descripción**					
-	-	-	-	-	-	**RFMT**	**CRCDIS**
1	RFMT	*Bit de formato de la respuesta.* Deshabilitar (0) seleccionar formato de respuesta de 136 bits. Habilitar (1) seleccionar formato de respuesta de 48 bits.					
0	CRCDIS	*Bit para desactivar CRC7.* Deshabilitar (0) para deshabilitar el cálculo de CRC7 cuando se reciba una respuesta. Habilitar (1) para habilitar el cálculo de CRC7 cuando se reciba una respuesta.					
Registro MMCON1							
Bit	**Nemónico**	**Descripción**					
-	-	-	-	-	-	**RESPEN**	**CMDEN**
1	RESPEN	*Bit de activación para recibir la respuesta.* Habilitar (1) y deshabilitar (0) para activar la recepción de la respuesta después del envío del comando.					
0	CMDEN	*Bit de activación para la transmisión de comandos.* Habilitar (1) y deshabilitar (0) para activar la transmisión de los comandos hacia la tarjeta MMC.					

Tabla 5.7. Descripción de Bits Empleados para la Configuración del Envío de Comandos

Después del envío de un comando, es necesario verificar si ha sido enviado en su totalidad y sin errores, con este fin se desarrolló la función “bit check_cflck()” que realiza la verificación, mediante el análisis de la bandera CFLCK en el registro MMSTA. Así, se muestra la descripción de esta bandera en la Tabla 5.8.

Registro MMSTA							
Bit	**Nemónico**	**Descripción**					
-	-	-	-	-	-	-	**CFLCK**
0	CFLCK	*Bit que indica estado de la FIFO.* Deshabilitado por hardware cuando se encuentra disponible la FIFO para la transmisión de los comandos (estado Idle). Habilitado por hardware cuando se está realizando la transmisión de los comandos de la FIFO (estado busy).					

Tabla 5.8. Descripción de la Bandera CFLCK

Para realizar el código de esta función se analiza el estado de la bandera CFLCK. La función devuelve un OK (1) si ya se envío el comando, caso contrario retorna un KO (0). Esto se muestra en el siguiente código.

```
bit check_cflck()
{
        if (MMSTA = MMSTA & 0X01)                    //CFLCK = 1
                return KO;
        else
                return OK;
}
```

La función “bit check_cflck()” (que se encuentra en la librería “mmc.c”) es evaluada en un bucle while, en el que se encuentra hasta que el valor retornado por la función sea igual a OK. El código recomendado para este proceso es el siguiente: while (!bit check_cflck());

Una vez que se haya enviado el comando, se espera la respuesta correspondiente por parte de la tarjeta MMC, y se analiza su formato con la función “mmc_check_response()” desarrollada por ATMEL, la cual evalúa los bits que fueron configurados en el envío de comandos y al recibir la respuesta cambian el valor original de los bits configurados. La función “mmc_check_response()” está desarrollada en la librería “mmc_drv.c”.

Este es un proceso que debe seguir a todo envío de comandos para asegurar que el envío es correcto, y prueba de ello es la recepción de una respuesta correcta. El desarrollo del código para la función “mmc_check_response()” se muestra a continuación:

```
bit mmc_check_response (void)
{
        if (Mmc_response_received())
        {                                             //Respuesta recibida
        if ((MMCON0 & MSK_CRCDIS) != 0)
        {                                            /* CRC7 no computado */
```

```
                if ((MMSTA & MSK_RESPFS) != 0)
                return (MMC_RESP_OK);
                else
                return (MMC_ERR_RESP);                          /* Error de formato */
        }
        else
        {                                                       /* CRC7 computado */
                if ((MMSTA & (MSK_RESPFS | MSK_CRC7S)) == (MSK_RESPFS | MSK_CRC7S))
                return (MMC_RESP_OK);
                else
                return (MMC_ERR_RESP);                          /* Error de formato de CRC7  */
        }
        }
        else                                                    /* No se recibió respuesta*/
        return (MMC_ERR_RESP);
}
```

La función retorna un valor de MMC_RESP_OK (1) cuando se recibió una respuesta y su formato es correcto; caso contrario, entrega un valor de MMC_ERR_RESP (0).

Para el análisis de la respuesta recibida, primero se utiliza la instrucción “Mmc_response_received()”, que corresponde a una función que está encargada de verificar si una respuesta ha sido recibida en su totalidad, esto se lo realiza mediante la comprobación de la bandera EORI que se encuentra en el registro MMINT. A continuación se muestra el código utilizado y la descripción de la bandera EORI en la Tabla 5.9.

```
#define Mmc_response_received()        ((MMINT & MSK_EORI) == MSK_EORI)
```

Registro MMINT							
Bit	**Nemónico**	**Descripción**					
-	**EORI**	-	-	-	-	-	-
6	EORI	*Bandera de interrupción de fin de respuesta.* Deshabilitado por hardware cuando se lee MMINT. Habilitado por hardware cuando se recibió el fin de una respuesta.					

Tabla 5.9. Descripción de la Bandera EORI

A continuación se procede a la comprobación de los bits configurados por hardware cuando se recibe una respuesta. La Tabla 5.10 muestra la descripción de los bits empleados en la función “Mmc_response_received()”.

Registro MMCON0							
Bit	**Nemónico**	**Descripción**					
-	-	-	-	-	-	**RFMT**	**CRCDIS**
0	CRCDIS	*Bit para desactivar CRC7.* Deshabilitar (0) para deshabilitar el cálculo de CRC7 cuando se reciba una respuesta. Habilitar (1) para habilitar el cálculo de CRC7 cuando se reciba una respuesta.					
Registro MMSTA							
Bit	**Nemónico**	**Descripción**					
-	-	-	-	-	-	**CRC7S**	**RESPFS**
1	CRC7S	*Bit de estado CRC7.* Deshabilitado por hardware cuando el cálculo de CRC7 no es correcto. Habilitado por hardware cuando el CRC7 de la respuesta es correcto. Este bit no es relevante cuando CRCDIS esta habilitado.					
0	RESPFS	*Bit de estado del formato de respuesta.* Deshabilitado por hardware cuando de la respuesta recibida es correcto. Habilitado por hardware cuando el formato de la respuesta es incorrecto.					

Tabla 5.10. Descripción de Bits Empleados en la Función “Mmc_response_received()”

Existe un punto importante en esta parte del proceso, se debe esperar un tiempo al microcontrolador de la tarjeta MMC para que envíe la respuesta esperada en un tiempo máximo de 64 ciclos de reloj MMC por recomendaciones de la MMCA; en caso de no recibir respuesta se debe declarar un tiempo muerto en el sistema para salir de tal situación.

Después de realizar la identificación de la tarjeta MMC, esta se encuentra en estado Ready y queda lista para su cambio a modo de transferencia según los requerimientos del programa. En esta parte se genera el valor del reloj MMC para

modo de transferencia de datos con el valor definido en MMC_CLK_TRANS, a través del siguiente código:

```
Mmc_set_clock(MMC_CLK_TRANS);          //MMC clock en estado de transferencia
```

Ejemplo para comprobar la trama de información en la línea de comandos.

En todo proceso es necesario realizar pruebas que verifiquen que se están cumpliendo los objetivos planteados, de tal manera de avanzar sin acarrear problemas que repercutan en un mal funcionamiento del sistema. A continuación se muestra un ejemplo de código implementado que permite visualizar la trama de información comunicada, al enviarse un comando por su respectiva línea de transmisión. La toma de esta gráfica se la realizó con la ayuda de un osciloscopio para poder capturar la imagen de la señal enviada.

Para poder visualizar la trama del comando enviado, es necesario bajar la frecuencia del reloj MMC hasta 2MHz, y ejecutar el envío de un comando dentro de un ciclo while que permita obtener la mejor toma de la trama esperada. Así, se procedió a desarrollar el siguiente código que se realiza como un ejemplo para comprobar que exista transmisión en la línea de comandos.

```
while (respuesta != OK)
{
        mmc_send_cmd(MMC_SEND_STATUS,MMC_RCA_1,MMC_RESP_R1);
        t = MMC_RESPONSE_TIME;
        while (mmc_check_response() == MMC_ERR_RESP)
        {
                if (t == MMC_TIME_OUT)
                {
                        return KO;              //Declaración de tiempo muerto
                }
                t--;
        }
        return OK;
}
```

Donde:

- **MMC_SEND_STATUS:** Es el índice del comando a enviar, el cual tiene el valor de CMD13.
- **MMC_RCA_1**: Es la dirección relativa asignada a la tarjeta en el proceso de identificación que tiene el valor de 1.
- **MMC_RESP_R1**: Es el formato de la respuesta esperada.

En la Figura 5.4 se observa la señal capturada mediante el osciloscopio para el comando CMD13.

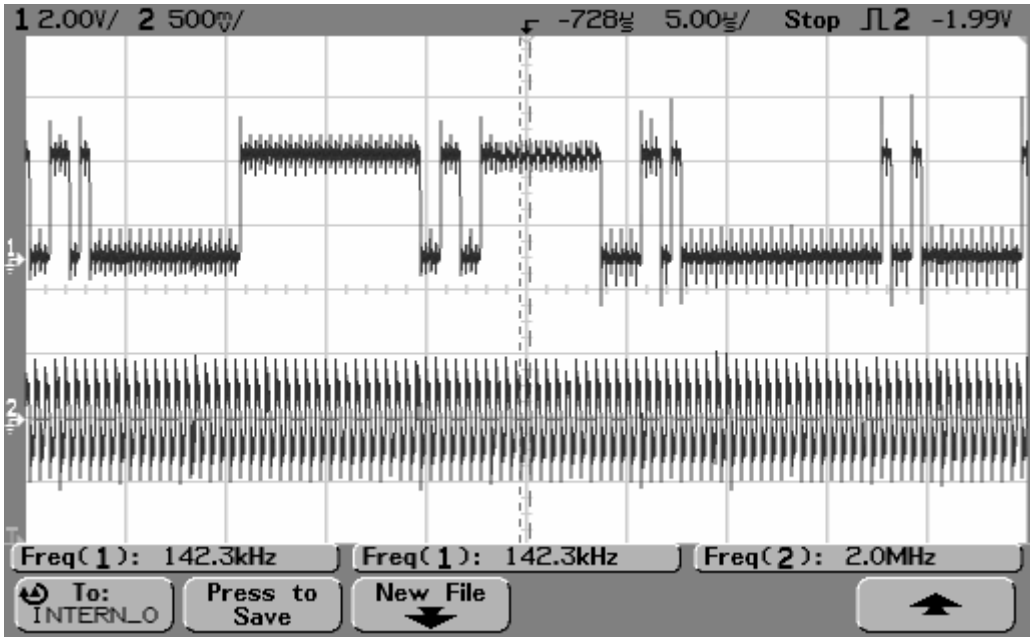


Figura 5.4. Comando SEND_STATUS Observado en el Osciloscopio

5.2.4. MODO DE TRANSFERENCIA

Después de realizar el proceso de identificación, la tarjeta MMC entra en un estado de STAND-BY, en cual se mantiene hasta que sea configurada en modo de transferencia. Para el modo de transferencia, se sigue las recomendaciones de los fabricantes de tarjetas MMC y las especificaciones de la MMCA, de las cuales se obtuvo el diagrama de flujo presentado en la Figura 5.5.

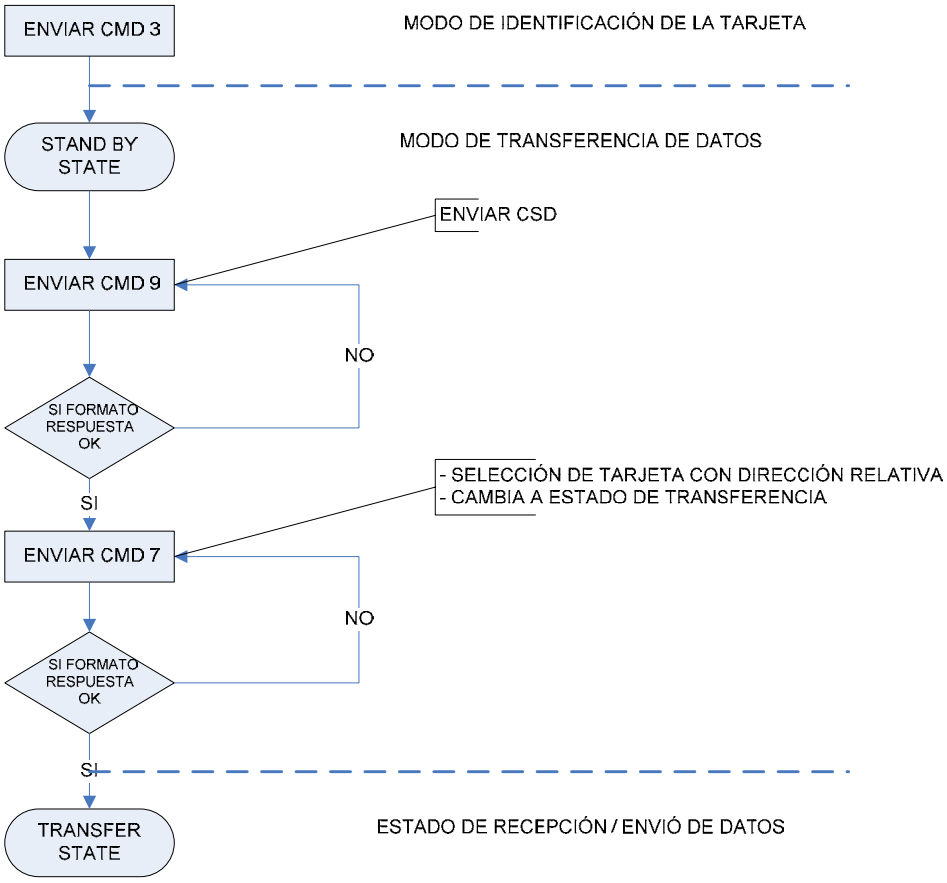


Figura 5.5. Diagrama de Flujo para Modo de Transferencia

Para configurar el modo de transferencia se debe enviar los comandos CMD7 y CMD9 que son descritos en la Tabla 5.11.

Comando	Índice	Argumento	Formato de respuesta	Descripción
SELECT / DESELECT_CARD	CMD7	[31:16] RCA [15:0] bits 0	R1	Cambia el estado de la tarjeta MMC a modo de transferencia.
SEND_CSD	CMD9	[31:16] RCA [15:0] bits 0	R2	Pregunta la dirección de la tarjeta MMC para que envíe el registro CSD por la línea de comandos

Tabla 5.11. Descripción de los Comandos Empleados en el Modo de Transferencia

El comando CMD7 (SELECT/DESELECT_CARD) selecciona la tarjeta con su dirección relativa (RCA) y automáticamente después de la comprobación de la respuesta recibida, habilita el modo de transferencia. En el caso de que existan varias tarjetas en el bus MMC, después de comunicarse con una tarjeta es necesario deseleccionarla enviando el comando CMD7 nuevamente.

Para enviar los comandos CMD7 y CMD9, se debe cumplir con todos los pasos necesarios del envío de un comando, es decir, se debe realizar la comprobación de envío del comando, la verificación de la recepción de una respuesta y la verificación del formato de la respuesta recibida.

Es importante señalar que el valor de la frecuencia del reloj MMC, ya debió haber sido configurado para 8 MHz (MMCD7:0 = 1), después del proceso de identificación de tarjetas MMC y previo al cambio al modo de transferencia.

5.2.5. ENVÍO DE DATOS

Al realizar el cambio de estado al modo de transferencia, el sistema permanece esperando que se ingrese un orden para ejecutarla. Esta orden es seleccionada del menú principal LCD. Las opciones del menú son mostradas en la Figura 4.12.

Una vez ingresada la orden, se procede a realizar las funciones que permitan el manejo de los datos almacenados en la tarjeta MMC. Para desarrollar esta parte del proceso se basó en el diagrama de flujo mostrado en la Figura 5.6.

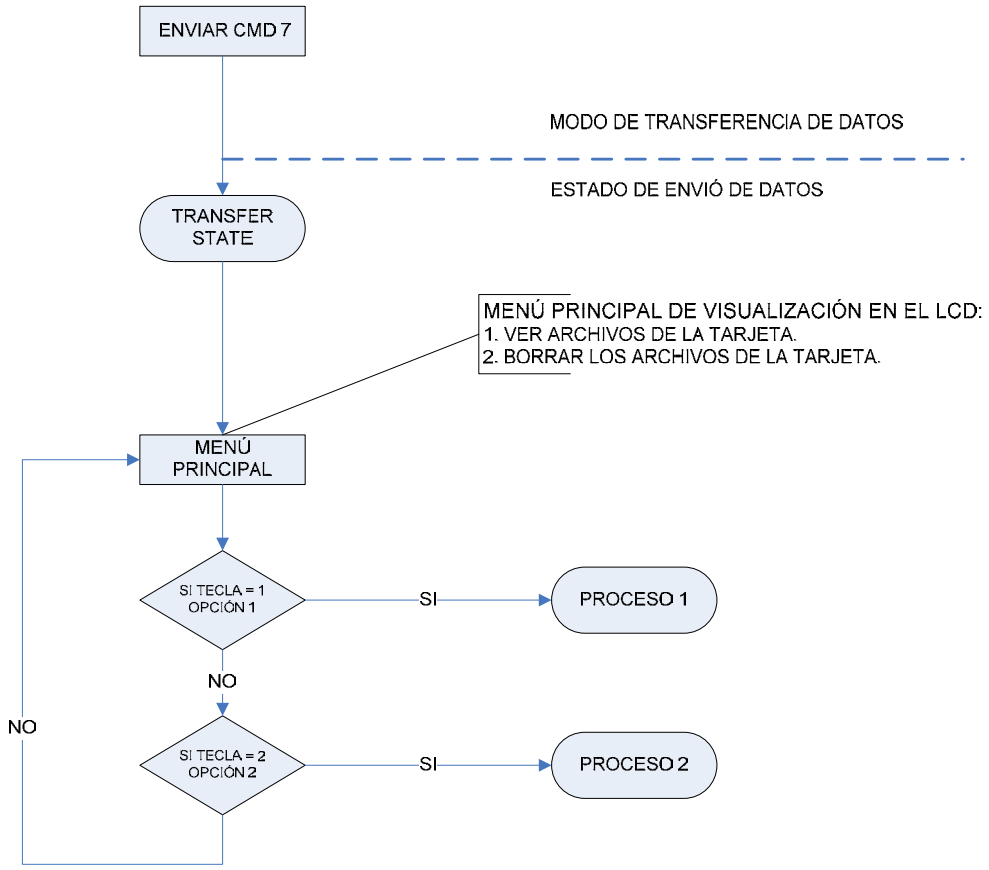


Figura 5.6. Diagrama de Flujo del Estado de Envío de Datos

Para realizar el barrido de la opción ingresada por teclado, se somete a un ciclo infinito las instrucciones que leen la tecla y es almacenada en el registro KBSTA. Esto se realiza en la función principal “main()”, donde se desarrolló el siguiente código:

```
void main()                                  //Función principal
{
        …
        …                                    //Inicializaciones
        while (1)
        {
                tecla=KBSTA;                 //Guarda tecla presionada
                KBSTA=0x00;                  //Pone en 0 las banderas de teclado
                tecla = tecla&0x0F;          //Enmascara valor de la tecla
                lcd_clear();                 //Borra LCD
                lcd_goto(0x00);
                lcd_puts("   1. Ver archivos");  //Imprime mensaje Menú principal
                lcd_goto(0x40);
```

```
                lcd_puts("2. Borrar MMC");      //Imprime mensaje Menú principal
                if(tecla==4)                    //Si se presiona la tecla de ver de archivos
                {
                        while(tecla==0);
                        leer_cab();             //Función para obtener los archivos de la Tarjeta
                }
                if(tecla==2)                    //Si se presiona la tecla de borrar MMC
                {
                        while(tecla==0);
                        lcd_clear();            //Borra LCD
                        borra();                //Función para borrar la Tarjeta
                        lcd_goto(0x00);
                        lcd_puts("   Borrado"); //Imprime mensaje
                        lcd_goto(0x40);         //Salto de línea
                        lcd_puts("    OK");     //Imprime mensaje
                        DelayMs(250);           //Retardo
                }
        }
}
```

En caso de seleccionar la opción 1, se llama a la función “leer_cab()”, que se explica en el proceso de lectura y presentación de archivos; mientras que si la opción seleccionada es la número 2, la función que se ejecuta es “borra()” la cual es presentada en el proceso de borrado de archivos.

Al ingresar una orden, es necesario esperar que el valor del registro KBSTA sea igual a 0 para evitar que se repita la función seleccionada. Esto se lo realiza con la instrucción “while(tecla==0);”. Después de ejecutar la función correspondiente a cualquiera de las dos opciones posibles, se retorna al menú principal LCD para que se pueda realizar una nueva orden.

5.2.6. LECTURA Y PRESENTACIÓN DE ARCHIVOS

La ejecución del proceso de lectura y presentación de archivos de la tarjeta MMC, es el resultado de que la opción 1 ha sido seleccionada del menú principal del LCD. A partir de este punto se llama a la función “leer_cab()”, la cual permite identificar y presentar los archivos que se encuentren en la tarjeta MMC, de acuerdo al diagrama de flujo presentado en la Figura 5.7.

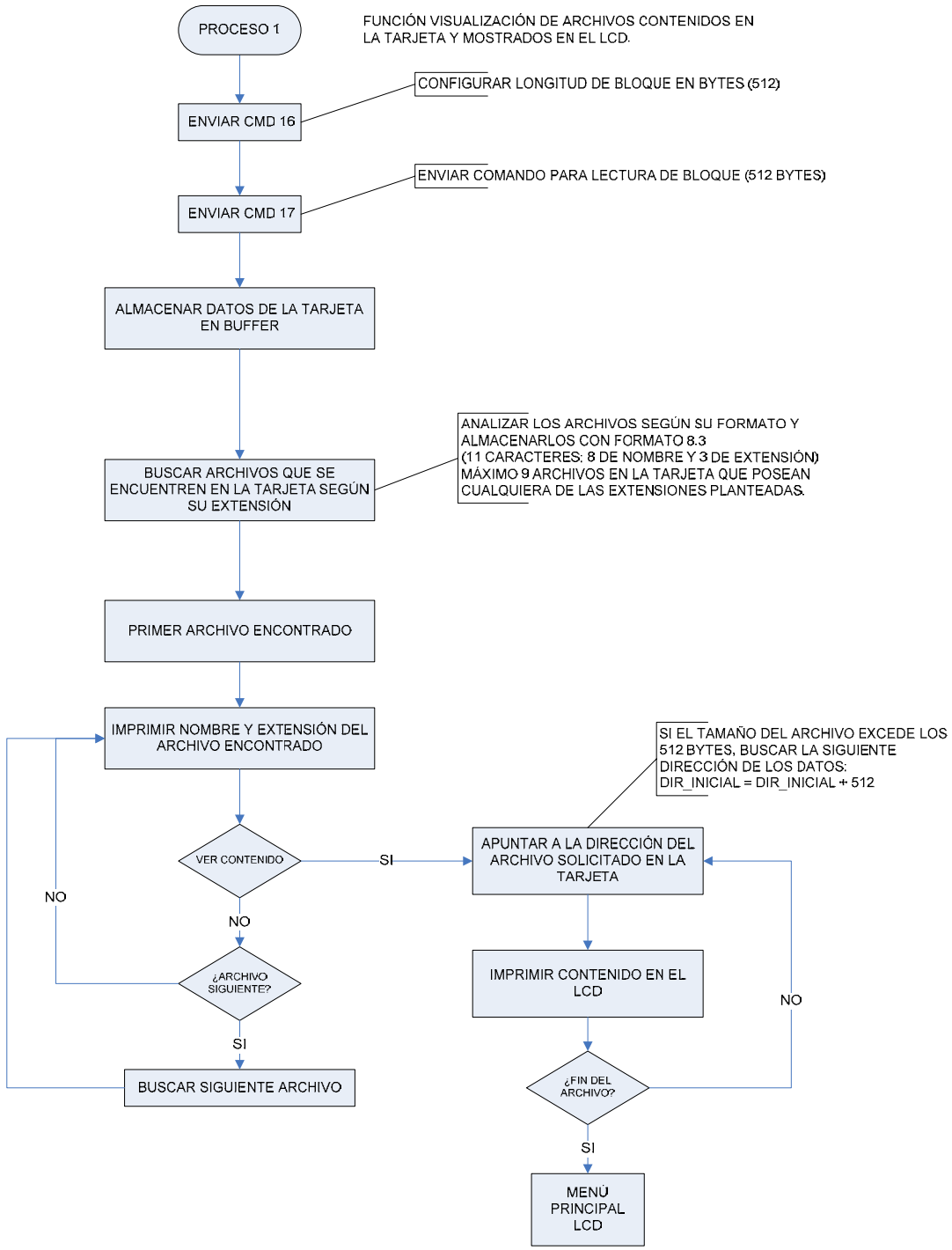


Figura 5.7. Diagrama de Flujo del Proceso de Lectura y Presentación de Archivos

Partiendo de este diagrama de flujo, se debe enviar el comando CMD16, que configura la longitud de bloque para 512 Bytes, lo cual se lo realiza con el siguiente código, y tiene su respaldo en la Tabla 5.12.

mmc_send_cmd(MMC_SET_BLOCK_LEN,CSD_BLEN_512,MMC_RESP_R1);

Donde:

CSD_BLEN_512: Selecciona la longitud de bloque en Bytes, para lo cual se define este parámetro con el valor de ((Byte)9) 2^9 = 512Bytes.

Comando	Índice	Argumento	Formato de respuesta	Descripción
SET_BLOCKLEN	CMD16	[31:0] longitud del bloque	R1	Seleccionado el tamaño de bloque en Bytes, cuando el formato de datos esta orientado a bloques

Tabla. 5.12. Descripción del Comando CMD16

Después se procede a enviar el comando para la lectura en bloque (CMD17), para realizar esta instrucción se muestra el código utilizado y la Tabla 5.13 que describe el formato del comando CMD17.

mmc_send_cmd(MMC_READ_SINGLE_BLOCK,0x0003E000,MMC_RESP_R1);

Donde:

0x0003E000: Es la dirección en la cual las cabeceras o nombres de los archivos se encuentran almacenados en la tarjeta MMC.

Comando	Índice	Argumento	Formato de respuesta	Descripción
READ_SINGLE_BLOCK	CMD17	[31:0] dirección de los datos	R1	Lee un bloque del tamaño configurado con CMD16 a partir de la dirección enviada como argumento.

Tabla 5.13. Descripción del Comando CMD17

La dirección inicial se obtuvo con el software FlexHEX Editor 2.0, el cual es de libre acceso por un tiempo de 30 días. La Figura 5.8 muestra los datos de las cabeceras almacenadas en la tarjeta MMC.

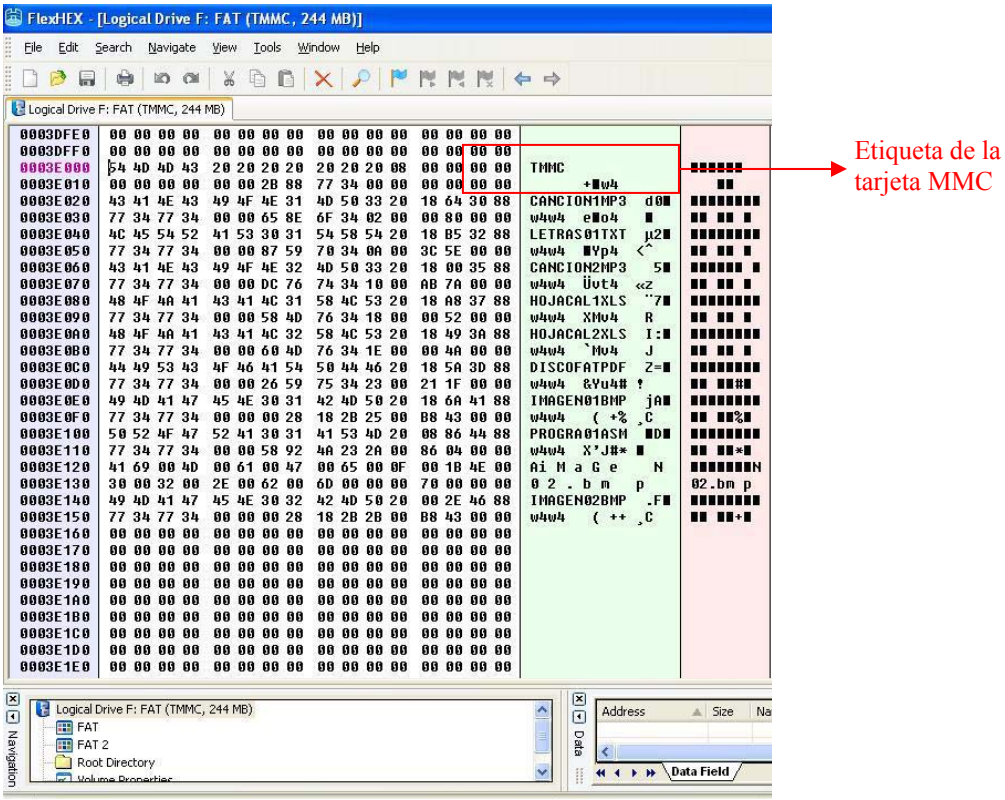


Figura 5.8. Nombres de las Cabeceras de los Archivos en la Tarjeta MMC

Al ser configurado previamente para una lectura de bloque de 512 Bytes basta con una sola lectura para guardar en la memoria del microcontrolador las cabeceras. Pero con la limitante de que el sistema está implementado para leer un máximo de nueve archivos.

Una vez enviado el comando CMD17 se procede a almacenar los datos de la tarjeta MMC en el buffer del microcontrolador AT89C51SND1 a través de la variable “dato_buffer”. Esto se lo realiza con las siguientes instrucciones, que son

generales para todos los datos que se desean almacenar y son leídos desde el registro MMDAT.

```
for (i = 0; i < 512; i++)                    //Almacena datos de la memoria
{
        dato_buffer[i++] = MMDAT;
        dato_buffer[i++]  = MMDAT;
        dato_buffer[i++]  = MMDAT;
        dato_buffer[i]     = MMDAT;
}
```

Posteriormente se procede a comprobar la existencia de archivos en la tarjeta MMC analizando la extensión del archivo. Si coincide con alguna de las extensiones impuestas se presenta el nombre del archivo en formato 8.3, el cual muestra 12 caracteres de los cuales 8 corresponden al nombre, uno al carácter "." y 3 a la extensión. A continuación se expone el código utilizado para la comparación de las extensiones en un caso particular, ya que todos los casos son iguales, solo cambian los valores de los caracteres de la extensión. El caso corresponde a la extensión "asm".

```
for (i = 0; i < 512; i++)                    //Verifica si existe un archivo *.ASM
{
        if(dato_buffer[i] =='A')             //Busca el primer carácter de la extensión
        {
                i++;                         //Incremento para buscar siguiente carácter
                if(dato_buffer[i]=='S')      //Busca el segundo carácter de la extensión
                {
                        i++;                 //Incremento para buscar siguiente carácter
                        if(dato_buffer[i]=='M')  //Busca el tercer carácter de la extensión
                        {
                                i=i-9;       //Contador al inicio del nombre del archivo
                                flag=i;
                                asm=1;       //Bandera selección tipo archivo encontrado
                                file_name(i); //Función imprimir el archivo encontrado
                                i=i+10;      //Contador para apuntar a siguiente archivo
                                conta++;     //Contador número de archivos existentes
                                caso++;      //Selecciona caso
                                while(!kbhit()); //Función para verificar la tecla presionada
                        }
                }
        }
}
```

Este programa realiza la búsqueda de la primera letra de la extensión de un archivo almacenado en el buffer de datos, en caso se coincidir con la letra “A”, se incrementa el contador y se procede a comparar con la letra “S”, en caso de ser afirmativa, se vuelve a incrementar el contador y se compara el siguiente dato del buffer con la letra “M”. Si los datos del buffer cumplen con estas tres condiciones, el archivo es un programa de ASSEMBLER, y se procede a la presentación del nombre y la aplicación de este. En caso de que el archivo no cumpliera con las condiciones planteadas para la extensión, se procede a realizar una nueva búsqueda de extensión bajo el mismo criterio pero con valores diferentes para los caracteres de la extensión.

Para la presentación del nombre del archivo encontrado, la función disminuye al contador “i” en 9 para presentar el nombre del archivo. Este contador es enviado a la función “file_name(i)”, que se encarga de imprimir el nombre y analizar la extensión según el resultado de la búsqueda realizada. El código de la función “file_name(i)” se muestra a continuación para el mismo caso de la extensión “asm”.

```
void file_name(int i)                     //Función imprimir nombre y extensión del archivo
{
        lcd_clear();                      //Borra LCD
        lcd_goto(0x00);                   //Cursor en posición 0

        //Obtiene  el nombre completo del archivo encontrado
        i=flag-2;
        let=dato_buffer[i]; i++;          //Primer carácter sin sentido del archivo
        lcd_putch(let);                   //Impresión del primer carácter encontrado
        let=dato_buffer[i]; i++;          //Siguiente carácter
        lcd_putch(let);                   //Impresión del carácter encontrado
        let=dato_buffer[i]; i++;          //Siguiente carácter
        lcd_putch(let);                   //Impresión del carácter encontrado
        let=dato_buffer[i]; i++;          //Siguiente carácter
        lcd_putch(let);                   //Impresión del carácter encontrado
        let=dato_buffer[i]; i++;          //Siguiente carácter
        lcd_putch(let);                   //Impresión del carácter encontrado
        let=dato_buffer[i]; i++;          //Siguiente carácter
        lcd_putch(let);                   //Impresión del carácter encontrado
        let=dato_buffer[i]; i++;          //Siguiente carácter
        lcd_putch(let);                   //Impresión del carácter encontrado
        let=dato_buffer[i]; i++;          //Siguiente carácter
        lcd_putch(let);                   //Impresión del carácter encontrado
        let=dato_buffer[i]; i++;          //Siguiente carácter
```

```
        lcd_putch(let);                         //Impresión del carácter encontrado

        let=0x2E;                               //Carácter "."
        lcd_putch(let);                         //Impresión del carácter

        if(asm==1)                              //Si el archivo encontrado ASSEMBLER "ASM"
         {
                flag=0;                         //Bandera en cero
                asm=0;                          //Variable en cero para próxima lectura
                lcd_puts("ASM");                //Impresión de la extensión del archivo
                lcd_goto(0x40);                 //Salto de línea
                lcd_puts("Assembler");          //Impresión del tipo de archivo encontrado
                retardo();                      //Retardo para visualización
                lcd_goto(0x40);
                lcd_puts(">> SIG     x VER");   //Menú: “Ver Siguiente” o “Visualizar Contenido”
        }
}
```

Esta función realiza la impresión de 9 caracteres, siendo el primero un carácter nulo por la configuración del reloj que afecta al LCD. Los 8 siguientes caracteres corresponden al nombre del archivo. En el caso de que el nombre del archivo no tenga 8 caracteres, se imprimen espacios en blanco. A continuación, se imprime el carácter “.” para separar el nombre del archivo de la extensión. Finalmente se analiza la bandera de búsqueda en este caso en particular es “asm” para imprimir la extensión y el tipo de archivo encontrado. A esta función solo se accede en el caso de que se haya encontrado un archivo y se requiera su presentación en el LCD.

Una vez impreso el nombre del archivo, su extensión y el nombre del programa para su aplicación, se espera un tiempo configurado en la función “retardo()” que es de 2 segundos para su visualización y se procede a borrar el tipo de aplicación del archivo para imprimir las opciones de “Siguiente Archivo” y “Ver Contenido del Archivo”.

Esta función termina su ejecución y regresa a la función “leer_cab()” que permanece en la espera del ingreso de alguna de las opciones (“Siguiente Archivo” o “Ver Contenido del Archivo”) para seguir su ejecución. Esto se realiza

con la implementación de la función “kbhit()” que fue diseñada para esta aplicación y su código se muestra a continuación:

```
bit kbhit()                             //Función para verificar la tecla presionada
{
        tecla=KBSTA;                    //Guarda tecla presionada
        KBSTA=0x00;                     //Pone en 0 las banderas de teclado
        tecla = tecla&0x0F;             //Enmascara valor de la tecla

        if(tecla==1)                    //Ingresar al contenido del archivo
        {
                while(tecla==0);
                ver_contenido();
                return OK;
        }
        if(tecla==8)                    //Seguir con la presentación de archivos
        {
                while(tecla==0);
                return OK;
                flag1=0;
        }
}
```

La función “kbhit()” realiza un barrido del teclado, en espera de alguna orden, y dependiendo de la instrucción ingresada se asigna la función especifica para la tecla presionada. La función retorna un valor de OK (1) cuando se presionó una tecla y se realizó la función respectiva; pero si no es presionada ninguna tecla, el sistema permanece en espera de alguna orden y retorna el valor de KO (0).

Cuando se escoge la opción “Siguiente Archivo”, se continua la búsqueda de los datos almacenados en el buffer y en caso de encontrar alguna coincidencia con las extensiones realiza el mismo proceso explicado para la extensión “asm”. Las extensiones que el sistema soporta se presentan en la Tabla 5.14.

Extensión	Aplicación
MP3	Archivo de audio.
XLS	Hoja de cálculo.
TXT	Archivo de texto.
PDF	Acrobat PDF
PPT	Presentación.
BMP	Mapa de bits.
ASM	ASSEMBLER.

Tabla. 5.14. Extensiones Soportadas por el Sistema

La opción “Ver Contenido del Archivo” llama a la función “ver_contenido()” en la cual se accede a la dirección inicial del archivo escogido y se calcula su tamaño, en relación al número de bloques de 512 Bytes, que son necesarios para desplegar toda la información contenida en el archivo.

La impresión se realiza después de almacenar en el buffer los 512 Bytes de datos, de la manera que se expuso anteriormente; y se la presenta para cada caso según las banderas activadas. Así, se presenta el código desarrollado para la aplicación en el caso específico de la extensión “asm”.

```
void ver_contenido()                    //Función para visualizar el contenido de la tarjeta
{
        int bucles;                     //Variable tamaño de archivo para 512 Bytes
        lcd_clear();                    //Borra LCD
        lcd_goto(0x00);                 //Cursor en posición 0

        if(caso==asm)                   //Si es el caso es el archivo ASSEMBLER "ASM"
        {
                dir_inicial=434176;     //Dirección inicial en decimal del archivo ASM en la tarjeta
                dir_final=435331;       //Dirección final en decimal del archivo ASM en la tarjeta
                flag1=1;                //Bandera para impresión del archivo
                bucles=((dir_final-dir_inicial)/512);
                                        //Número de repeticiones para el archivo
                j=0;
                while (j!=bucles)       //Lectura del tamaño del archivo
                {
                mmc_send_cmd(MMC_READ_SINGLE_BLOCK,434176+512*j, MMC_RESP_R1);
                                        //Envio de comando de lectura de bloques (512 Bytes)
                DelayMs(2);     //Retardo
```

```
                for (i = 0; i < 512; i++)    //Almacena los datos
                {
                        dato_buffer[i++] = MMDAT;
                        dato_buffer[i++] = MMDAT;
                        dato_buffer[i++] = MMDAT;
                        dato_buffer[i]   = MMDAT;
                }
                for (i = 0; i < 512; i++)    //Impresión de los datos
                {
                        let=dato_buffer[i];
                        lcd_putch(let);
                        DelayMs(5);
                }
                j++;
        }
}
}
```

En la función “mmc_send_cmd(MMC_READ_SINGLE_BLOCK, 434176+512*j, MMC_RESP_R1)”, el valor de 434176 corresponde de la dirección inicial del archivo. La variable bucle se calcula con la Ecuación 5.2:

$$bucles = \frac{dirección\_final - dirección\_inicial}{512}$$

Ecuación 5.2. Cálculo del Tamaño del Archivo en Bloques de 512 Bytes

Una vez enviado el comando para leer el primer sector de 512 Bytes, se almacena los datos en la variable “dato_buffer” y a continuación se los presentan en el LCD. Estos datos pueden ser confirmados empleando el programa FLEX HEX Editor 2.0, así como se muestra en la Figura 5.9.

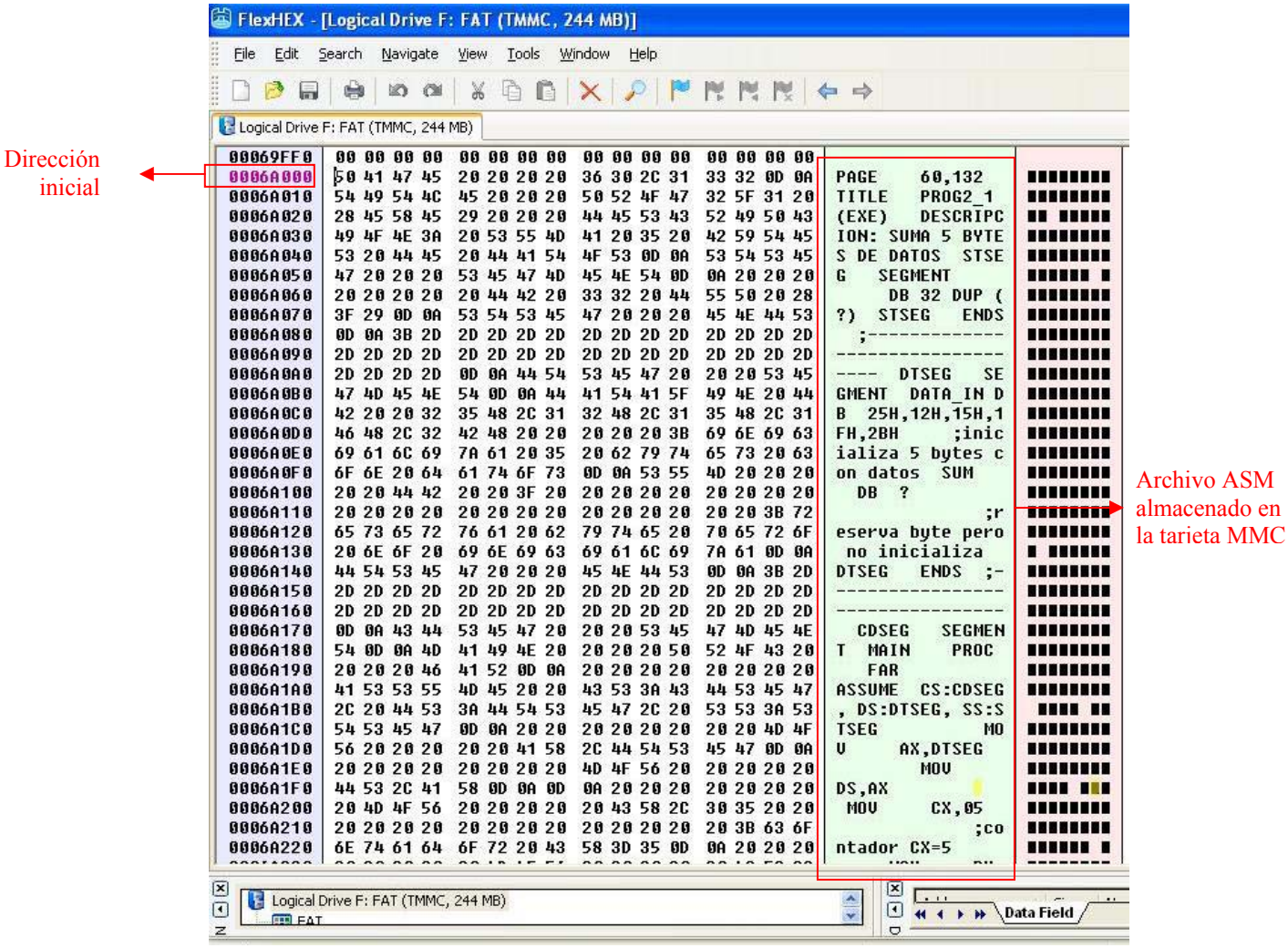


Figura. 5.9. Contenido del Archivo “asm”

Este proceso es repetido el número de veces que se asignen en la variable “bucles” hasta concluir con la presentación de todos los datos almacenados en la tarjeta MMC que corresponden al archivo seleccionado. Es importante señalar que el proceso presentado para la extensión “asm” se realiza también para todas las extensiones. Al concluir la ejecución de esta función, se regresa a la función “kbhit()” que retorna al menú principal LCD, donde se puede ingresar nuevamente cualquier opción.

Para la comprobación del envío de datos en la línea DAT, se utiliza nuevamente el osciloscopio para visualizar la trama de información. Para este caso no fue necesario realizar ningún otro programa adicional, solo se tomaron las imágenes de la línea de datos, así como se muestra en la Figura 5.10, basada en una comunicación con una longitud de bloque de 512 Bytes.

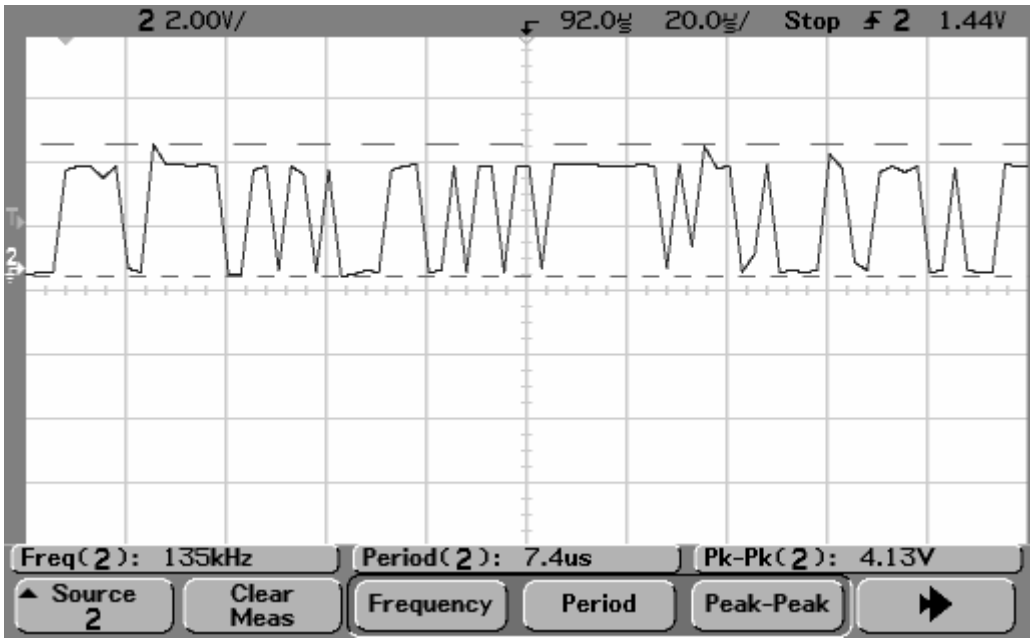


Figura 5.10. Trama de Información en la Línea de Datos

5.2.7. BORRADO DE ARCHIVOS

El proceso de borrado de archivos corresponde a la opción 2 del menú principal LCD, en la cual se ingresa al contenido de la tarjeta MMC para realizar un borrado físico de un sector seleccionado a través del envío de comandos específicos de las tarjetas MMC. Para ingresar las direcciones que se deseen borrar, se escoge una dirección de inicio y una dirección de fin; este sector puede ser manipulado a criterio del programador. Para esta aplicación se selecciona un sector desde la dirección 0x00000200h hasta la dirección 0x0F4BDFFFh que corresponde a la última posición de memoria de la tarjeta MMC.

El primer bloque de 512 Bytes (200h) no es borrado, puesto que contiene la información de identificación del sistema de archivos FAT, en lo que respecta al Master Boot Root (MBR). En la Figura 5.11 se muestra el diagrama de flujo para el desarrollo de esta función:

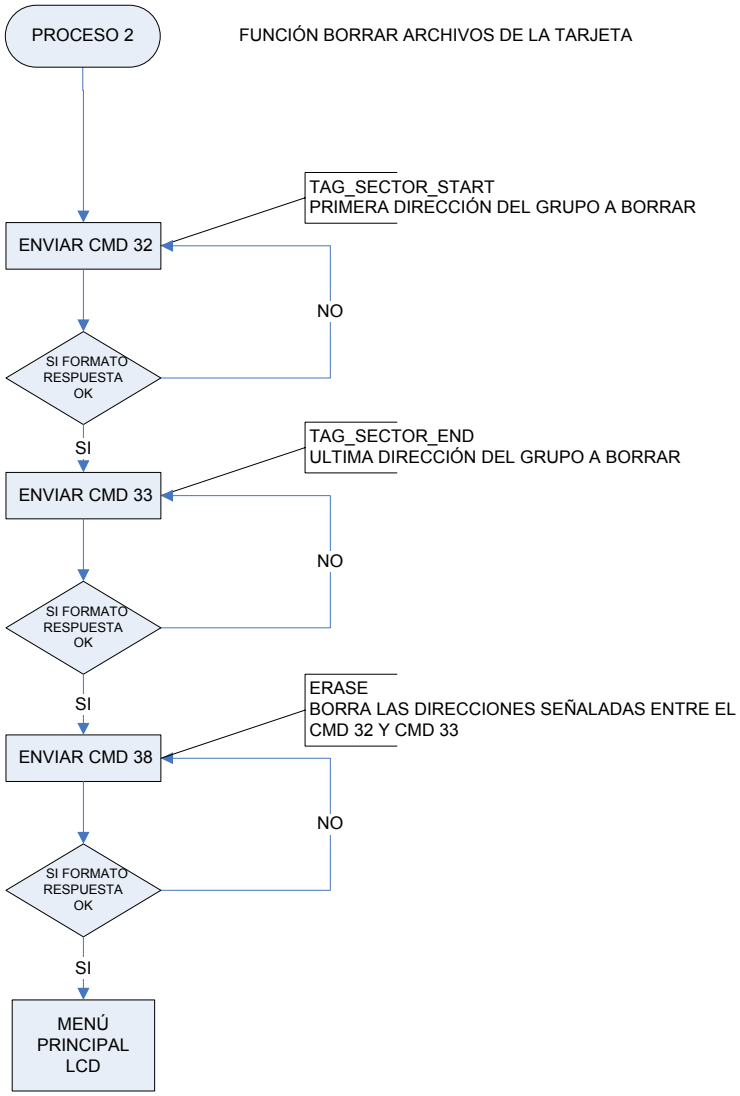


Figura 5.11. Diagrama de Flujo para el Proceso de Borrar Archivos

Los comandos implementados para el desarrollo de esta etapa del proyecto, se muestran en la Tabla 5.15.

Comando	Índice	Argumento	Formato de respuesta	Descripción
TAG_SECTOR_START	CMD32	[31:0] dirección de los datos	R1	Señala la dirección inicial de un sector para formar un grupo para borrar.
TAG_SECTOR_END	CMD33	[31:0] dirección de los datos	R1	Señala la dirección final de un sector y forma un grupo par ser borrado
ERASE	CMD38	[31:0] bits 0	R1	Borra el sector previamente seleccionado.

Tabla 5.15. Comandos Empleados para el Proceso de Borrado de Archivos

Después del envío de cada comando, se debe realizar las mismas operaciones de verificación de envío y formato de la respuesta recibida. Para después de su total ejecución retornar al menú principal del LCD.

Bajo estas condiciones se realiza el siguiente código que permite las operaciones para borrado de la tarjeta MMC, el cual está desarrollado en la función “borra()”.

```
void borra()                                    //Función para borrar contenido de la tarjeta
{
        mmc_send_cmd(MMC_TAG_SECTOR_START ,0x00000200, MMC_RESP_R1);
                                                //Dirección inicial para borrar
        DelayMs(10);                            //Retardo para esperar una respuesta
        t = MMC_RESPONSE_TIME;
        while (mmc_check_response() == MMC_ERR_RESP)
                                                //Verifica la respuesta
        {
                if (t == MMC_TIME_OUT)          //Sin respuesta                         {
                        lcd_clear();            //Borra LCD
                        lcd_goto(0x00);         //Cursor en posición 0
                        lcd_puts("   No hay respuesta");
                }
                t--;
        }

        mmc_send_cmd(MMC_TAG_SECTOR_END, 0x0F4BDFFF, MMC_RESP_R1);
                                                //Dirección final que se desea borrar
        DelayMs(10);                            //Retardo para esperar una respuesta
        t = MMC_RESPONSE_TIME;
```

```
        while (mmc_check_response() == MMC_ERR_RESP)
                                                //Verifica la respuesta
        {
                if (t == MMC_TIME_OUT)          //Sin respuesta
                {
                        lcd_clear();            //Borra LCD
                        lcd_goto(0x00);         //Cursor en posición 0
                        lcd_puts("   No hay respuesta");
                }
                t--;
        }

        mmc_send_cmd(MMC_ERASE ,MMC_NO_ARG, MMC_RESP_R1);
                                                //Borra sector
        DelayMs(20);                            //Retardo para esperar una respuesta
        t = MMC_RESPONSE_TIME;
        while (mmc_check_response() == MMC_ERR_RESP)
                                                //Verifica la respuesta
        {
                if (t == MMC_TIME_OUT)          //Sin respuesta
                {
                        lcd_clear();            //Borra LCD
                        lcd_goto(0x00);         //Cursor en posición 0
                        lcd_puts("   No hay respuesta");
                }
                t--;
        }
}
```

CAPÍTULO VI

CONCLUSIONES Y RECOMENDACIONES

6.1. CONCLUSIONES

En la actualidad el uso de las tarjetas MMC se ha convertido en un medio muy común para el almacenamiento de archivos, como por ejemplo imágenes, sonido e incluso archivos multimedia. Esto es debido a su gran capacidad, tamaño, fiabilidad y bajo costo. Por ello están siendo utilizadas por los principales manufacturadores de tecnología multimedia, que promueven un constante desarrollo para el aumento de capacidad y reducción de tamaño físico.

Las tarjetas MMC se mantendrán bajo el estándar abierto de las especificaciones técnicas publicadas por la MMCA para permitir el diseño e implementación de sistemas de bajo costo, escasa circuitería y con la posibilidad de acceder a todas las versiones de tarjetas MMC existentes en el mercado.

En la comunicación en modo MMC cada vez que se envíe un comando desde el host al microcontrolador de la tarjeta se debe esperar un tiempo aproximado de 2 ms, según recomendaciones de la MMCA, para recibir la respuesta y posteriormente verificar los bits de formato. En caso de existir un error en la respuesta, se deberá enviar el comando a la tarjeta hasta recibir los datos esperados.

Las tarjetas MMC incluyen comandos específicos para la comunicación en modo MMC. Estos comandos son enviados desde el host a la tarjeta y permiten realizar diferentes operaciones como: borrado de un sector seleccionado, protección de datos temporal o permanente con clave, cambiar la etiqueta (Label) de la tarjeta MMC, entre otros.

El sistema desarrollado con el microcontrolador AT89C51SND1C posee la capacidad de brindar un entorno de trabajo eficiente para la transmisión de datos con alta velocidad para todo tipo de tarjetas MMC, siendo la única restricción el sistema de archivos que se les haya asignado.

El módulo MMC del microcontrolador AT89C51SND1C permite el manejo de archivos en tarjetas SD con modificaciones en el hardware para la selección de la tarjeta y ampliando el desarrollo del código de programa. El estándar de comunicación para tarjetas SD es el modo SPI, por ello en la mayoría de microcontroladores poseen un módulo destinado a esta aplicación.

Para aplicaciones en las que se almacene datos en bloque con una longitud mayor al tamaño del buffer se debe evaluar la posibilidad de incrementar el tamaño de la memoria RAM interna para evitar la pérdida de datos. El microcontrolador AT89C51SND1C posee un espacio de memoria de 256 Bytes que puede ser expandido a 2 KBytes, esto se lo realiza mediante la activación de los bits XRS1:0 en el registro AUXR.

Se logro recopilar y documentar la mayor cantidad de información disponible de libre acceso, la cual en la mayoría de veces no fue la necesaria para la realización de este proyecto, debido a que las tarjetas MMC se rigen al estándar de la MMCA las cuales son de acceso restringido y tienen un valor comercial para

especificaciones de tarjetas versión 3.1 con un precio de $500 (US) y para versión 4.1 (MMCplus) de $1,000 (US).

Para un manejo confiable en la manipulación de los datos de la tarjeta MMC se empleó la herramienta de software FLEXHEX editor 2.0, por medio de la cual se puede observar el contenido de medios físicos extraíbles, y así confirmar que los datos desplegados en el LCD son correctos.

Actualmente la investigación en el área de microcontroladores brinda la capacidad de desarrollar sistemas de bajo costo en comparación a los existentes en el mercado, lo que impulsa a seguir realizando proyectos de esta clase de complejidad para generar un nivel de desarrollo y avance científico, que conduzca a desarrollar y producir, y no solo a consumir tecnología.

6.2 RECOMENDACIONES

Para mantenimiento y ampliaciones del sistema se recomienda la realización de un manual técnico con el propósito de brindar información que sirva como base para el desarrollo de futuros proyectos que aprovechen todas las características del microcontrolador AT89C51SND1C.

La manera eficaz de realizar el código para un correcto funcionamiento del sistema consiste en probar por separado cada módulo del software elaborado para finalmente probarlo de una forma integral.

El prototipo desarrollado se limita a proveer la información contenida en un número determinado de archivos. Un futuro trabajo podría extender el desarrollo para mejorar la manipulación de datos en este tipo de tarjetas manejando

sistemas de archivos FAT. En el caso del módulo MMC del microcontrolador AT89C51SND1C se puede acceder a dispositivos FAT16 por lo que se recomienda limitar el manejo de tarjetas MMC con una capacidad máxima de 256 MBytes.

Verificar el control de flujo es de vital importancia en la transmisión de datos debido a que permite conocer el estado de la transmisión, a través de la activación de banderas y detectar el tipo de error que se comete en la transmisión.

Se recomienda utilizar una comunicación basada en la transferencia de datos en bloques ya que evita el manejo de banderas para controlar el estado de las FIFO; pero de igual manera es importante verificar la máxima longitud de tamaño de bloque que soporte la tarjeta MMC.

Comprobar que el estado de la tarjeta MMC sea cambiado con el comando CMD7 para asegurar que se encuentra en el estado de transmisión de datos para acceder al contenido de la tarjeta MMC. Al finalizar las operaciones de transmisión, se debe enviar el comando CMD7 nuevamente para regresar a la tarjeta al estado Stand By.

Contar con una herramienta de trabajo como un osciloscopio, la cual brinda la facilidad de visualizar señales para verificar que las tramas de las señales del reloj, comandos y datos cumplan con las especificaciones técnicas del módulo MMC del microcontrolador utilizado.

REFERENCIAS BIBLIOGRÁFICAS

- http://www.pcmcia.org/benefits.htm, PC Card.
- http://www.pcmcia.org/pccard/pccardtechnologyprimer.htm, PC Card.
- http://www.compactflash.org/compactflash/cfassociationinformation.htm, Compact Flash.
- www.hscjpn.co.jp/in_english/down/pdf/CF.pdf, Compact Flash
- http://www.sony.com.mx/novedades/memorystick/, memory stick
- http://www.sonystyle.com/is-bin/INTERSHOP.enfinity/eCS/Store/en/-/USD/LC_BrowseCatalog-Start?CategoryName=lc_dcc_DIDigitalCameras, Memory Stick.
- http://www.pcmcia.org/smartmedia.htm, SmartMedia
- www.ssfdc.or.jp/english, SmartMedia.
- http://www.xd-picture.com/info/index.html, xD Picture
- http://www.sdcard.org/sd_memorycard/index.html, SD
- http://www.sdcard.org/sd_applications/index.html, SD
- http://www.sdcard.com/usa/TextPage.asp?Page=2, SD
- http://computer.howstuffworks.com/flash-memory.htm, Memorias flash USB.
- http://aplawrence.com/Makwana/nonvolmem.html, Memorias flash USB.
- http://211.45.27.23/Products/Semiconductor/Flash/WhatisFlash/FlashStructure.htm, MMC
- http://www.mmca.org, MMC
- http://www.atmel.com/dyn/products/product_card.asp?PN=AT89C51SND1C , Microcontrolador AT89C51SND1C.
- http://www.atmel.com/dyn/products/devices.asp?family_id=617, Microcontrolador AT89C51SND1C.
- http://www.hitachisemiconductor.com/, MMC
- http://www.joegeluso.com/software/articles/ccitt.htm, CCITT.
- http://cr.yp.to/hardware/ecc.html, ECC
- Referirse a las especificaciones técnicas que se encuentran en el CD carpeta DATASHEETS.

ANEXO 1

COMANDOS

Comandos

Los comandos están divididos en clases según el tipo de tarjeta MMC, las cuales pueden soportar las siguientes clases:

Card Command Class (CCC)	Descripción de la clase de comando
Clase 0	Básicos (Basic)
Clase 1	Lectura en cadena (stream read)
Clase 2	Lectura en bloque (block read)
Clase 3	Escritura en cadena (stream write)
Clase 4	Escritura en bloque (block write)
Clase 5	Borrar (erase)
Clase 6	Protección de escritura (write protection)
Clase 7	Protección tarjeta (lock card)

Los comandos de Clase 0 están incluidos en todas las tarjetas MMC, ya que son empleados en los procesos de identificación. La información sobre el tipo de comandos que soporta la tarjeta se provee en el registro CSD. Se definen 4 tipos de comandos para el bus MMC que son enviados por la línea CMD:

Símbolo	Nombre	Descripción
bc	Broadcast commands	No reciben respuesta.
bcr	Broadcast commands with response	Todas las tarjetas responden simultáneamente por la línea CMD.
ac	Addressed commands	Direccionados punto – punto. La respuesta se transmite por la línea CMD.
adtc	Addressed data transfer commands	Direccionados punto – punto. La respuesta se trasmite por la línea CMD y los datos por la línea DAT.

Comandos básicos (Clase 0) y comandos para lectura en cadena (Clase 1).

Índice CMD	Tipo	Argumento	Respuesta	Abreviación	Descripción del comando
CMD0	bc	[31:0] x bits	-	GO_IDLE_STATE	Envía todas la tarjetas al estado IDLE
CMD1	bcr	[31:0] OCR sin Busy	R3	SEND_OP_COND	Verifica que las tarjetas soporten el correcto rango de voltaje.
CMD2	bcr	[31: 0] x bits	R2	ALL_SEND_CID	Solicita a todas las tarjetas sus números CID.
CMD3	ac	[31:16] RCA [15:0] x bits	R1	SET_RELATIVE_ADDR	Asigna una dirección relativa en el proceso de identificación.
CMD4	bc	[31:16] DSR [15:0] x bits	-	SET_DSR	Programa el DSR de todas las tarjetas en estado Stand-by.
CMD7	ac	[31:16] RCA [15:0] x bits	R1	SELECT/ DESELECT_CARD	Actúa como un switch para cambiar a una tarjeta del estado Stand-by al estado de transmisión. Después de realizar las operaciones en la tarjeta es necesario reenviar el CMD7 para retornarla al estado Stand-by.
CMD9	ac	[31:16] RCA [15:0] x bits	R2	SEND_CSD	Solicita a la tarjeta direccionada enviar su CSD.
CMD10	ac	[31:16] RCA [15:0] x bits	R2	SEND_CID	Solicita a la tarjeta direccionada enviar su CID.
CMD11	adtc	[31:0] dirección de datos	R1	READ_DAT_ UNTIL_STOP	Lee los datos en cadena desde la tarjeta empezando por la dirección enviada, hasta recibir el comando STOP_TRANSMISSION.
CMD12	ac	[31:0] x bits	R1	STOP_TRANSMISSION	Señala la detención de la transmisión.
CMD13	ac	[31:16] RCA [15:0] x bits	R1	SEND_STATUS	Solicita a la tarjeta direccionada enviar su registro de status.
CMD15	ac	[31:16] RCA [15:0] x bits	-	GO_INACTIVE_STATE	Envía las tarjetas al estado Inactive con el fin de proteger la pila de la tarjeta contra de interrupciones en la comunicación.

Comandos de lectura para bloques (Clase 2).

Índice CMD	Tipo	Argumento	Respuesta	Abreviación	Descripción del comando
CMD16	ac	[31:0] longitud de bloque (datos)	R1	SET_BLOCKLEN	Selecciona la longitud del bloque en Bytes.
CMD17	adtc	[31:0] dirección de datos	R1	READ_SINGLE_BLOCK	Lee un bloque del tamaño seleccionado por el comando SET_BOLCKLEN.
CMD18	adtc	[31: 0] dirección de datos	R1	READ_MULTIPLE_ BLOCK	Envía bloques de datos continuamente hasta ser interrumpido por una señal de parada.

Comandos para escritura en cadena (Clase 3).

Índice CMD	Tipo	Argumento	Respuesta	Abreviación	Descripción del comando
CMD20	adtc	[31:0] dirección de datos	R1	WRITE_DAT_ UNTIL_STOP	Escribe una cadena de datos desde el Host, empezando por la dirección señalada hasta recibir el comando STOP_TRANSMISSION.

Comando de lectura / escritura para bloques (Clase 2 / 4).

Índice CMD	Tipo	Argumento	Respuesta	Abreviación	Descripción del comando
CMD23	ac	[31:16] inicia 0 [15:0] número de bloques	R1	SET_BLOCK_COUNT	Define el número de bloques que se va a transmitir inmediatamente después de recibir el comando de lectura o escritura en bloque.

Comandos de escritura para bloques (Clase 4).

Índice CMD	Tipo	Argumento	Respuesta	Abreviación	Descripción del comando
CMD24	adtc	[31:0] dirección de datos	R1	WRITE_BLOCK	Escribe un bloque.
CMD25	adtc	[31:0] dirección de datos	R1	WRITE_MULTIPLE_ BLOCK	Continuamente escribe un bloque de datos hasta recibir el comando STOP_TRANSMISSION.
CMD26	adtc	[31:0] x bits	R1	PROGRAM_CID	Programa el registro CID. Este comando puede ser ocupado una vez por tarjeta MMC. Es necesario implementar hardware para evitar nuevas operaciones de programación. Normalmente este comando es reservado para el uso del fabricante. El registro CID es programado en la fabricación de la tarjeta.
CMD27	adtc	[31:0] x bits	R1	PROGRAM_CSD	Programa los bits programables del registro CSD

Comandos para borrar (Clase 5).

Índice CMD	Tipo	Argumento	Respuesta	Abreviación	Descripción del comando
CMD32	ac	[31:0] dirección de datos	R1	TAG_SECTOR_START	Indica la dirección inicial del sector a seleccionar.
CMD33	ac	[31:0] dirección de datos	R1	TAG_SECTOR_END	Indica la dirección final del sector a seleccionar.
CMD34	ac	[31:0] dirección de datos	R1	UNTAG_SECTOR	Borra un valor previamente seleccionado para un sector
CMD35	ac	[31:0] dirección de datos	R1	TAG_ERASE_ GROUP_START	Indica la dirección del primer grupo seleccionado para borrar.
CMD36	ac	[31:0] dirección de datos	R1	TAG_ERASE_ GROUP_END	Indica la dirección del último grupo seleccionado para borrar.
CMD37	ac	[31:0] dirección de datos	R1	UNTAG_ERASE_ GROUP	Borra un valor previamente seleccionado para un grupo
CMD38	ac	[31:0] x bits	R1	ERASE	Borra todos los sectores previamente seleccionados.

Comandos para protección de escritura (Clase 6).

Índice CMD	Tipo	Argumento	Respuesta	Abreviación	Descripción del comando
CMD28	ac	[31:0] dirección de datos	R1	SET_WRITE_PROTECT	Si una tarjeta tiene protección de escritura, este comando activa el bit write_protect para el sector. Esta característica se informa en el registro CSD.
CMD29	ac	[31:0] dirección de datos	R1	CLR_WRITE_PROTECT	Si una tarjeta tiene protección de escritura, este comando desactiva el bit write_protect para el sector.
CMD30	adtc	[31:0] protección de escritura de datos	R1	SEND_WRITE_PROT	Si una tarjeta tiene protección de escritura, este comando solicita a la tarjeta enviar el estado de los bits write_protect.

Comando para proteger la tarjeta (Clase 7).

Índice CMD	Tipo	Argumento	Respuesta	Abreviación	Descripción del comando
CMD42	adtc	[31:0] x bits	R1	LOCK_UNLOCK	Es usado para activar/desactivar la contraseña o proteger/desproteger de la tarjeta. El tamaño de los bloques de datos es configurado con el comando SET_BLOCKLEN.

Comandos reservados

CMD5
CMD6
CMD8
CMD14
CMD19
CMD21
CMD22
CMD31
CMD39
CMD40
CMD41
CMD43
CMD54
CMD55
CMD56
CMD57
CMD63

ANEXO 2

CÓDIGO DE PROGRAMA

Código de programa

```
//librerias LCD
#include "lcd.h"                          //incluye librería para uso de LCD
#include "lcd.c"                          //incluye funciones para uso de LCD

//librerias delay
#include "delay.h"                        //incluye librería de retardo
#include "delay.c"                        //incluye código de retardo

//Libreria para inicialización de la Tarjeta
#include "tareas_mmc.c"                   //definiciones variables

//librerias MMC
#include "mmc.c"                          // incluye funciones para uso de MMC
#include "mmc.h"
#include "mmc_drv.c"                      //incluye librería para uso de MMC
#include "mmc_drv.h"


//DECLARACION DE FUNCIONES

//extern void leer_teclado(void);         //declara función de lectura de teclado
static void main (void);                  //declara función principal

unsigned char let;
unsigned int tecla;                       // valor de tecla presionada
bit bandera;                              // bandera
int i,j,k,conta;                          //contadores

int flag,flag1=0;                         // bandera
int mp3,txt,xls,ppt,pdf,bmp,asm;          //variables para ubicar archivos
int caso;                                 //variable
int dir_inicial,dir_final;                //variable para ubicar archivo en la MMC

//Variables globales usadas
bdata   bit     gl_memory;                // Selecciona tipo memoria
bdata   bit     gl_mem_failure;           // falla en la memoria
xdata   Byte    dato_buffer[BUF_SIZE];    // puntero buffer (512 bytes)

Byte t;                                   //variable para evitar tiempo muerto

void EnableX2(void)                       //función para habilitar modo X2
{
        CKCON=CKCON|0x01;                 //Configuración de registro CKCON para habilitar modo X2
}
```

```
void inicializa()                                   //inicializacion MMC
{
        mmc_init();
        mem_task_init();
        mem_task();                                 //inicializacion tarjeta
}

void capacidad()                                    //Función para leer la capacidad de la tarjeta
{
        lcd_clear();                                //borra LCD
        lcd_goto(0x00);                             //cursor en posición 0
        lcd_puts("   MultimediaCard");              //imprime mensaje
        lcd_goto(0x40);
        lcd_puts(" MMC ");
        if(mmc_mem_size == MMC_SIZE_32MB)           //imprime el tamaño de la tarjeta según sea el caso
        lcd_puts("32 MB");
        if(mmc_mem_size == MMC_SIZE_64MB)
        lcd_puts("64 MB");
        if(mmc_mem_size == MMC_SIZE_128MB)
        lcd_puts("128 MB");
        if(mmc_mem_size == MMC_SIZE_256MB)
        lcd_puts("256 MB");

        DelayMs(250);                               //retardo
        DelayMs(250);
        DelayMs(250);
        DelayMs(250);
        DelayMs(250);
        DelayMs(250);
}

void borra()                                        //Función para borrar contenido de la tarjeta
{
        Byte t;
        MMCON2 &= ~MSK_FLOWC;                       //deshabilita control de flujo durante la transmisión

        MMCON2 |= MSK_CCR;                          //setea y se encera el bit CCR para resetear el controlador de
        MMCON2 &= ~MSK_CCR;                         //la línea de comando

        MMCON2 |= MSK_DCR;                          //setea y se encera el bit DCR para resetear el controlador de
        MMCON2 &= ~MSK_DCR;                         //la línea de datos

        MMCON1 |= MSK_DATDIR;                       //modo escritura

        mmc_send_cmd(MMC_TAG_SECTOR_START ,0x00000000, MMC_RESP_R1);
                                                    //Selecciona la dirección inicial para borrar
        DelayMs(10);                                //retardo para esperar una respuesta
        t = MMC_RESPONSE_TIME;
        while (mmc_check_response() == MMC_ERR_RESP)        //Verifica la respuesta
```

```
{
            if (t == MMC_TIME_OUT)              //Sin respuesta da un tiempo
            {
                        lcd_clear();            //borra LCD
                        lcd_goto(0x00);         //cursor en posición 0
                        Lcd_puts("   No hay respuesta");
                        DelayMs(250);           //retardo
                        DelayMs(250);
                        DelayMs(250);
                        DelayMs(250);
            }
             t--;
 }
mmc_send_cmd(MMC_TAG_SECTOR_END,0x00000200, MMC_RESP_R1);
                                                // Selecciona la direccion final que desea borrar
DelayMs(10);                                    //retardo para esperar una respuesta
t = MMC_RESPONSE_TIME;
while (mmc_check_response() == MMC_ERR_RESP)
{
            if (t == MMC_TIME_OUT)               //Sin respuesta da un tiempo
            {
                        lcd_clear();            //borra LCD
                        lcd_goto(0x00);         //cursor en posición 0
                        lcd_puts("   No hay respuesta");
                        DelayMs(250);           //retardo
                        DelayMs(250);
                        DelayMs(250);
                        DelayMs(250);
            }
             t--;
   }

mmc_send_cmd(MMC_ERASE ,MMC_NO_ARG, MMC_RESP_R1);
                                                //Borra la direccion seleccionada anteriormente
DelayMs(20);                                    //retardo para esperar una respuesta
t = MMC_RESPONSE_TIME;
 while (mmc_check_response() == MMC_ERR_RESP)
{
            if (t == MMC_TIME_OUT)              //Sin respuesta da un tiempo
            {
                        lcd_clear();            //borra LCD
                        lcd_goto(0x00);         //cursor en posición 0
                        lcd_puts("   No hay respuesta");
                        DelayMs(250);           //retardo
                        DelayMs(250);
                        DelayMs(250);
                        DelayMs(250);
            }
            t--;
```

```
        }
}

void ver_contenido()                                    //Función para visualizar el contenido de la tarjeta
{
        int bucles;                                     //variable para obtener el numero de repeticiones
        lcd_clear();                                    //borra LCD
        lcd_goto(0x00);                                 //cursor en posición 0

        if(caso==1)                                     //Si es el caso es un archivo "MP3"
        {
                dir_inicial=270336;                     //dirección inicial del archivo MP3 en la tarjeta
                dir_final=300720;                       //dirección final del archivo MP3 en la tarjeta
                flag1=1;                                //bandera para impresión del archivo
                bucles=((dir_final-dir_inicial)/512);   //Número de repeticiones para observar todo el archivo
                j=0;                                    //variable
                while (j!=bucles)                       //Lectura del archivo dependiendo de su tamaño
                {
                        mmc_send_cmd(MMC_READ_SINGLE_BLOCK, 270336 + 512*j, MMC_RESP_R1);
                        DelayMs(2);                     //Retardo
                        for (i = 0; i < 512; i++)       //Almacena los datos
                        {
                                dato_buffer[i++] = MMDAT;
                                dato_buffer[i++] = MMDAT;
                                dato_buffer[i++] = MMDAT;
                                dato_buffer[i]   = MMDAT;
                        }
                        for (i = 0; i < 512; i++)       //Impresión de los datos
                        {
                                let=dato_buffer[i];
                                lcd_putch(let);
                                DelayMs(5);
                        }
                        j++;
                }
        }
        if(caso==3)                                     //Si es el caso es un otro archivo "MP3"
        {
                dir_inicial=327680;                     //dirección inicial del segundo archivo MP3 en la tarjeta
                dir_final=358982;                       //dirección final del archivo MP3 en la tarjeta
                flag1=1;                                //bandera para impresión del archivo
                bucles=((dir_final-dir_inicial)/512);   //Número de repeticiones para observar todo el archivo
                j=0;
                while (j!=bucles)                       //Lectura del archivo dependiendo de su tamaño
                {
                        mmc_send_cmd(MMC_READ_SINGLE_BLOCK, 327680 + 512*j, MMC_RESP_R1);
                        DelayMs(2);
                        for (i = 0; i < 512; i++)       //Almacenar los datos
                        {
```

```
                        dato_buffer[i++]=  MMDAT;
                        dato_buffer[i++] = MMDAT;
                        dato_buffer[i++] = MMDAT;
                        dato_buffer[i]   = MMDAT;
                }
                for (i = 0; i < 512; i++)          //Impresión de los datos
                {
                        let=dato_buffer[i];
                        lcd_putch(let);
                        DelayMs(5);
                }
                j++;
        }
}
if(caso==2)                                        //Si es el caso es un  archivo de texto "TXT"
{
        dir_inicial=303104;                        //dirección inicial del  archivo TXT en la tarjeta
        dir_final=327227;                          //dirección final del archivo TXT en la tarjeta
        flag1=1;                                   //bandera para impresión del archivo
        bucles=((dir_final-dir_inicial)/512);      //Número de repeticiones para observar todo el archivo
        j=0;
        while (j!=bucles)                          //Lectura del archivo dependiendo de su tamaño
        {
                mmc_send_cmd(MMC_READ_SINGLE_BLOCK, 303104 + 512*j, MMC_RESP_R1);
                DelayMs(2);
                for (i = 0; i < 512; i++)          //Almacenar los datos
                {
                        dato_buffer[i++] = MMDAT;
                        dato_buffer[i++] = MMDAT;
                        dato_buffer[i++] = MMDAT;
                        dato_buffer[i]   = MMDAT;
                }
                for (i = 0; i < 512; i++)          //Impresión de los datos
                {
                        let=dato_buffer[i];
                        lcd_putch(let);
                        DelayMs(5);
                }
                j++;
        }
}
if(caso==4)                                        //Si es el caso es el primer archivo de hoja de calculo "XLS"
{
        dir_inicial=360448;                        //dirección inicial del primer archivo XLS en la tarjeta
        dir_final=372415;                          //dirección final del primer archivo XLS en la tarjeta
        flag1=1;                                   //bandera para impresión del archivo
        bucles=((dir_final-dir_inicial)/512);      //Número de repeticiones para observar todo el archivo
        j=0;
        while (j!=bucles)                          //Lectura del archivo dependiendo de su tamaño
```

```
        {
                mmc_send_cmd(MMC_READ_SINGLE_BLOCK, 360448 + 512*j, MMC_RESP_R1);
                DelayMs(2);
                for (i = 0; i < 512; i++)          //Almacenar los datos
                {
                        dato_buffer[i++] = MMDAT;
                        dato_buffer[i++] = MMDAT;
                        dato_buffer[i++] = MMDAT;
                        dato_buffer[i++] = MMDAT;
                }
                for (i = 0; i < 512; i++)          //Impresión de los datos
                {
                        let=dato_buffer[i];
                        lcd_putch(let);
                        DelayMs(5);
                }
                j++;
        }
}
if(caso==5)                                        //Si es el caso es el Segundo archivo de hoja de calculo "XLS"
{
        dir_inicial=376320;                        //dirección inicial del segundo archivo XLS en la tarjeta
        dir_final=376583;                          //dirección final del primer archivo XLS en la tarjeta
        flag1=1;                                   //bandera para impresión del archivo
        bucles=((dir_final-dir_inicial)/512);      //Número de repeticiones para observar todo el archivo
        j=0;
        while (j!=bucles)                          //Lectura del archivo dependiendo de su tamaño
        {
                mmc_send_cmd(MMC_READ_SINGLE_BLOCK, 376320 + 512*j, MMC_RESP_R1);
                DelayMs(2);
                for (i = 0; i < 512; i++)          //Almacenar los datos
                {
                        dato_buffer[i++] = MMDAT;
                        dato_buffer[i++]=  MMDAT;
                        dato_buffer[i++] = MMDAT;
                        dato_buffer[i]   = MMDAT;
                }
                for (i = 0; i < 512; i++)          //Impresión de los datos
                {
                        let=dato_buffer[i];
                        lcd_putch(let);
                        DelayMs(5);
                }
                j++;
        }
}
if(caso==6)                                        //Si es el caso es un archivo Acrobat Reader "PDF"
{
        dir_inicial=403968;                        //dirección inicial del archivo PDF en la tarjeta
```

```
        dir_final=413471;                   //dirección final del archivo PDF en la tarjeta
        flag1=1;                            //bandera para impresión del archivo
        bucles=((dir_final-dir_inicial)/512);   //Número de repeticiones para observar todo el archivo
        j=0;
        while (j!=bucles)                   //Lectura del archivo dependiendo de su tamaño
        {
                mmc_send_cmd(MMC_READ_SINGLE_BLOCK, 403968 + 512*j, MMC_RESP_R1);
                DelayMs(2);
                for (i = 0; i < 512; i++)       //Almacenar los datos
                {
                        dato_buffer[i++] = MMDAT;
                        dato_buffer[i++] = MMDAT;
                        dato_buffer[i++] = MMDAT;
                        dato_buffer[i]   = MMDAT;
                }
                for (i = 0; i < 512; i++)       //Impresión de los datos
                {
                        let=dato_buffer[i];
                        lcd_putch(let);
                        DelayMs(5);
                }
                j++;
        }
}
if(caso==7)                                 //Si es el caso es el primer archivo Mapa de bits "BMP"
{
        dir_inicial=413696;                 //dirección inicial del archivo BMP en la tarjeta
        dir_final=431029;                   //dirección final del archivo BMP en la tarjeta
        flag1=1;                            //bandera para impresión del archivo
        bucles=((dir_final-dir_inicial)/512);   //Número de repeticiones para observar todo el archivo
        j=0;
        while (j!=bucles)                   //Lectura del archivo dependiendo de su tamaño
        {
                mmc_send_cmd(MMC_READ_SINGLE_BLOCK, 413696 + 512*j, MMC_RESP_R1);
                DelayMs(2);
                for (i = 0; i < 512; i++)       //Almacenar los datos
                {
                        dato_buffer[i++] = MMDAT;
                        dato_buffer[i++] = MMDAT;
                        dato_buffer[i++] = MMDAT;
                        dato_buffer[i]   = MMDAT;
                }
                for (i = 0; i < 512; i++)       //Impresión de los datos
                {
                        let=dato_buffer[i];
                        lcd_putch(let);
                        DelayMs(5);
                }
                j++;
```

```
        }
}
if(caso==8)                             //Si es el caso es el archivo ASSEMBLER "ASM"
{
        dir_inicial=434176;             //dirección inicial del archivo ASM en la tarjeta
        dir_final=435331;               //dirección final del archivo ASM en la tarjeta
        flag1=1;                        //bandera para impresión del archivo
        bucles=((dir_final-dir_inicial)/512);   //Número de repeticiones para observar todo el archivo
        j=0;
        while (j!=bucles)               //Lectura del archivo dependiendo de su tamaño
        {
                mmc_send_cmd(MMC_READ_SINGLE_BLOCK, 434176 + 512*j, MMC_RESP_R1);
                DelayMs(2);
                for (i = 0; i < 512; i++)       //Almacenar los datos
                {
                        dato_buffer[i++] = MMDAT;
                        dato_buffer[i++] = MMDAT;
                        dato_buffer[i++] = MMDAT;
                        dato_buffer[i]   = MMDAT;
                }
                for (i = 0; i < 512; i++)       //Impresión de los datos
                {
                        let=dato_buffer[i];
                        lcd_putch(let);
                        DelayMs(5);
                }
                j++;
        }
}
if(caso==9)                             //Si es el caso es el Segundo archivo Mapa de bits "BMP"
{
        dir_inicial=438272;             //dirección inicial del Segundo archivo BMP en la tarjeta
        dir_final=471989;               //dirección final del Segundo archivo BMP en la tarjeta
        flag1=1;                        //bandera para impresión del archivo
        bucles=((dir_final-dir_inicial)/512);   //Número de repeticiones para observar todo el archivo
        j=0;
        while (j!=bucles)               //Lectura del archivo dependiendo de su tamaño
        {
                mmc_send_cmd(MMC_READ_SINGLE_BLOCK, 438272 + 512*j, MMC_RESP_R1);
                DelayMs(2);
                for (i = 0; i < 512; i++)       //Almacenar los datos
                {
                        dato_buffer[i++] = MMDAT;
                        dato_buffer[i++] = MMDAT;
                        dato_buffer[i++] = MMDAT;
                        dato_buffer[i]  =  MMDAT;
                }
                for (i = 0; i < 512; i++)       //Impresión de los datos
                {
```

```
                                        let=dato_buffer[i];
                                        lcd_putch(let);
                                        DelayMs(5);
                                }
                                j++;
                        }
                }
        }
}

bit kbhit()                                     //Función para verificar la tecla presionada
{
        tecla=KBSTA;                            //Guarda tecla presionada
        KBSTA=0x00;                             //Pone en 0 las banderas de teclado
        tecla = tecla&0x0F;                     //Enmascara valor de la tecla

        if(tecla==1)                            //Ingresar al contenido del archivo
        {
                while(tecla==0);
                ver_contenido();
                return OK;
        }
        if(tecla==8)                            //Seguir con la presentacion de archivos
        {
                while(tecla==0);
                return OK;
                flag1=0;
        }
}

void retardo()                                  //Función retardo para 2 segundos
{
        DelayMs(250);
        DelayMs(250);
        DelayMs(250);
        DelayMs(250);
        DelayMs(250);
        DelayMs(250);
        DelayMs(250);
        DelayMs(250);
}

void file_name(int i)                           //Función para imprimir el archivo de la tarjeta
{
        lcd_clear();                            //borra LCD
        lcd_goto(0x00);                         //cursor en posición 0
        //Obtiene  el nombre completo del archivo encontrado
        i=flag-2;
        let=dato_buffer[i]; i++;                //primer carácter sin sentido del archivo
        lcd_putch(let);                         //Impresión del primer carácter encontrado
```

```
DelayMs(5);                          //Retardo
let=dato_buffer[i]; i++;             //siguiente carácter del nombre del archivo
lcd_putch(let);                      //Impresión del carácter encontrado
DelayMs(5);                          //Retardo
let=dato_buffer[i]; i++;             //siguiente carácter del nombre del archivo
lcd_putch(let);                      //Impresión del carácter encontrado
DelayMs(5);                          //Retardo
let=dato_buffer[i]; i++;             //siguiente carácter del nombre del archivo
lcd_putch(let);                      //Impresión del carácter encontrado
DelayMs(5);                          //Retardo
let=dato_buffer[i]; i++;             //siguiente carácter del nombre del archivo
lcd_putch(let);                      //Impresión del carácter encontrado
DelayMs(5);                          //Retardo
let=dato_buffer[i]; i++;             //siguiente carácter del nombre del archivo
lcd_putch(let);                      //Impresión del carácter encontrado
DelayMs(5);                          //Retardo
let=dato_buffer[i]; i++;             //siguiente carácter del nombre del archivo
lcd_putch(let);                      //Impresión del carácter encontrado
DelayMs(5);                          //Retardo
let=dato_buffer[i]; i++;             //siguiente carácter del nombre del archivo
lcd_putch(let);                      //Impresión del carácter encontrado
DelayMs(5);                          //Retardo
let=dato_buffer[i]; i++;             //siguiente carácter del nombre del archivo
lcd_putch(let);                      //Impresión del carácter encontrado
DelayMs(5);                          //Retardo

let=0x2E;                            //carácter "."
lcd_putch(let);                      //Impresión del carácter
DelayMs(5);                          //Retardo

if(mp3==1)                           //Si el archivo encontrado fue una canción "MP3"
{
        flag=0;                      //bandera en cero
        mp3=0;                       //variable en cero para próxima lectura
        lcd_puts("MP3");             //Impresión de la extensión del archivo
        lcd_goto(0x40);              //Salto de línea
        lcd_puts("Archivo de audio"); //Impresión del tipo de archivo encontrado
        retardo();                   //Retardo para visualización
        lcd_goto(0x40);
        lcd_puts(">> SIG     x VER"); //Menú para ver siguiente archivo o para visualizar su contenido
}
if(txt==1)                           //Si el archivo encontrado fue de texto "TXT"
{
        flag=0;                      //bandera en cero
        txt=0;                       //variable en cero para próxima lectura
        lcd_puts("TXT");             //Impresión de la extensión del archivo
        lcd_goto(0x40);              //Salto de línea
        lcd_puts("Archivo de texto"); //Impresión del tipo de archivo encontrado
        retardo();                   //Retardo para visualización
```

```
            lcd_goto(0x40);
            lcd_puts(">> SIG    x VER");          //Menu para ver siguiente archivo o para visualizar su contenido
}
if(xls==1)                                      //Si el archivo encontrado fue una Hoja de Cálculo "XLS"
{
            flag=0;                             //bandera en cero
            xls=0;                              //variable en cero para próxima lectura
            lcd_puts("XLS");                    //Impresión de la extensión del archivo
            lcd_goto(0x40);                     //Salto de línea
            lcd_puts("Hoja de Cálculo");        //Impresión del tipo de archivo encontrado
            retardo();                          //Retardo para visualización
            lcd_goto(0x40);
            lcd_puts(">> SIG    x VER");        //Menu para ver siguiente archivo o para visualizar su contenido
}
if(pdf==1)                                      //Si el archivo encontrado fue un archivo Acrobat Reader "PDF"
{
            flag=0;                             //bandera en cero
            pdf=0;                              //variable en cero para próxima lectura
            lcd_puts("PDF");                    //Impresión de la extensión del archivo
            lcd_goto(0x40);                     //Salto de línea
            lcd_puts("Acrobat PDF");            //Impresión del tipo de archivo encontrado
            retardo();                          //Retardo para visualización
            lcd_goto(0x40);
            lcd_puts(">> SIG    x VER");        //Menu para ver siguiente archivo o para visualizar su contenido
}
if(ppt==1)                                      //Si el archivo encontrado fue un archivo de presentación "PPT"
{
            flag=0;                             //bandera en cero
            ppt=0;                              //variable en cero para próxima lectura
            lcd_puts("PPT");                    //Impresión de la extensión del archivo
            lcd_goto(0x40);                     //Salto de línea
            lcd_puts("Presentacion");           //Impresión del tipo de archivo encontrado
            retardo();                          //Retardo para visualización
            lcd_goto(0x40);
            lcd_puts(">> SIG    x VER");        //Menu para ver siguiente archivo o para visualizar su contenido
}
if(bmp==1)                                      //Si el archivo encontrado fue un archivo "BMP"
{
            flag=0;                             //bandera en cero
            bmp=0;                              //variable en cero para próxima lectura
            lcd_puts("BMP");                    //Impresión de la extensión del archivo
            lcd_goto(0x40);                     //Salto de línea
            lcd_puts("Mapa de bits");           //Impresión del tipo de archivo encontrado
            retardo();                          //Retardo para visualización
            lcd_goto(0x40);
            lcd_puts(">> SIG    x VER");        //Menu para ver siguiente archivo o para visualizar su contenido
}
if(asm==1)                                      //Si el archivo encontrado fue un archivo ASSEMBLER "ASM"
{
```

```
                flag=0;                                 //bandera en cero
                asm=0;                                  //variable en cero para próxima lectura
                lcd_puts("ASM");                        //Impresión de la extensión del archivo
                lcd_goto(0x40);                         //Salto de línea
                lcd_puts("Assembler");                  //Impresión del tipo de archivo encontrado
                retardo();                              //Retardo para visualización
                lcd_goto(0x40);
                lcd_puts(">> SIG    x VER");            //Menu para ver siguiente archivo o para visualizar su contenido
        }
}


void leer_cab()                                         //Función para identificar los archivos
{
        lcd_clear();                                    //borra LCD
        lcd_goto(0x00);                                 //cursor en posición 0
        conta=0;                                        //contador # de archivos encontrados
        i=0;                                            //variables en cero
        flag=0;
        caso=0;

        mmc_send_cmd(MMC_READ_SINGLE_BLOCK, 253952, MMC_RESP_R1);
        DelayMs(2);
        for (i = 0; i < 512; i++)                        //Almacenar las cabeceras de los archivos encontrados
        {
                dato_buffer[i++] =   MMDAT;
                dato_buffer[i++]  = MMDAT;
                dato_buffer[i++]  = MMDAT;
                dato_buffer[i]    = MMDAT;
        }
        for (i = 0; i < 512; i++)                       //Verifica si existe un archivo *.MP3
        {
                if(dato_buffer[i] =='M')                //Busca el primer carácter de la extensión
                {
                        i++;
                        if(dato_buffer[i]=='P')
                        {
                                i++;
                                if(dato_buffer[i]=='3')
                                {
                                        i=i-9;
                                        flag=i;
                                        mp3=1;
                                        file_name(i);
                                        i=i+10;
                                        conta++;
                                        caso++;
                                        while(!kbhit());
                                }
```

```
                }
        }
        if(dato_buffer[i]=='T')
        {
                i++;
                if(dato_buffer[i]=='X')
                {
                        i++;
                        if(dato_buffer[i]=='T')
                        {
                                i=i-9;
                                flag=i;
                                txt=1;
                                file_name(i);
                                i=i+10;
                                conta++;
                                caso++;
                                while(!kbhit());
                        }
                }
        }
        if(dato_buffer[i]=='X')
        {
                i++;
                if(dato_buffer[i]=='L')
                {
                        i++;
                        if(dato_buffer[i]=='S')
                        {
                                i=i-9;
                                flag=i;
                                xls=1;
                                file_name(i);
                                i=i+10;
                                conta++;
                                caso++;
                                while(!kbhit());
                        }
                }
        }
        if(dato_buffer[i]=='P')
        {
                i++;
                if(dato_buffer[i]=='D')
                {
                        i++;
                        if(dato_buffer[i]=='F')
                        {
```

```
                                        i=i-9;
                                        flag=i;
                                        pdf=1;
                                        file_name(i);
                                        i=i+10;
                                        conta++;
                                        caso++;
                                        while(!kbhit());
                                }
                        }
                }
                if(dato_buffer[i]=='P')
                {
                        i++;
                        if(dato_buffer[i]=='P')
                        {
                                i++;
                                if(dato_buffer[i]=='T')
                                {
                                        i=i-9;
                                        flag=i;
                                        ppt=1;
                                        file_name(i);
                                        i=i+10;
                                        conta++;
                                        caso++;
                                        while(!kbhit());
                                 }
                         }
                }
                if(dato_buffer[i]=='B')
                {
                        i++;
                        if(dato_buffer[i]=='M')
                        {
                                i++;
                                if(dato_buffer[i]=='P')
                                {
                                        i=i-9;
                                        flag=i;
                                        bmp=1;
                                        file_name(i);
                                        i=i+10;
                                        conta++;
                                        caso++;
                                        while(!kbhit());
                                }
                        }
```

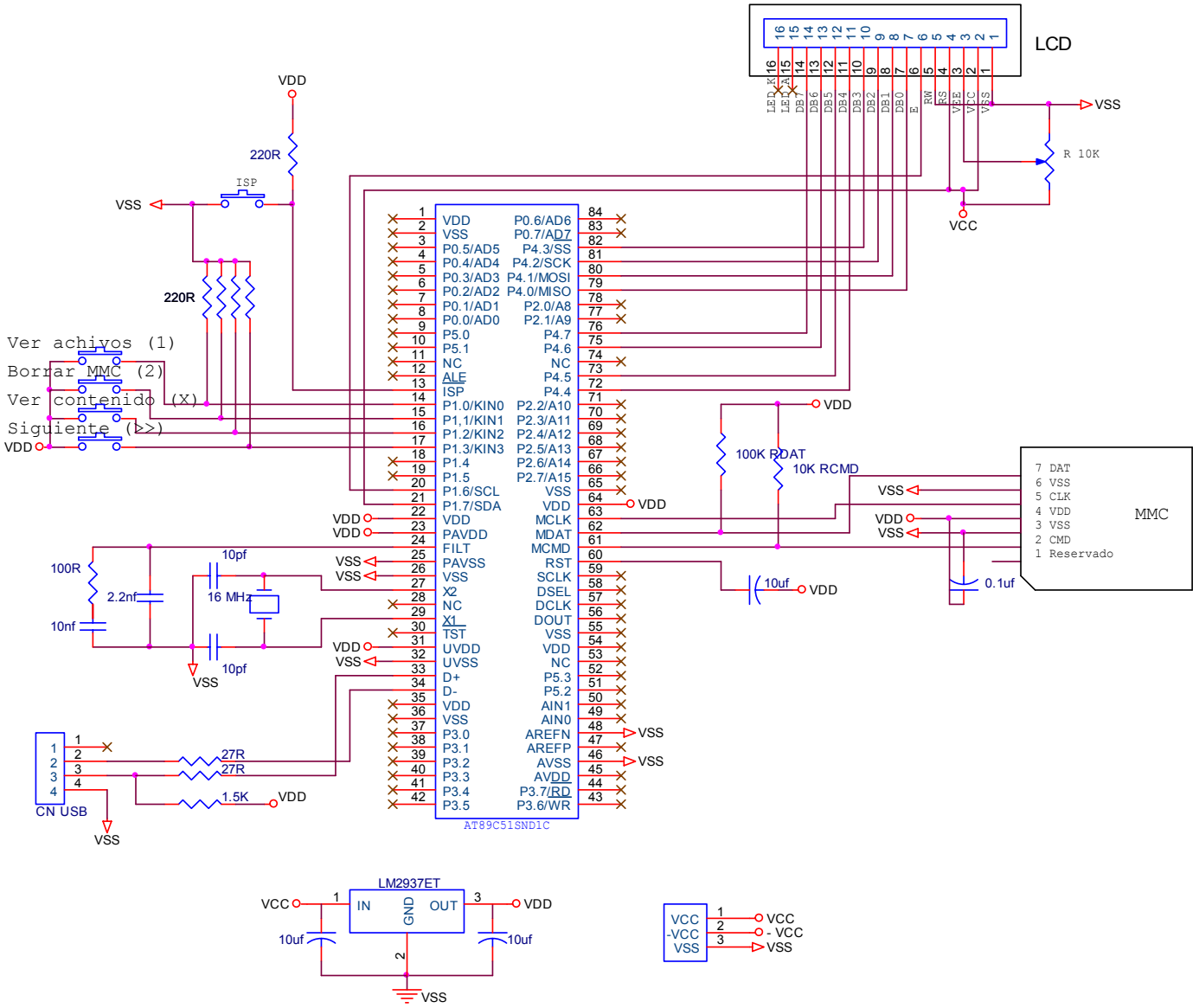
```
		}
		if(dato_buffer[i]=='A')
		{
			i++;
			if(dato_buffer[i]=='S')
			{
				i++;
				if(dato_buffer[i]=='M')
				{
					i=i-9;
					flag=i;
					asm=1;
					file_name(i);
					i=i+10;
					conta++;
					caso++;
					while(!kbhit());
				}
			}
		}
	}
	if(flag1==0)
	{
		if(conta==0)
		{
			lcd_clear();
			lcd_goto(0x00);
			lcd_puts("   No hay archivos");
		}
		else
		{
			lcd_clear();
			lcd_goto(0x00);
			lcd_puts("    Existen ");

			//encuentra con un límite para 9 archivos
			if (conta==1)		lcd_puts("  1");
			if (conta==2)		lcd_puts("  2");
			if (conta==3)		lcd_puts("  3");
			if (conta==4)		lcd_puts("  4");
			if (conta==5)		lcd_puts("  5");
			if (conta==6)		lcd_puts("  6");
			if (conta==7)		lcd_puts("  7");
			if (conta==8)		lcd_puts("  8");
			if (conta==9)		lcd_puts("  9");

			lcd_goto(0x40);
			lcd_puts("  archivo(s)");
		}
```

```
                DelayMs(250);
                DelayMs(250);                   //Retardo para visualización
                DelayMs(250);
                DelayMs(250);
                DelayMs(250);
                DelayMs(250);
                DelayMs(250);
                DelayMs(250);
                DelayMs(250);
                DelayMs(250);
                DelayMs(250);
                DelayMs(250);
        }
}

/*****************************
        MAIN
*****************************/
void main()                                     //función principal
{
        EnableX2();                             //Entra en Modo X2 ->Fcpu = Fper = Fosc
        lcd_init();                             //Inicialización LCD
        lcd_clear();                            //Borra LCD
        inicializa();                           //Inicialización MMC, referirse a librerías MMC suministradas
                                                //por ATMEL
        capacidad();                            //Obtiene la capacidad de la tarjeta

        while (1)
        {
                tecla=KBSTA;                    //Guarda tecla presionada
                KBSTA=0x00;                     //Pone en 0 las banderas de teclado
                tecla = tecla&0x0F;             //Enmascara valor de la tecla
                lcd_clear();                    //Borra LCD
                lcd_goto(0x00);
                lcd_puts("   1. Ver archivos"); //Imprime mensaje Menu principal
                lcd_goto(0x40);
                lcd_puts("2. Borrar MMC");      //Imprime mensaje Menu principal
                DelayMs(250);
                DelayMs(250);
                DelayMs(250);
                DelayMs(250);
                if(tecla==4)                    //Si se presiona la tecla de verificación de archivos
                {
                        while(tecla==0);
                        leer_cab();             //Función para obtener los archivos de la Tarjeta
                }
                if(tecla==2)                    //Si se presiona la tecla de borrar MMC
                {
                        while(tecla==0);
```

```
                        lcd_clear();                //Borra LCD
                        borra();                    //Función para borrar la Tarjeta
                        lcd_goto(0x00);
                        lcd_puts("   Borrado");     //Imprime mensaje
                        lcd_goto(0x40);             //Salto de línea
                        lcd_puts("     OK");        //Imprime mensaje
                        DelayMs(250);               //Retardo
                        DelayMs(250);
                        DelayMs(250);
                        DelayMs(250);
                }
        }
}
```

ANEXO 3

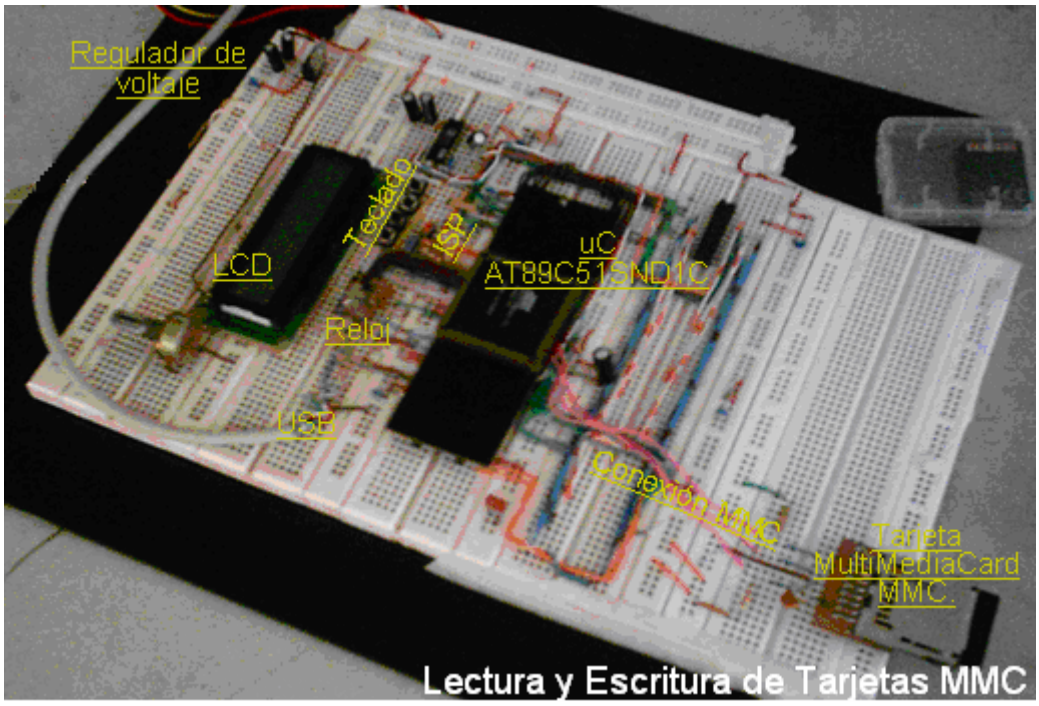
DIAGRAMA ESQUEMÁTICO

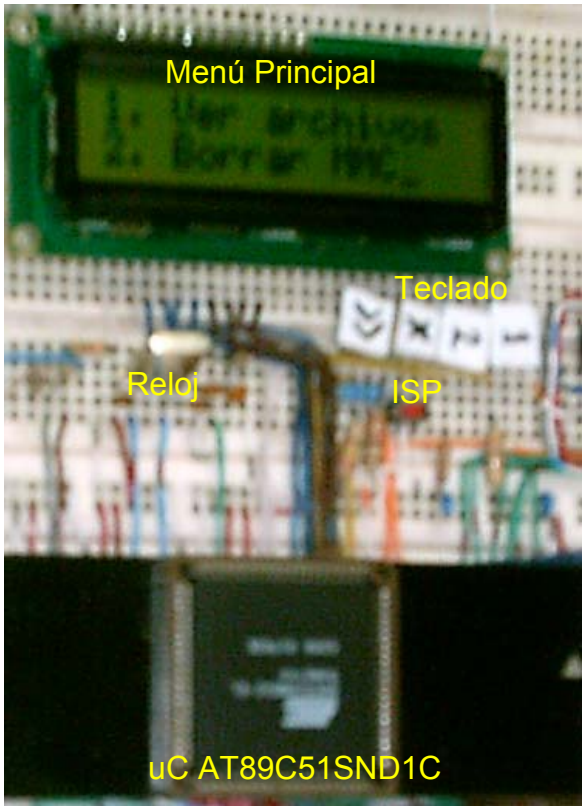

LCD
VDD
220R
ISP
VSS
220R
Ver achivos (1)
Borrar MMC (2)
Ver contenido (X)
Siguiente (>>)
VDD
R 10K
VCC
VSS
AT89C51SND1C
P0.6/AD6
P0.7/AD7
P4.3/SS
P4.2/SCK
P4.1/MOSI
P4.0/MISO
P2.0/A8
P2.1/A9
P4.7
P4.6
NC
P4.5
P4.4
P2.2/A10
P2.3/A11
P2.4/A12
P2.5/A13
P2.6/A14
P2.7/A15
MCLK
MDAT
MCMD
RST
SCLK
DSEL
DCLK
DOUT
P5.3
P5.2
AIN1
AIN0
AREFN
AREFP
AVSS
AVDD
P3.7/RD
P3.6/WR
P1.0/KIN0
P1,1/KIN1
P1.2/KIN2
P1.3/KIN3
P1.4
P1.5
P1.6/SCL
P1.7/SDA
PAVDD
FILT
PAVSS
X2
X1
TST
UVDD
UVSS
D+
D-
P3.0
P3.1
P3.2
P3.3
P3.4
P3.5
100K RDAT
10K RCMD
10uf
0.1uf
MMC
7 DAT
6 VSS
5 CLK
4 VDD
3 VSS
2 CMD
1 Reservado
100R
2.2nf
10nf
10pf
16 MHz
10pf
27R
27R
1.5K
CN USB
LM2937ET
IN
GND
OUT
10uf
10uf
VCC
-VCC
VSS
- VCC

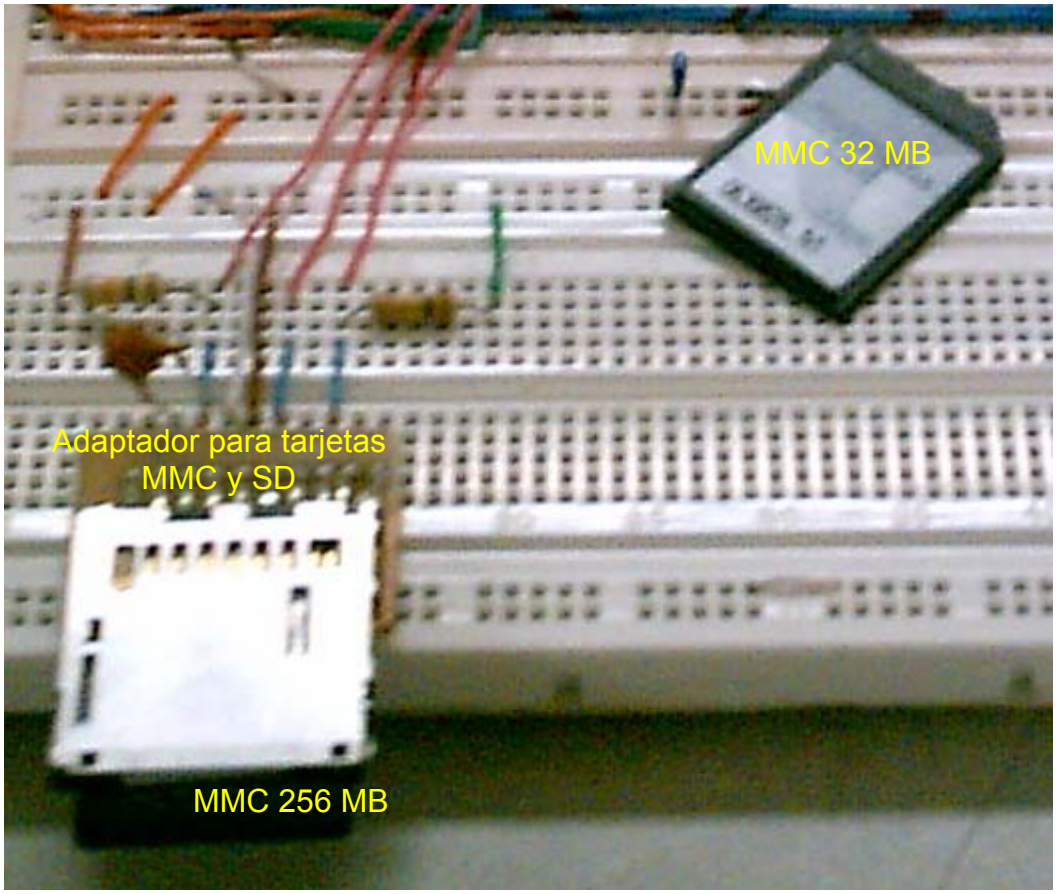
ANEXO 4

IMÁGENES DEL PROYECTO

IMÁGENES DEL PROYECTO






MMC 32 MB
Adaptador para tarjetas
MMC y SD
MMC 256 MB

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

GLOSARIO

EEPROM. Siglas en inglés de: Electrically Erasable Programmable Read-Only Memory, memoria de lectura solamente borrable y programable eléctricamente.

PCMCIA. Asociación Internacional de Tarjetas de Memoria para Computadoras Personales que fue fundada en 1989.

CF. Tarjeta CompactFlash regida a las especificaciones de la CFA

CFA. CompactFlash Association.

MS. Tarjeta Memory Stick de Sony.

SM. Tarjeta SmartMedia.

SSFDC. Siglas en ingles de: Solid State Floppy Disk Card.

xD. Siglas en inglés de: Extreme Digital. Tarjeta xD Picture.

SD. Tarjeta Secure Digital.

SDA. Secure Digital Association.

SPI. Comunicación estándar para tarjetas multimedia. Siglas en inglés de: Serial Peripherical Interface.

MMC. Tarjeta MultiMediaCard.

MMCA. MultiMediaCard Association.

RS-MMC. Reduce Size MMC.

CRC. Control de redundancia cíclica.

SDMI. Es un foro de más de 180 compañías y organizaciones que representan la tecnología de la información, consumidores electrónicos, tecnología de seguridad, industria magnetofónica mundial e Internet involucrados en proveer protección para la información multimedia digital.

ISP. In-System Programming.

PCB. Printed Circuit Board.

FAT. File Allocation Table. Conjunto de entradas en una tabla que contiene el nombre del archivo, el tamaño en bytes, la dirección del sector de inicio del archivo en el cluster (conjunto de sectores físicos del disco) y otra información de datos. La estructura de FAT es utilizada para identificar el siguiente cluster en un archivo y establecer el estado de asignación de un cluster: disponible o asignado.

MBR. Master Boot Root.

DATASHEETS

- Módulo MMC del microcontrolador ATMEL AT8xC51SND1C.
- Notes on Design of a MultiMediaCard™ System.
- Tarjeta MMC Hitachi HB28E016/D032/D064/B128MM2.

FECHA DE ENTREGA DEL PROYECTO

El presente proyecto de grado fue entregado a la Facultad de Ingeniería Electrónica y reposa en la Escuela Politécnica del Ejército desde:

Sangolquí, a _____ de Junio del 2006

Decano **Secretario Académico**

_____ _____

Ing. Xavier Martinez
TCRN. E.M.

Dr. Jorge Carvajal R

Autores

_____ _____

Edwin Xavier Mora N. **Marco Antonio Villalba O.**