UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DISEÑO E IMPLEMENTACIÓN DE UNA INTERFAZ DE REGISTRO DE DATOS PARA LA UNIDAD SÍSMICA SR04

T E S I S

QUE PARA OBTENER EL TÍTULO DE :

INGENIERO ELÉCTRICO ELECTRÓNICO

(Á R E A : E L E C T R Ó N I C A)

P R E S E N T A :

HUMBERTO HERNÁNDEZ ARIAS

DIRECTOR DE TESIS: M. I. LAURO SANTIAGO CRUZ

MÉXICO, D.F. 2010

AGRADECIMIENTOS:

A la ***Universidad Nacional Autónoma de México*** *por darme la oportunidad de realizar una carrera y por haberme dado las experiencias que constituyen la mejor época de mi vida.*

A la ***Facultad de Ingeniería*** *y a todos mis profesores.*

Al ***Instituto de Ingeniería*** *de la UNAM por darme todas las facilidades para la realización de este proyecto.*

Al mi Maestro el ***M.I. Lauro Santiago Cruz*** *por el apoyo, confianza y amistad brindados.*

A mis sinodales por haber revisado este trabajo y hecho valiosas observaciones para mejorarlo.

A ***todos*** *mis amigos por su amistad y las experiencias que hemos compartido.*

Quiero agradecer en especial a Gerardo, Jesus y Pedro, y a mis compañeros del laboratorio de Instrumentación Aleidi, Anuar, Angelica, Belit, Christian, David, Edgar, Germán, Mario, Martín, Nelly y William.

Dedicatorias:

Para Ixchel A. Sandoval García:

Gracias por todo el amor y el apoyo que me has brindado.

Para mis padres Alma Rosa y Germán.
Para mis hermanos Eduardo, Luis, Silvia y Tania.

Para la familia de Ixchel, que me ha acogido como uno más de sus miembros: Gracias.

CONTENIDO

ÍNDICE DE FIGURAS

CAPÍTULO 2

CAPITULO 3

CAPITULO 4

ÍNDICE DE TABLAS

CAPÍTULO 2

CAPÍTULO 3

CAPÍTULO 4

PRÓLOGO

El Instituto de Ingeniería de la Universidad Nacional Autónoma de México tiene como principal objetivo el contribuir al desarrollo del país al realizar investigaciones orientadas a problemas generales de ingeniería, formando recursos humanos y generando tecnología nacional. En particular, en los campos de la sismología y de la instrumentación se cuenta con una amplia experiencia, que ha permitido desarrollar nuevas tecnologías en dichos campos, como son los sistemas de registro sísmico.

En el presente trabajo se describe el proceso de diseño y desarrollo de un sistema de registro que se acopla a la unidad sísmica SR04, con lo que se pretende dar autonomía en la colección y registro de datos a este tipo de sistemas. El propósito de la interfaz propuesta es hacer posible el uso de la unidad sísmica, empleada por los especialistas del Instituto de Ingeniería, sin requerir un equipo de cómputo para efectuar un registro de datos por periodos prolongados de tiempo.

El sistema desarrollado permite configurar la frecuencia de muestreo a la que trabaja la unidad SR04, teniendo la posibilidad de escoger entre tres valores distintos, enviando el comando adecuado; además, es posible verificar que la unidad sísmica se encuentra trabajando correctamente, contando con un display en el que se pueden graficar los datos digitalizados de los sensores que se conectan a cada canal de dicha unidad y mostrando la estampa de tiempo que proporciona el GPS, esto es día, mes y año actual y la hora GMT en un formato de hora, minuto y segundo de la última muestra. Finalmente, la interfaz permite colectar y almacenar los datos provenientes de la unidad SR04 en tarjetas de memoria flash SD (*SD Card*), dando al usuario la posibilidad de configurar el tiempo de registro y generando archivos que pueden ser leídos por casi cualquier computadora, al implementar los sistemas de archivos FAT16 ó FAT32.

Este trabajo se divide en cinco capítulos, desarrollados de la siguiente manera:

Capítulo 1. En este capítulo se da una breve introducción a la tesis, presentando el problema que se propone resolver y la propuesta de solución planteada.

Capítulo 2. Dentro de este capítulo se comentan los temas básicos que permiten comprender el proceso de desarrollo de la Interfaz de registro. Dentro de los principales temas tratados tenemos: topologías y protocolos de comunicación de las tarjetas de memoria SD, especificaciones de los sistemas de archivos FAT, microcontroladores y una descripción detallada de la unidad sísmica SR04.

Capítulo 3. Aquí se plantean formalmente las necesidades que debe cubrir el proyecto, presentando el desarrollo del sistema propuesto para la solución de las mismas. Como partes principales se tienen: el desarrollo del *hardware* y el desarrollo del *software* constituido por las funciones y rutinas programadas para el microcontrolador.

Capitulo 4. Aquí se describen las pruebas que se realizaron a las distintas partes tanto de *hardware* como de *software* que conforman a la interfaz de registro, y las pruebas que se realizaron al sistema en su etapa final. Se comentan las pruebas al display, pruebas de lectura y escritura de las tarjetas de memoria SD, pruebas al sistema de archivos implementado y pruebas de registro, entre otras.

Capitulo 5. Finalmente, dentro de este capítulo se proporcionan los resultados que fueron obtenidos con el desarrollo de este trabajo. Así mismo, se presentan las conclusiones a las que se llegaron y algunas recomendaciones para mejorar el sistema desarrollado.

También se presenta un pequeño glosario con los términos en inglés que son utilizados a lo largo del presente escrito, incluyendo, además, algunos anexos importantes, y por último la bibliografía utilizada.

1. INTRODUCCIÓN

La sismología es la rama de la geofísica que se encarga de estudiar las vibraciones producidas natural o artificialmente en la Tierra. Sus objetivos son: estudiar la propagación de ondas sísmicas; determinar las causas que dan origen a éstas y proporcionar herramientas para prevenir los daños relacionados a ellas.

En el área de Ingeniería Sismológica del Instituto de Ingeniería de la UNAM, se cuenta con equipos de detección y adquisición de eventos, que son llevados a campo para hacer estudios, que implican el registro de datos apoyándose con computadoras portátiles.

Uno de los equipos utilizados por los especialistas del Instituto es la unidad sísmica SR04. Esta unidad es propiamente un digitalizador de datos, con todos los componentes necesarios para convertir una señal sísmica analógica a su forma digital. Dicha unidad permite la detección de los eventos de interés, enviando los datos obtenidos por el puerto serie con el que cuenta, a un equipo de cómputo, para que éste último los guarde en archivos y puedan ser procesados posteriormente.

Un problema importante que presenta la unidad SR04 es el hecho que para colectar, guardar y procesar la información digitalizada siempre debe estar conectada a una computadora, personal o *laptop*. En aplicaciones en donde se requiere únicamente colectar y guardar dicha información, para largos periodos de muestreo, esto no sería posible sin la computadora.

Hay aplicaciones en las que los estudios a efectuar con la unidad SR04 requieren de la realización de pruebas de campo, que pueden llegar a durar días y en las que los diversos factores ambientales y de terreno dificultan el traslado de un equipo de cómputo a ellos, además del costo de dicho equipo

Dado lo anterior, en el presente trabajo se propone el desarrollo de una interfaz que permita configurar la colección, visualización y almacenamiento de datos, liberando así a la SR04 del uso de cualquier computadora. El registro de datos deberá hacerse en unidades de

memoria con tecnología actual y con la característica de ser portátil, de preferencia *tarjetas de memoria flash*. La información adquirida será compatible con los sistemas de archivos FAT (***File Allocation Table***) de mayor uso en la actualidad: FAT16 y FAT32, y podrá ser leída directamente de la memoria por una PC que soporte estos sistemas de archivos.

El sistema a desarrollar permitirá, además, la visualización del comportamiento de los tres sensores sísmicos de la unidad SR04, a través de un display gráfico. En la pantalla, se mostrará un menú de opciones que incluye:

- Selección de la frecuencia de muestreo de la unidad sísmica
- Visualización del comportamiento del canal deseado o de la marca de tiempo GMT
- Configuración del tiempo de registro

Se desarrollará el sistema construyendo el *hardware* específico y programando las tareas requeridas (*software*).

El *hardware* lo conformará un microcontrolador, un *display* gráfico, un teclado, un conector para memorias *SD card*, un transceptor para la comunicación USB, otro para la comunicación RS232 y algunos componentes adicionales.

El software estará integrado por las diversas tareas programadas en el microcontrolador, éstas comprenden:

- Manejo de display y teclado
- Comunicación con la memoria flash
- Implementación de un sistema de archivos FAT
- Configuración del registro de los datos de la unidad sísmica SR04
- Configuración de la unidad sísmica SR04
- Adquisición y registro de datos
- Comunicación con una PC por medio de un puerto USB

Cabe mencionar que existe un antecedente de este trabajo, en el que se realiza el registro de datos en tarjetas de memoria flash SD, implementando los sistemas de archivos FAT16 y FAT32. En dicho trabajo se implementa un sistema basado en un microcontrolador que se comunica con tarjetas SD, generando un archivo de tamaño fijo que puede ser leído por casi cualquier computadora, a través de un editor de texto.

Después de esta breve introducción, en el siguiente capítulo trataremos los temas básicos que permitirán la comprensión de las partes que componen al sistema propuesto.

2. GENERALIDADES DEL SISTEMA

En este capítulo se presentarán, de manera general, temas básicos relacionados con el sistema propuesto. Los temas a tratar incluyen la adquisición y digitalización de datos, las memorias flash SD y sus topologías de comunicación, los sistemas de archivos FAT, la unidad sísmica SR04, los microcontroladores y por último se comentará sobre dispositivos periféricos como el teclado y el display gráfico.

2.1. Adquisición y digitalización de datos

El objetivo de cualquier sistema de adquisición de datos es recabar información referente a alguna característica de un fenómeno físico que nos interese, obteniendo así, determinados elementos para el análisis del mismo. Esto se lleva a cabo a partir de tomar muestras del sistema analógico bajo estudio, obteniendo información que puede ser almacenada o procesada por algún sistema digital.

Un sistema típico de adquisición de datos está compuesto, básicamente, por los bloques mostrados en la fig. 2.1.

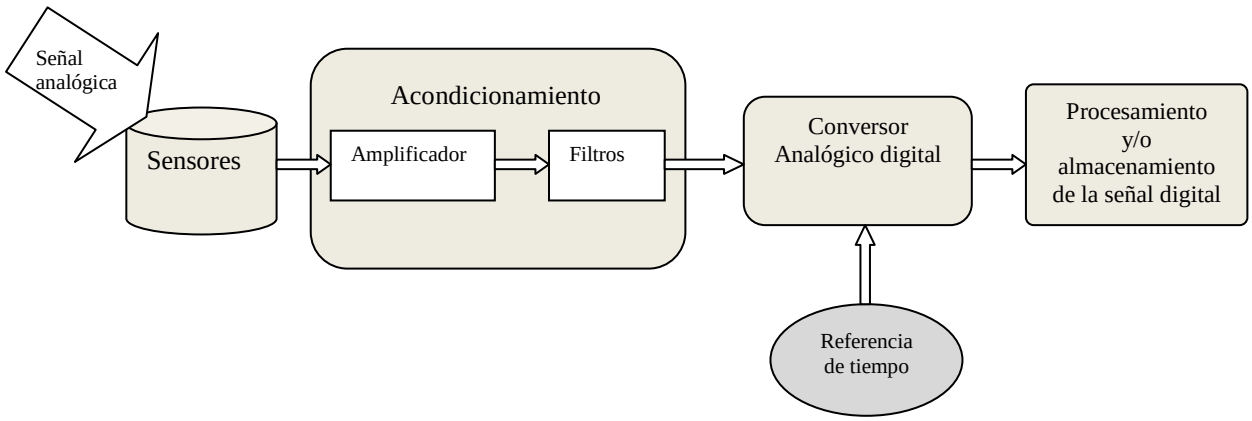


Figura 2.1. Sistema típico de adquisición y registro de datos.

Primeramente se tiene la variable a medir, en seguida, la parte que capta la señal analógica externa, mediante sensores, con los cuales obtiene un voltaje o corriente. Esta señal de voltaje o corriente es acondicionada por amplificadores y filtros, facilitando con ello su manejo. En la siguiente etapa se digitaliza la señal acondicionada haciendo uso de un conversor analógico a digital (*ADC: Analog to Digital Converter*) y se manda a través de una interfaz hacia alguna herramienta electrónica digital de procesamiento o registro de datos, donde se puede hacer el tratamiento, análisis, compresión o almacenamiento de los datos adquiridos que hacen referencia a la variable que se pretende estudiar.

A la señal digital generada se le debe agregar una referencia o marca de tiempo, la cual es enviada al dispositivo de control del ADC, que agrega dicha información a las muestras recabadas. La marca de tiempo es con el fin de que cada dato adquirido esté etiquetado.

En general, un sistema de registro de señales sísmicas está formado por una computadora de propósito general y el software necesario y específico para realizar dicha función. El trabajo de estos sistemas se puede dividir en los siguientes puntos:

- Leer los datos provenientes del sistema de adquisición.
- Plasmar la etiqueta de tiempo, si es que no cuentan con ella.
- Almacenar la información, organizándola en archivos con algún formato.
- Servir de interfaz proporcionando la comunicación con la unidad de adquisición, haciendo posible la configuración de ésta última.

La información obtenida a partir de un sistema de adquisición y digitalización debe ser registrada en algún medio de almacenamiento a largo plazo, de tal forma que pueda ser procesada y analizada posteriormente.

En la actualidad se dispone de una amplia variedad de dispositivos y medios para almacenar información de forma masiva, entre los más comunes contamos a los siguientes:

- Cintas magnéticas
- Discos duros
- Discos flexibles
- Discos ópticos (CD – ROM/RW, DVD – ROM/ RW)
- Tarjetas de memoria flash (en sus diversos formatos)
- Discos duros de estado solido
- Unidades USB de memoria flash

Estos medios han ido evolucionando, incrementando la capacidad de almacenamiento y en el caso de los discos duros y tarjetas de memoria flash, se ha logrado disminuir el tamaño requerido para registrar información.

2.2. Memorias Flash *Secure Digital*

2.2.1. Memoria Flash

Las memorias CMOS (*Complementary Metal-Oxide-Semiconductor*) pueden dividirse en dos categorías principales:

- Memorias de acceso aleatorio (*RAM: Random Access Memory*), que son del tipo volátil, es decir, pierden la información retenida cuando su fuente de alimentación es apagada.

- Memorias de sólo – lectura (*ROM: Read Only Memory*) siendo éstas de tipo no-volátil; conservan la información aun cuando están desconectadas de la fuente de alimentación.

Las memorias EEPROM (*Electrically-Erasable Programmable Read-Only Memory*) son un tipo de memoria ROM, que puede ser programada, borrada y reprogramada eléctricamente de 100 000 a 1 000 000 de veces.

Las memorias Flash representan una mejora de las celdas EEPROM y, al igual que éstas, se programan, borran y reprograman eléctricamente. Los primeros modelos de Flash requerían de una fuente de alimentación externa y de algún dispositivo para gestionar las operaciones de escritura. Actualmente, las memorias Flash que podemos encontrar vienen en conjunto con microcontroladores embebidos, que se encargan de las tareas requeridas, ofreciendo la posibilidad de trabajar con sectores, en vez de un solo bit (como es el caso de las EEPROM) y de utilizar sólo una fuente de alimentación.

Dos de las principales aplicaciones en donde podemos encontrar a las memorias Flash son: la integración de memoria no-volátil, en sistemas lógicos, principalmente en microprocesadores, y en la creación de elementos de almacenamiento, como tarjetas de memoria o discos duros de estado sólido compuestos por arreglos de celdas Flash.

Las memorias Flash son la tecnología dominante en aplicaciones que requieren grandes cantidades de almacenamiento de información, a un costo razonable; han ganado popularidad en aplicaciones portátiles debido a su bajo consumo de energía, tamaño reducido, durabilidad, bajo costo y a que no tienen partes móviles.

Una de las formas más comunes en la que podemos encontrar memorias flash es en las tarjetas de memoria. En la tabla 2.1 se muestran los formatos más comunes de tarjetas de memoria flash que existen.

Uno de los formatos de tarjeta de memoria flash, más populares y ampliamente usado, es el Secure Digital (*SD card memory*), creado en colaboración por Panasonic Corporation, SanDisk Corporation y Toshiba Corporation. Estas empresas conforman actualmente la asociación SD (*SD Association*).

Formato	Sigla	Capacidad máxima
PC Card	PCMCIA	8 GB
Compact Flash I	CF – I	12 GB
Compact Flash II	CF – II	48 GB
Smart Media	SM / SMC	128 MB
Memory Stick	MS	128 MB
Memory Stick Duo	MSD	256 MB
Memory Stick PRO Duo	MSPD	32 GB
Memory Stick PRO-HG Duo	MSPDX	32 GB
Memory Stick Micro M2	M2	32 GB
MiltiMedia Card	MMC	4 GB
Reduce Size Multimedia Card	RE – MMC	2 GB
MMC micro card	MMC micro	2 GB
Secure digital card	SD	2 GB
miniSD card	miniSD	2 GB
microSD card	microSD	2 GB
SD card High Capacity (mini, micro)	SDHC	32 GB
xD-Picture card	xD	2 GB

Tabla 2.1. Formatos y capacidades de memoria flash[1].

2.2.2. Especificaciones de las Memorias Flash SD (versión 2.0)

En el diseño del sistema en cuestión se optó por las tarjetas SD, para realizar el registro de los datos provenientes de la unidad sísmica. Para esto, se hizo uso de la versión simplificada de las especificaciones técnicas (*simplified Version of phisical layer specification*), que es un subconjunto de las especificaciones completas (*Phisical Layer Specification*), y puede obtenerse de manera gratuita desde la página de la *SD Association.* En adelante, nos referiremos a éstas como las especificaciones técnicas o simplemente como especificaciones.

Las especificaciones técnicas describen la interfaz física y el protocolo de comandos utilizado por las tarjetas de memoria flash SD. Su propósito es definir a las tarjetas de memoria SD: su estructura y su manipulación. Entre las características principales de la memoria flash SD se encuentran las siguientes, ver figuras 2.2 y 2.3:

- Diseñadas para su uso en aplicaciones portátiles y de escritorio.
- Capacidad de memoria:
    - Memorias SD de capacidad estándar: hasta e incluyendo 2 GB.

[1] Información obtenida en las páginas web de los fabricantes.

  - Memorias SD de alta capacidad: más de 2 GB (limitada hasta e incluyendo 32 GB).
- Margen de voltaje
  - Tarjetas de memoria de voltaje alto: voltaje de operación de 2.7 - 3.6 V.
  - Tarjetas de memoria que soportan tanto voltaje bajo (aún no definido) como voltaje alto (2.7 - 3.6 V).
- Diseños de tarjetas de sólo lectura y de lectura- escritura.
- Modo default: reloj de 0 - 25 MHz y velocidad de la interfaz hasta 12.5 MB/s (utilizando 4 líneas de datos).
- Modo de alta velocidad: reloj de 0 - 50 MHz y velocidad de la interfaz hasta 25 MB/s (utilizando 4 líneas de datos).
- Corrección de errores de campos de memoria.
- Retirar la tarjeta durante la lectura nunca causará daños en el contenido de ésta.
- Mecanismo de protección contra escritura.
- Detección de tarjeta: inserción y extracción.
- Se define el protocolo del canal de comunicación con los atributos mencionados en la tabla 2.2.

Canal de comunicación de las memorias SD card
Canal de comunicación de seis vías (Reloj, comandos, 4 líneas de datos)
Transferencia de datos con protección de errores
Transferencia de datos por bloque o múltiples bloques.

Tabla 2.2. Características del Canal de comunicación de las memorias SD.

- Factor de forma:
  - Tarjetas de memoria SD de tamaño estándar.
  - Tarjetas de memoria miniSD.
  - Tarjetas de memoria microSD.

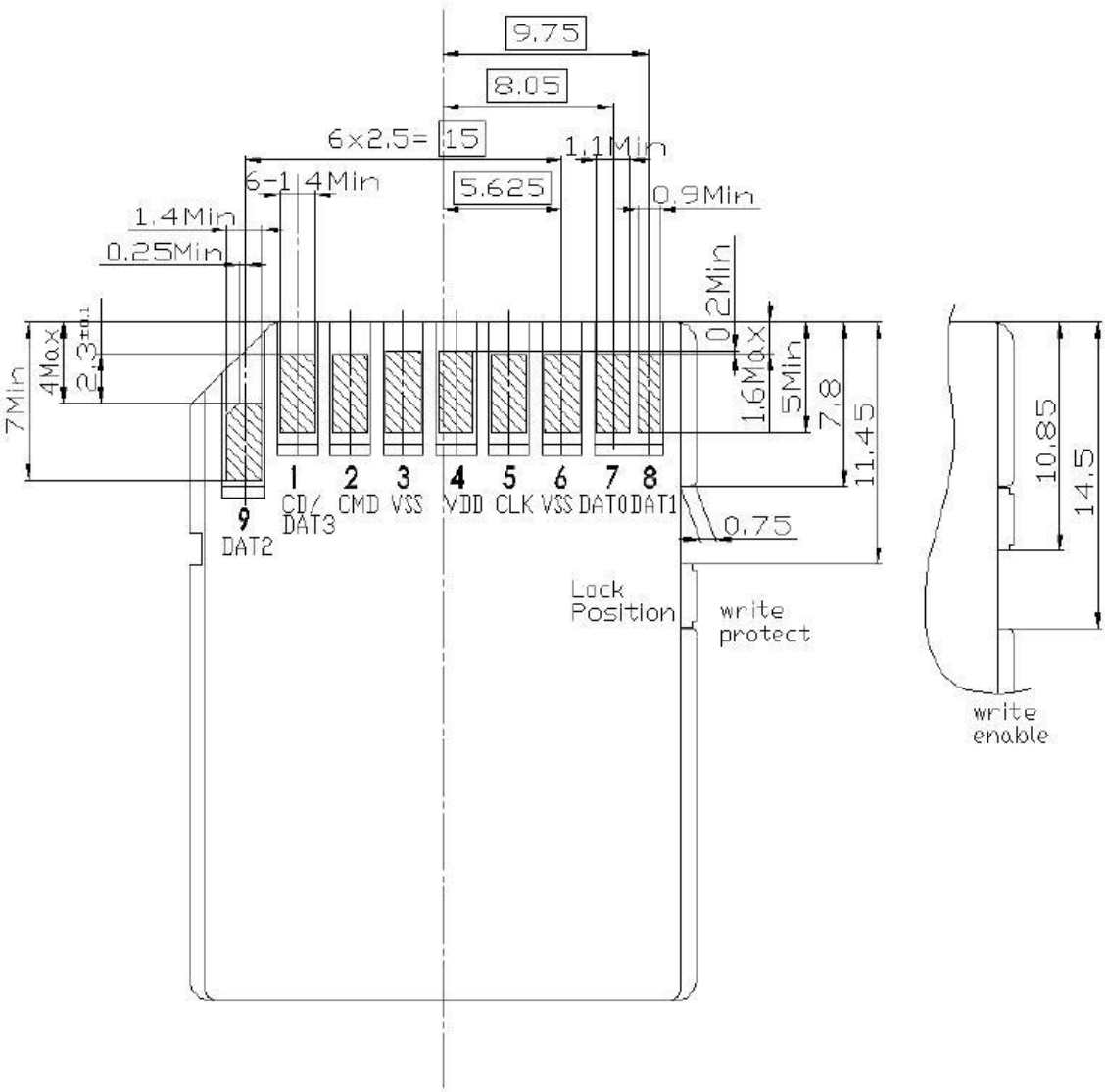


Figura 2.2. Forma y dimensiones de las tarjetas de memoria flash SD.

Tarjeta SD estándar

Tarjeta SD de alta capacidad

Figura 2.3. Apariencia de las tarjetas de memoria SD.

- Espesor del tamaño estándar de tarjeta, definido como 2.1 mm (normal) y 1.4 mm (tarjeta de memoria SD delgada).

Además de las características mencionadas, existe una clasificación de acuerdo a la velocidad, se definen cuatro clases y se indica el desempeño mínimo de las tarjetas:

Clase 0 - Estas tarjetas no especifican su desempeño.

Clase 2 - Con velocidad de 2 MB/s o más.

Clase 4 - Con velocidad de 4 MB/s o más.

Clase 6 - Con velocidad de 6 MB/s o más.

2.2.3. Topologías de comunicación de las tarjetas de memoria SD

Para las tarjetas SD se especifican dos protocolos alternativos de comunicación: SD y SPI. Las aplicaciones pueden escoger entre estos modos de comunicación. La tarjeta detecta automáticamente el modo seleccionado a partir del primer comando de *reset* y esperará las siguientes operaciones de comunicación en el mismo modo.

Bus SD

El modo SD o bus SD es el modo nativo desarrollado para este tipo de dispositivos. En éste, la comunicación está basada en el flujo de bits de comandos y datos, iniciado por un *bit de inicio* y terminada por el *bit de paro*; se realiza a través de una interfaz de 9 terminales (reloj, comandos, 4 terminales para datos y 3 para alimentación). Está diseñada para trabajar a una frecuencia máxima de 50 MHz, a bajo voltaje. En la figura 2.4 se muestra la topología del bus SD.

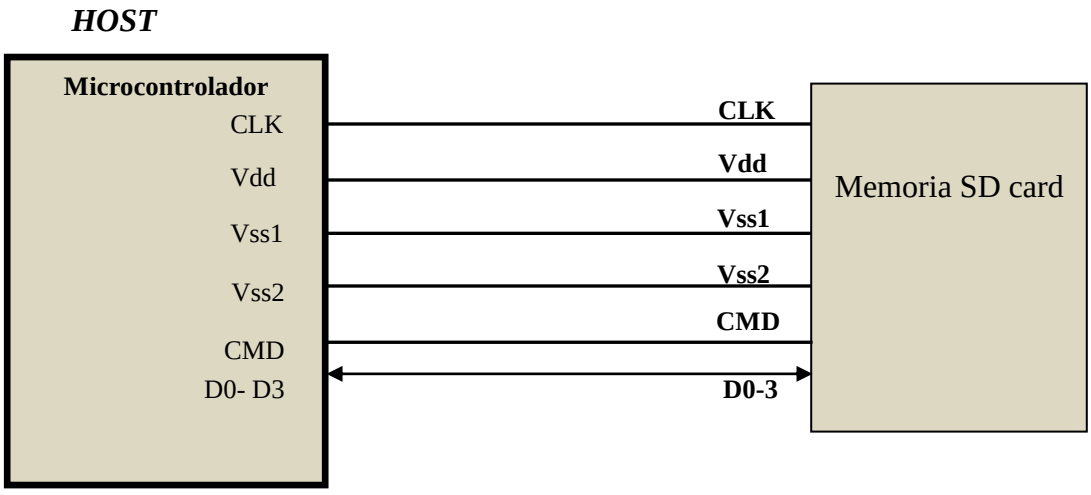


Figura 2.4. Topología del bus SD.

El bus SD se conforma por las siguientes señales:

CLK: Señal de reloj del anfitrión a la tarjeta.
CMD: Señal bidireccional Comando/Respuesta.
DAT0- DAT3: 4 señales bidireccionales de datos
VDD, VSS1, VSS2: Alimentación y tierra.

En el modo Bus SD de comunicación se tiene la capacidad para configurar el número de líneas de datos de manera dinámica. Después de ser energizada, la tarjeta SD utiliza únicamente la línea DAT0 para la transferencia de datos. Enseguida del proceso de inicialización, el *host* puede cambiar el número de líneas de datos activas.

Bus SPI

El modo SPI (*Serial Peripheral Interface*) consiste en un protocolo de comunicación secundario soportado por las memorias SD. Este modo es una división del protocolo de las tarjetas de memoria SD, diseñado para comunicarse a través del canal SPI, encontrado en la mayoría de los microcontroladores del mercado.

La interfaz es seleccionada durante el primer comando de reset (CMD0), después de ser energizada la tarjeta. En el estándar SPI se define, únicamente, la conexión física; la implementación del bus SPI en las tarjetas de memoria SD utiliza sólo una parte del protocolo y de la serie de comandos; se utilizan, solamente, 7 de las 9 señales del bus SD.

La ventaja del modo SPI es la capacidad que se tiene para implementar un *HOST* a partir de un microcontrolador de propósito general, facilitando el diseño. La desventaja que se tiene es la reducción del desempeño del sistema, que se puede observar en la reducción de la velocidad de comunicación, al poder utilizarse únicamente una línea para datos. En la figura 2.5 se presenta la topología del bus SPI.

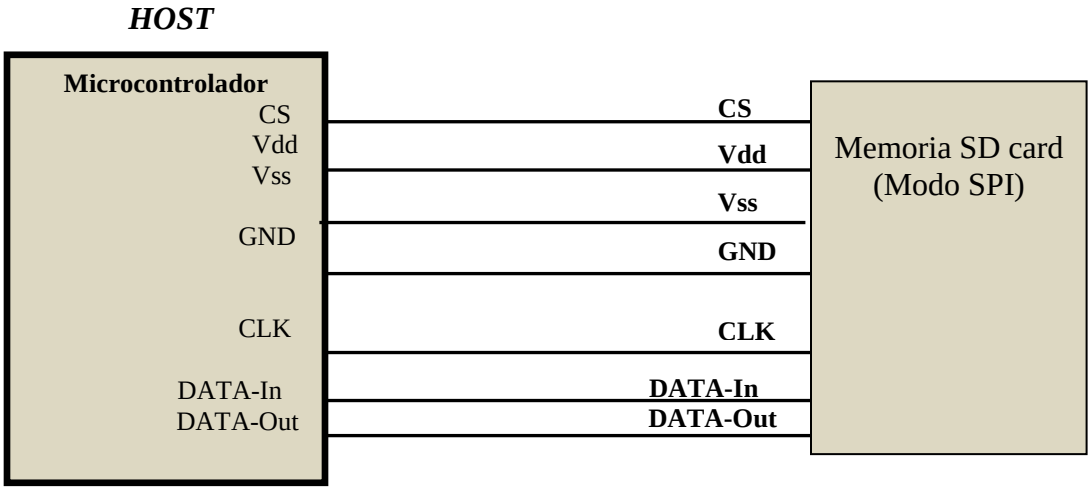


Figura 2.5. Topología del bus SPI.

El canal SPI se compone de las siguientes señales:

CS: Señal de selección de *chip* (*Chip Select*): Host → Memoria SD
CLK: Señal de reloj: Host → Memoria SD
DATAIN: Señal de datos: *H*ost → Memoria SD
DATAOUT: Señal de datos: Memoria SD →*Host*

2.2.4. Protocolo de comunicación SPI para las tarjetas SD

Como ya se ha comentado, el protocolo del bus SPI comprende sólo una parte del protocolo SD, sin embargo, existen diferencias significativas. A continuación se presentan las características y consideraciones que definen al protocolo SPI.

- A diferencia del protocolo del bus SD, la comunicación en el canal SPI está orientada a bytes; todos los comandos y bloques de datos están conformados por bytes de 8 bits.

- En SPI existen operaciones de comandos, respuestas y transferencia de datos.

- Todas las transacciones entre el *host* y las tarjetas son controladas por el *host* (maestro); el *host* inicia cada actividad del bus, activando la señal CS.

- Las tarjetas siempre responden a los comandos.

- Si la tarjeta encuentra un problema de datos en la operación de lectura, responderá con un error que reemplazará al bloque de datos esperado.

- En el caso de las tarjetas con capacidad estándar (*Standard Capacity Memory Card*), un bloque de datos puede ser tan grande como un *bloque de escritura* (512 Bytes) y tan pequeño como un Byte.

- En cuanto a las tarjetas de memorias con alta capacidad (*High Capacity SD Memory Card*), el tamaño de los bloques de datos es única: 512 bytes.

- Los comandos de protección contra escritura no son soportados.

- El código de detección de errores CRC (*Cyclic Redundancy Code*) está deshabilitado y es ignorado por el *host.*

- El *host* debe tratar a todas las tarjetas como si fueran de clase 0, en cuanto a velocidad.

Selección del modo e inicialización

Siempre que la tarjeta SD es energizada enciende en el modo SD; entrará en el modo SPI si la señal CS se encuentra activa durante la recepción del comando de reset (CMD0), una vez que se encuentra en este modo envía la respuesta R1. En la figura 2.6 se muestra el diagrama de flujo de la secuencia de comandos necesaria para la inicialización de las tarjetas SD en el modo SPI.

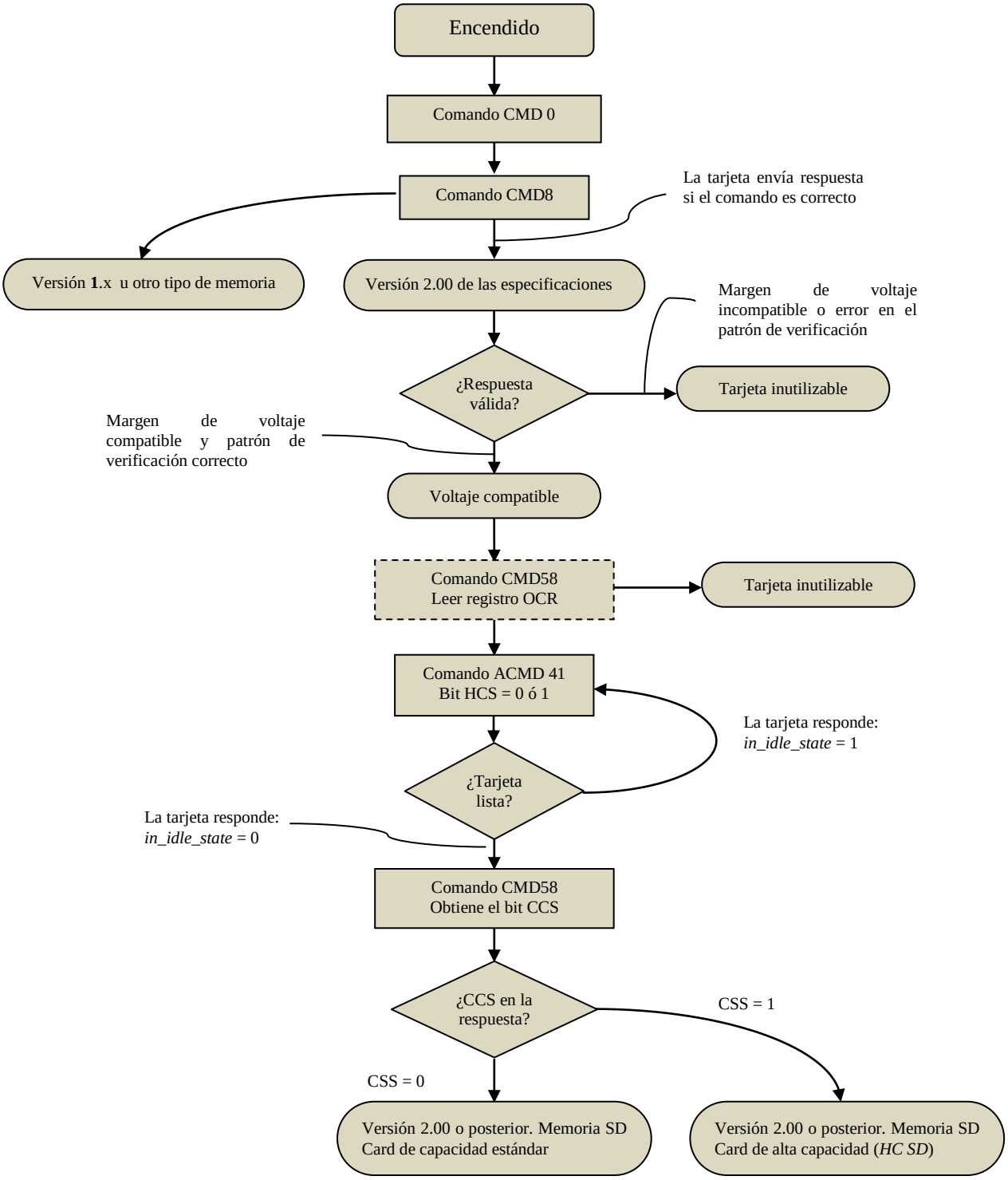


Figura 2.6. Diagrama de flujo de la inicialización de la tarjeta de memoria SD en el modo SPI.

La inicialización de la tarjeta se lleva a cabo enviado la secuencia de comandos indicada en el diagrama de flujo anterior. La tarjeta analizará el argumento de éstos y responderá según corresponda. Al final del proceso, el *host* tendrá información suficiente para saber de qué versión de tarjeta se trata y si puede o no trabajar con ella. Las funciones realizadas por los comandos son las siguientes:

El comando CMD8 (*Envía las condiciones de operación de la interfaz*) es utilizado para verificar las condiciones de operación de la conexión establecida entre la tarjeta de memoria SD y el *host*. La tarjeta comprueba la validez de estas condiciones analizando el argumento del CMD8, a su vez, el *host* lo hace analizando la respuesta R7, en la que se envía el voltaje aceptado por la tarjeta.

El comando CMD58 (*Lee OCR:Operation Condition Register*) es utilizado por el *host* como un mecanismo para identificar a las tarjetas que no pueden trabajar con el margen de voltaje V_{DD} suministrado. Si existe incompatibilidad de voltajes no se continuará con el proceso de inicialización. La respuesta asociada a este comando es R3.

El comando ACMD41 (*Envío de condiciones de operación*) es utilizado para comenzar la secuencia de inicialización y para verificar si la tarjeta ha completado ésta. Es obligatorio que el *host* envíe el CMD8 antes de enviar por primera vez el ACMD41, de esta manera, se expanden las funciones de los comandos CMD58 y ACMD41; se podrá utilizar el campo HCS (*High Capacity Suport*) en el argumento del ACMD41 y el CCS (*Card Capacity Status*) en la respuesta al CMD58.

El bit "*in_idle_state*" en la respuesta R1 al ACMD41 es utilizado por la tarjeta para informar al *host* si el procedimiento de inicialización ha terminado; el tener dicho bit un valor de "1" indica que la tarjeta SD aún se encuentra en el ciclo de inicialización, y entregando un valor de "0" significará que se ha completado la inicialización.

Habiéndose completado la secuencia de inicialización de manera correcta, el *host* deberá obtener la información del campo CCS:

- **CCS = 1** significa que la tarjeta recién inicializada es de Alta Capacidad (*High Capacity SD Memory Card*)
- **CCS = 0** significa que se trata de una tarjeta de Capacidad Estándar (*Standard Capacity SD Memory Card*)

En la tabla 2.3 se presenta la definición del registro OCR de la tarjeta de memoria SD, que es enviado como respuesta al comando CMD58. Con la información proporcionada por este registro conoceremos las siguientes condiciones: el margen de voltaje soportado por la tarjeta, si la tarjeta es o no de alta capacidad y, a través del bit *Card Power up Status*, se podrá saber si la tarjeta ha terminado el ciclo de encendido (*power up cicle*).

Posición de Bits	Definición del campo del OCR
0 – 6	Reservado
7	Reservado para margen de bajo voltaje
8 – 14	Reservado
15	2.7 – 2.8
16	2.8 – 2.9
17	2.9 – 3.0
18	3.0 – 3.1
19	3.1 – 3.2
20	3.2 – 3.3
21	3.3 – 3.4
22	3.4 – 3.5
23	3.5 – 3.6
24- 29	Reservado
30	**CCS** (*Card Capacity Status*)
31	***Card power up status bit*** *(busy)*

Tabla 2.3. Definición del registro OCR.

Operaciones de Lectura

En el modo SPI se soportan operaciones de lectura. Tanto de un solo bloque como de múltiples bloques, a través de los comandos CMD17 (*Lectura de bloque individual*) y CMD18 (*Lectura múltiple de bloques*). Después de la recepción de un comando de lectura válido, la tarjeta enviará la respuesta correspondiente seguida por el bloque de datos; cada bloque de datos será de 512 bytes. En la figura 2.7 se muestra el diagrama de la operación de lectura, las líneas mostradas son de la tarjeta SD.

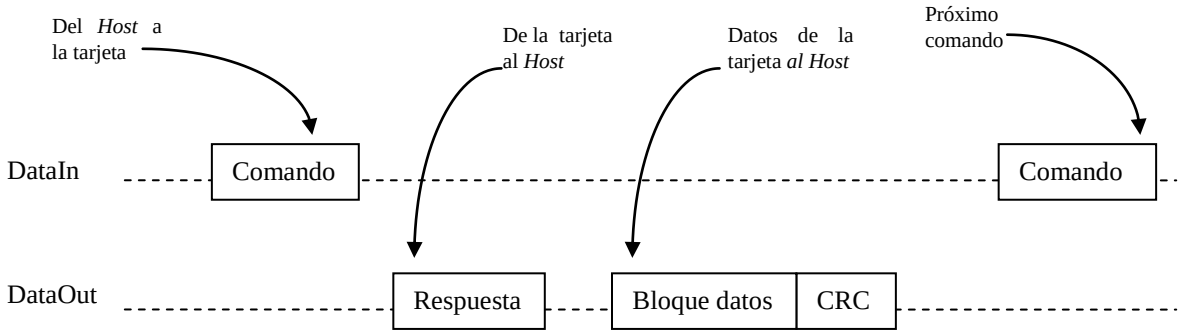


Figura 2.7. Diagrama de la operación de lectura de un bloque de datos.

Cada bloque de datos enviado por la tarjeta viene acompañado por un bloque de bits, que representa un código de detección de errores del tipo CRC (*Ciclic Redundancy Code*) de

16 bits. En el modo SPI, en general, los bits del CRC son ignorados, sin embargo, el primer comando que se envía a la tarjeta debe respetar el CRC, debido a que la tarjeta siempre inicia en el modo SD para el cual el CRC no puede ser ignorado.

En el caso de la lectura de múltiples bloques, el envío de éstos por parte de la tarjeta estará acompañado por los bits de CRC. De existir error de lectura, la tarjeta enviará al *host*, en lugar de datos, la palabra de error y esperará el envío de un nuevo comando.

La operación de lectura de varios bloques de datos, como se muestra en la figura 2.8, se lleva a cabo bajo el mismo esquema que para la lectura de un solo bloque, sólo que en este caso la tarjeta enviará los bloques de manera continua hasta recibir el comando de *fin de transmisión* (CMD12*)* que, una vez recibido, cesará de enviar bloques y terminará la operación enviando la respuesta correspondiente.

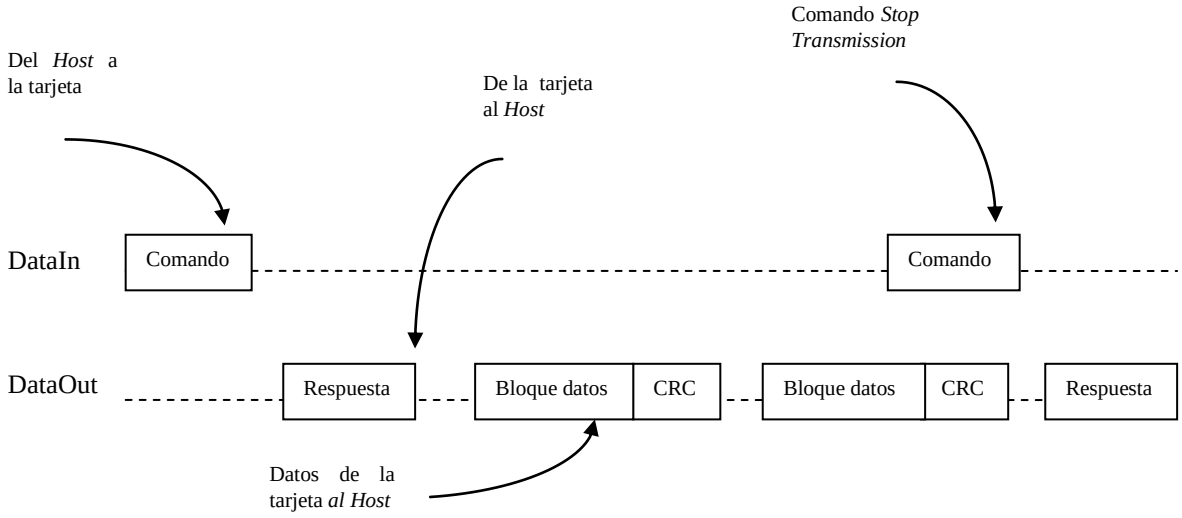


Figura 2.8. Diagrama de la operación de lectura múltiple de bloques de datos.

Operaciones de Escritura

En el modo SPI también se pueden realizar operaciones de escritura de un bloque o de varios, según sea la necesidad. Esto se logra a través de los comandos definidos para estas transacciones: CMD24 (*Escribir bloque*) y CMD25 (*Escritura múltiple de bloques*).

El esquema a seguir para las operaciones de escritura de un bloque es el siguiente: después de la recepción de un comando válido de escritura, la tarjeta enviará la respuesta correspondiente y esperará el bloque de datos que enviará el *host*. Para cada bloque de datos, el *host* debe enviar el byte de inicio de bloque (*Start Block Token*). Una vez recibido el bloque de datos, la tarjeta enviará la *respuesta de datos* (*Data_Response*). Mientras es programando el bloque en la tarjeta se enviará la señal *busy*. Si los datos son enviados sin errores serán programados en la memoria *Flash.* En la figura 2.9 puede observase el esquema definido para la operación de escritura de un bloque de datos.

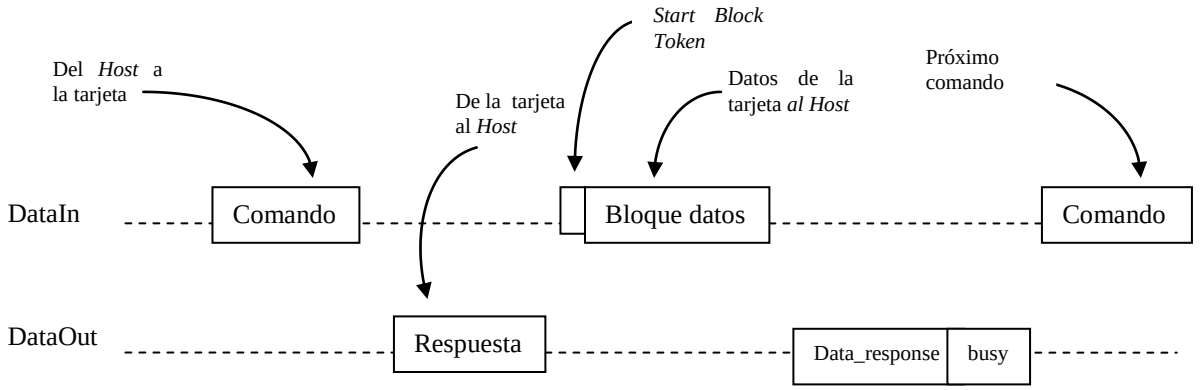


Figura 2.9. Diagrama de la operación de escritura de un bloque de datos.

En cuanto a la escritura de varios archivos, el fin de la transmisión se llevará a cabo enviando el byte de fin de transmisión, llamado "*Stop Tran*", al final del último bloque transmitido. En la figura 2.10 podemos observar el esquema de la escritura múltiple de bloques y el fin de transmisión, como en el caso anterior, la tarjeta envía la señal *busy* mientras graba los datos.

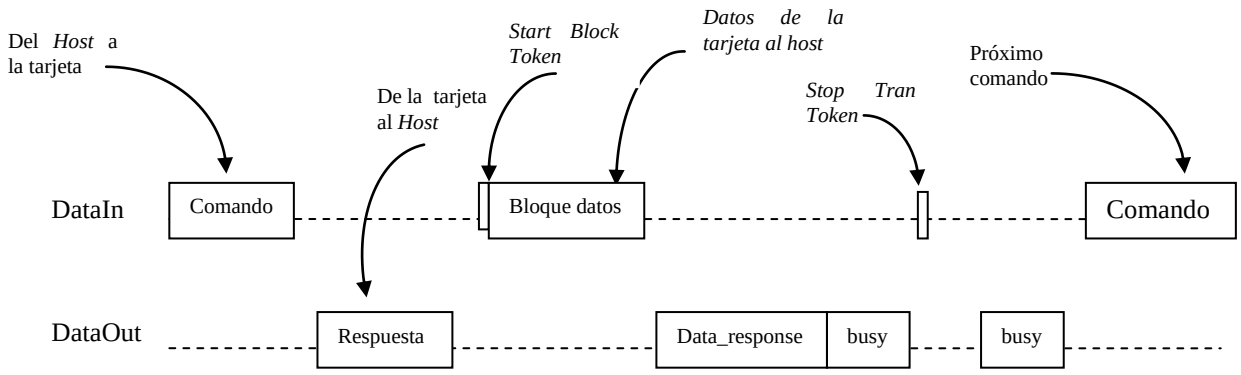


Figura 2.10. Diagrama de la operación de escritura de varios bloques.

Como ya se mencionó anteriormente, en el protocolo SPI la tarjeta responderá siempre a cualquier comando recibido, en la respuesta indicará la aceptación o rechazo del comando, teniendo como causas de rechazo las siguientes:

- Si es enviado mientras la tarjeta se encuentra en una operación de lectura
- Si la tarjeta se encuentra en estado *Busy*
- Si se trata de una tarjeta que no es SD
- Si existe error de CRC
- Si el argumento del comando es ilegal

Se debe evitar que las condiciones mencionadas no ocurran, debido a que pueden provocar la pérdida de información.

Formato de comandos

En el modo SPI, todos los comandos de la tarjeta de memoria SD son conformados por un bloque de 6 bytes, de ocho bits cada uno, como puede observarse en la tabla 2.4. La transmisión de los comandos se realiza enviando primero el bit más significativo (*MSB*) de la cadena correspondiente.

Posición del bit	47	46	[45:40]	[39:8]	[7:1]	0
Cantidad de bits	1	1	6	32	7	1
Valor	'0'	'1'	x	x	x	'1'
Descripción	Bit de inicio	Bit de transmisión	Índice del comando	Argumento	CRC	Bit de fin

Tabla 2.4. Formato de comandos en el modo SPI.

Formato de respuestas

Se especifican varios tipos de respuesta y, al igual que en los comandos, se transmite primero el MSB. Las características principales y el formato de las respuestas son:

Respuesta *R1*

La respuesta **R1** es enviada por la tarjeta después de recibir cada comando, con excepción del comando CMD13 (*SEND_STATUS*). Es de un byte de longitud y su MSB siempre tiene valor cero, el resto de los bits indican condiciones de error, manifiesta al presentarse un "1" en el bit correspondiente. El formato de esta respuesta se muestra en la figura 2.11.

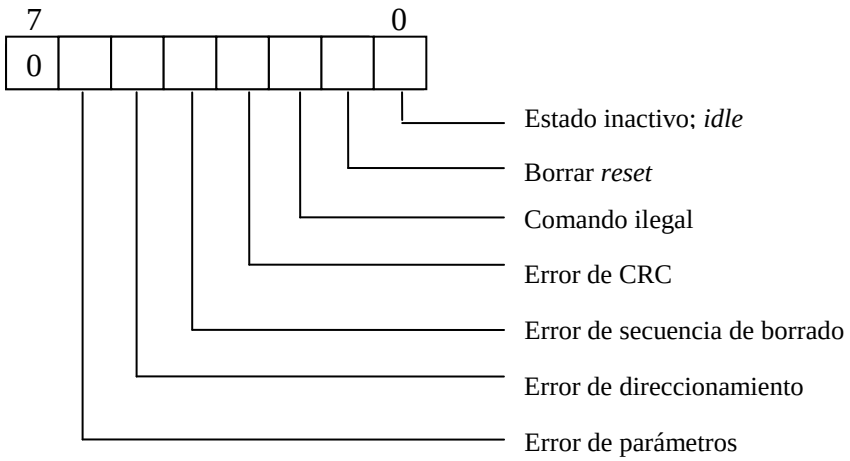


Figura 2.11. Formato de la respuesta R1.

El significado de cada bit se enuncia a continuación:

- **Estado inactivo o *idle***: la tarjeta se encuentra en estado inactivo o corriendo la secuencia de inicialización.
- **Borrar *reset***: la secuencia de borrado fue cancelada debido a la recepción de un comando de reset.
- **Comando ilegal**: se detectó un comando no válido.
- **Error de CRC**: la verificación de los bits de CRC del último comando ha fallado.
- **Error de secuencia de borrado**: ha ocurrido un error en la secuencia de borrado.
- **Error de direccionamiento**: la dirección especificada no está alineada conforme al tamaño de los bloques.
- **Error de parámetro**: El argumento del comando está fuera del margen permitido para la tarjeta actual.

Respuesta *R2*

La longitud de esta respuesta es de dos bytes y es enviada como respuesta al comando *SEND_STATUS* (CMD13), el formato para ésta es el mostrado en la figura 2.12.

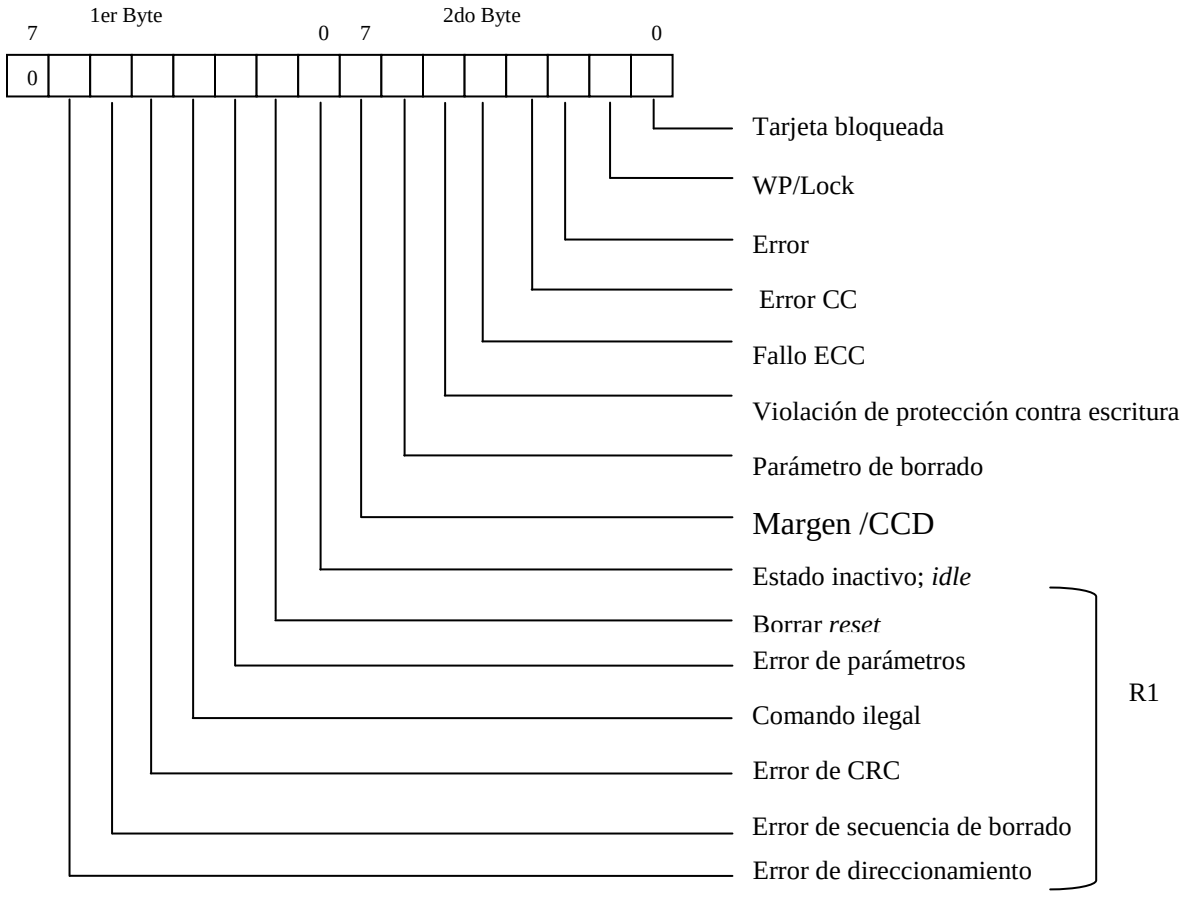


Figura 2.12. Formato de la respuesta R2.

El primer byte de esta respuesta corresponde a la respuesta R1. Los demás bits tienen el siguiente significado:

- **Parámetro de borrado**: la selección sectores o grupos de borrado es inválida.
- **Violación de protección de lectura**: se trató de escribir en un bloque protegido contra escritura.
- **Fallo ECC**: El Código de Corrección de Errores (*Error Correction Code*) fue aplicado pero falló en la corrección de datos.
- **Error CC**: Error interno del controlador de la tarjeta (*Card Controller*).
- **Error**: Un error general o desconocido ocurrió durante la operación.
- **WP/Lock**: Este bit de estado tiene dos funciones; está activo cuando el *host* intenta borrar un sector protegido contra escritura; o si se ejecuta una secuencia incorrecta u ocurren errores en la contraseña durante la operación de bloqueo/desbloqueo de la tarjeta.
- **Tarjeta bloqueada**: Este bit está activo cuando el usuario ha bloqueado la tarjeta, protección contra escritura.

Respuesta *R3*

Esta respuesta es enviada por la tarjeta cuando recibe el comando de lectura del OCR (*CMD58*), tiene una longitud de 5 bytes, como se muestra en la figura 2.13. La estructura del primer byte (MSB) corresponde a la respuesta *R1*. Los siguientes cuatro bytes contienen el registro OCR (ver tabla 2.3).

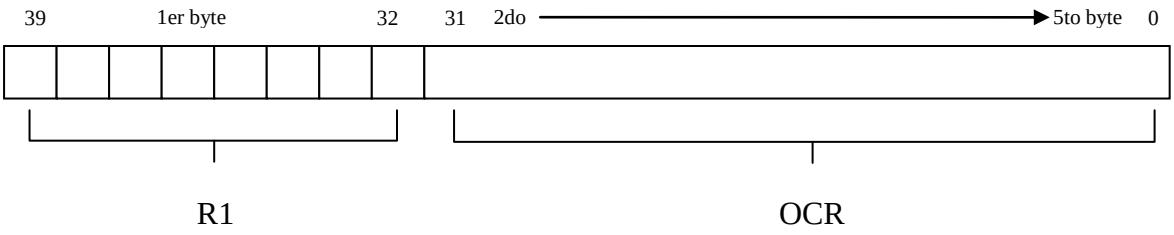


Figura 2.13. Formato de la respuesta R3.

Respuesta R7

Si la tarjeta ha recibido con éxito el comando CMD8 (*Enviar condiciones de interfaz*), ésta enviará el esquema correspondiente a la respuesta R7. El formato de esta respuesta es de 5 bytes, en donde el primer byte (MSB) tiene una estructura que corresponde a la respuesta R1. En los bytes restantes la tarjeta envía la información del voltaje de operación de la tarjeta y

regresa el *patrón de verificación* que le envió el *host* en el argumento del comando correspondiente. En la figura 2.14 podemos observar el formato de la respuesta R7.

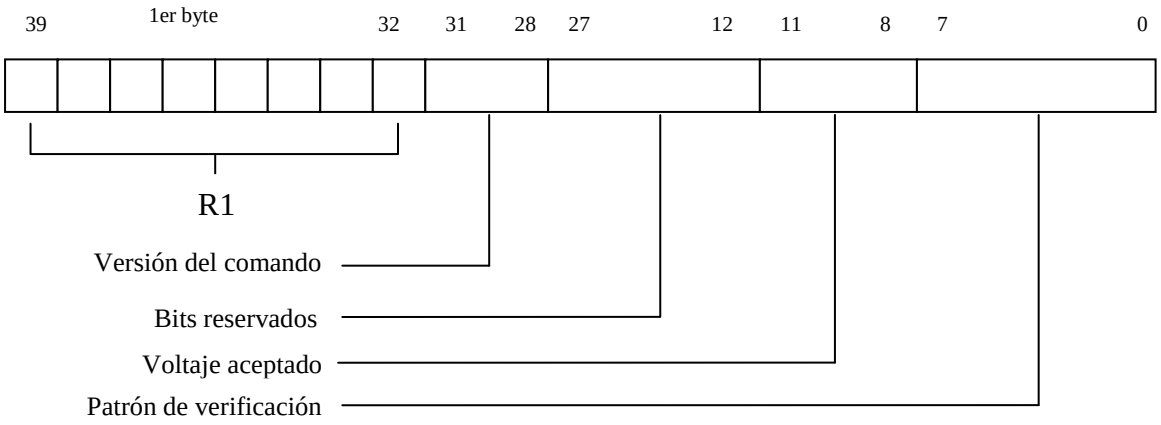


Figura 2.14. Formato de la respuesta R7.

Adicionalmente se definen otras señales para las transacciones entre las tarjetas SD y el *host*, mismas que se describen a continuación.

Respuesta *Respuesta_a_Datos.*

Cada bloque escrito en la tarjeta enviará, a manera de acuse de recibo, la señal *Respuesta_a_Datos*, la cual consta de 1 byte, cuyo formato es presentado en la figura 2.15:

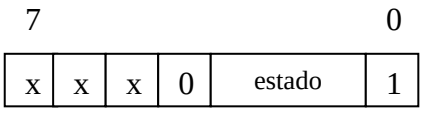


Figura 2.15. Formato de la Respuesta_a_Datos.

El significado de los bits de estado está definido de la siguiente manera:

‘010’ – los datos fueron aceptados.
‘101’ – los datos fueron rechazados debido a un error de CRC.
‘110’ – los datos fueron rechazados debido a un error de escritura.

A través de esta señal, el *host* podrá determinar la posible causa del error, en caso de que éste ocurra.

Señales *Inicio de Bloque* y *Finalizar Transmisión*

En las operaciones de lectura y en las de escritura se tienen comandos que están asociados a ciertas señales; los datos son transmitidos o recibidos por la tarjeta a través de las llamadas **señales de datos**. Éstas tienen una longitud que va desde los 4 hasta los 515 bytes, teniendo el formato presentado en la figura 2.16.

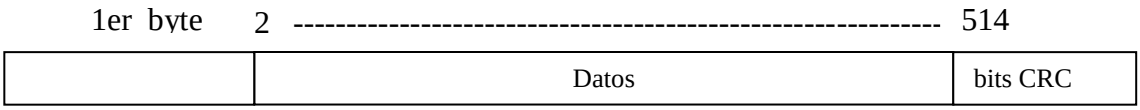


Figura 2.16. Formato de los bloques de datos.

Para operaciones de lectura y escritura de un bloque y lectura de varios bloques:

- Primer byte: El Bloque de Inicio (*start_block token*) tiene el formato mostrado en la figura 2.17.

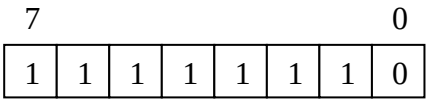


Figura 2.17. Bloque de inicio para lectura múltiple y para lectura y escritura individual.

- Los bytes 2 a 513 corresponden a los datos (información que se lee o escribe).
- Los dos últimos bytes son los bits de CRC (ignorados por el *host* en el modo SPI).

Para operaciones de escritura múltiple de bloques:

- El primer byte de cada bloque:
  - Si los datos han de transmitirse, se envía la señal de inicio de bloque (*start_block token*), mostrada en la figura 2.18:

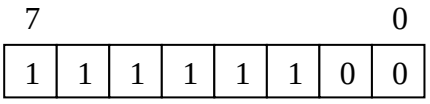


Figura 2.18. Bloque de inicio para escritura múltiple.

- Si el fin de la transmisión es requerido, se envía la señal *Stop_Tran_token*, que se puede observar en la figura 2.19 :

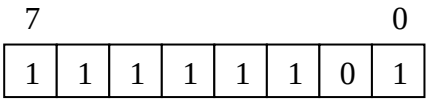


Figura 2.19. Bloque para finalizar la operación de escritura múltiple (Stop_Tran_token).

Este último bloque es enviado por el *host* cuando se requiere terminar con la operación de escritura de varios bloques de datos. En el caso de que la operación sea de lectura múltiple, para finalizar dicha operación, se necesita enviar el comando CMD12 (*Detener transmisión*), que cumple con este propósito.

Señal de error de datos

Si durante la operación de lectura se llegase a presentar una falla y la tarjeta no puede obtener los datos solicitados, ésta enviará una señal de error de datos en sustitución de la información no obtenida. Esta señal está compuesta por un bloque de un byte de longitud, en donde los últimos 4 bits (LSB) corresponden a los bits de error de la respuesta R2. En la figura 2.20 se puede observar el detalle de esta señal.

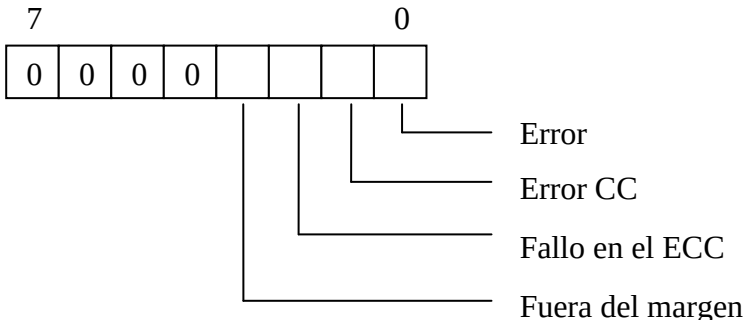


Figura 2.20. Bloque 'Señal de error de datos'.

2.3. Sistemas de archivos

Las tarjetas de memoria flash y demás dispositivos de almacenamiento masivo requieren que los datos que guardan estén organizados, para poder encontrarlos y acceder a ellos de manera eficiente. Como se ejemplifica en la figura 2.21, en una computadora personal el Sistema Operativo (SO) tiene como una de sus funciones principales controlar el acceso a los archivos. Para esto, debe conocer la naturaleza del dispositivo que alberga la información (*hardware* de almacenamiento) y el sistema de archivos con el que cumplen los datos registrados. El SO sirve como interfaz entre las aplicaciones utilizadas por el usuario y el *hardware* del equipo de cómputo.

Figura 2.21. Interacción del Sistema Operativo con Hardware y Software.

Se puede pensar un sistema de archivos como una base de datos de propósito específico que manipula, organiza y le da una estructura a los datos almacenados, para que puedan ser leídos posteriormente por equipos que conozcan las reglas con las que se guardó la información, es decir, que soporten el sistema de archivos utilizado.

Se han desarrollado muchos sistemas de archivos, la mayoría de los sistemas operativos utiliza su propio sistema de archivos y permiten compatibilidad con otros además del propio. Algunos ejemplos de sistemas operativos y los sistemas de archivos soportados se presentan en la tabla 2.5. De los sistemas operativos presentados en dicha tabla, y que podemos encontrar en la gran mayoría de las computadoras personales actuales, *Windows* de *Microsoft* es de los más utilizados a nivel comercial. Este SO utiliza el sistema de archivos FAT (**File Allocation Table**) para el manejo de la información almacenada o por almacenar.

Sistema de archivos	Sistema Operativo
Ext, Ext2, Ext3, Ext4: Extended file system	Linux
XFS: X file system	Linux
HFS, HFS+: Hierarquical File System	Mac OS
HPFS: High Performance File System	OS/2
FAT12, FAT16, FAT32: File Allocation Table	Windows
NTFS: New Technology File System	Windows

Tabla 2.5. Sistemas operativos y los sistemas de archivos soportados.

2.3.1. Sistema de archivos FAT

El sistema de archivos FAT tiene sus orígenes entre finales de 1970 y principios de 1980, y fue el sistema de archivos soportado por el sistema operativo *Microsoft MS-DOS*. Originalmente fue desarrollado como un sistema de archivos simple, adecuado para las unidades de disco flexible con capacidad menor a 500 KB. Con el tiempo fue mejorado para soportar volúmenes de mayor capacidad.

Existen tres tipos de sistemas de archivos FAT: FAT12, FAT16 y FAT32. La diferencia básica en estos subtipos de FAT y el motivo de los nombres, es el tamaño en bits de las entradas en la región FAT en el disco; hay 12 bits en una entrada FAT en FAT12, 16 bits en FAT16 y 32 bits en FAT32. Los sistemas de archivos FAT de mayor uso en la actualidad son FAT16 y FAT32; en el presente trabajo únicamente se hablará de los sistemas FAT16 y FAT32.

Un aspecto importante que se debe considerar al implementar este sistema de archivos, es que, el espacio para datos de una unidad de almacenamiento se divide en pequeñas unidades de 512 bytes (tamaño mínimo), llamadas sectores, esto es a nivel físico. Sin embargo, a nivel lógico el sistema de archivos divide el medio de almacenamiento en unidades llamadas *clúster*; un *clúster* agrupa una cierta cantidad de sectores, determinada de forma automática por el SO. Para el sistema de archivos el clúster es la unidad mínima de almacenamiento que se puede asignar a un archivo y direccionar en la tabla FAT.

El sistema de archivos FAT está compuesto por cuatro regiones, que se encuentran en el siguiente orden dentro del medio de almacenamiento:

0. **Región reservada.**
1. **Región FAT.**
2. **Región del directorio raíz (no existe para FAT32).**
3. **Región de datos, directorios y archivos.**

En la figura 2.22 se puede observar la disposición de las regiones que componen a la estructura de un volumen con sistema de archivos FAT: FAT16 y FAT32.

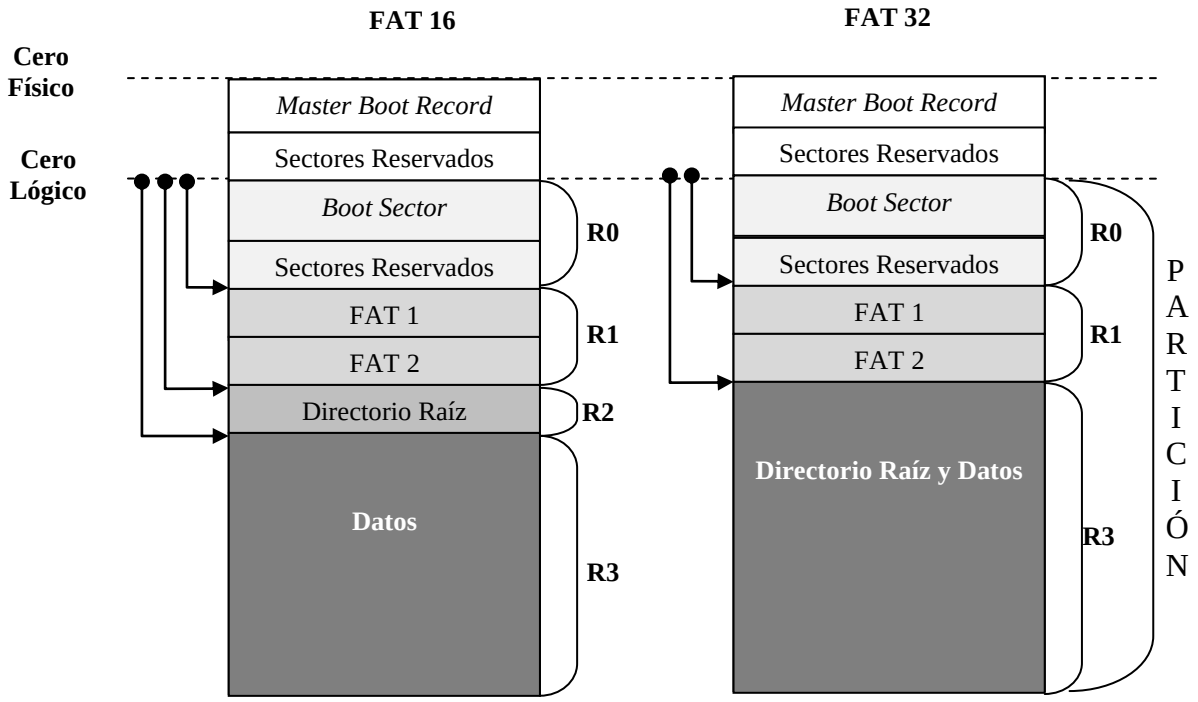


Figura 2.22. Estructura de los sistemas de archivos FAT: FAT16 y FAT32.

El primer sector físico de la unidad de almacenamiento es el llamado Sector Maestro de Arranque o MBR (*Master Boot Record*). En el caso de una PC los primeros 446 bytes del MBR contienen código que sirve para *arrancarla*. Después se encuentran 64 bytes de una tabla de particiones, que es una serie de 4 registros de 16 bytes cada uno, conteniendo información sobre la división lógica. Después del MBR se encuentra la primera partición, que contendrá la estructura del sistema de archivos.

El primer sector en la región reservada (***R0***) es el llamado *Boot Sector* o sector de arranque, en éste se encuentra una estructura llamada Bloque de Parámetros del BIOS (*Basic Input –Output System*) o BPB (*BIOS Parameter Block*). El BPB contiene la información sobre el sistema de archivos, que nos servirá para tener acceso al medio de almacenamiento o volumen, es decir, para poder leer y escribir en él.

En la tabla 2.6, se muestran los campos más importantes contenidos en el *Boot Sector*, se señala un offset que permite localizar al campo en cuestión. Dicho offset está contado a partir del cero lógico, donde empieza la partición. Cabe señalar que en dicha tabla se muestran exclusivamente los campos utilizados en este trabajo, por lo que el offset mostrado no es continuo.

Nombre del campo	**Offset (en bytes)**	**Tamaño (en bytes)**
Bytes por sector (***BytsPerSec***)	11	2
Sectores por clúster (***SecsPerClus***)	13	1
Sectores reservados (***ResvdSecCnt***)	14	2
Número de tablas FAT (***NumFats***)	16	1
Entradas en el directorio raíz en FAT16 (***RootEntCnt***)	17	2
Total de sectores en el volumen en FAT16 (***TotSecs16***)	19	2
Sectores por FAT en FAT16 (***FATSz16***)	22	2
Total de sectores en el volumen en FAT32 (***TotSecs32***)	32	4
Sectores por FAT en FAT32 (***FATSz32***)	36	4
Número del primer clúster del directorio raíz (***RootClus***)	44	4

Tabla 2.6. Campos importantes del sector de arranque o Boot Sector.

La región FAT (***R1***) almacena la llamada Tabla de Asignación de Archivos, que está definida como una lista ligada de los clústers que componen a los distintos archivos y sirve como un mapa de la región de datos de la partición, de acuerdo al número de clúster. Cada clúster es identificado por un número con el que se crea una entrada en la región FAT. Las entradas en la región FAT se crean en una posición correspondiente al número de clúster; el contenido de estas entradas indican el número del siguiente clúster de un archivo dado y con una marca especial el fin de dicho archivo. Por lo general, en la región FAT existen dos tablas FAT, siendo la segunda un duplicado exacto de la primera, sirviendo como copia de seguridad.

En la figura 2.23 se muestra un ejemplo del seguimiento de archivos en una tabla *FAT*. En dicha figura, se presenta el aspecto que tendría la región FAT de un volumen que contiene a tres archivos y al directorio raíz. Los clústers que componen a cada archivo se en listan en los recuadros de la derecha. El Directorio Raíz ocupa los clústers 0x0002, 0x0009, 0x000A y 0x0011, por lo que las entradas correspondientes se encuentran en las posiciones 0x0002, 0x0009, 0x000A y 0x0011 respectivamente. Puede notarse que el contenido de la entrada localizada en la posición 0x0002 es 0x0009, es decir que indica el número del siguiente clúster. Esto aplica para el resto de las entradas, excepto en la última, en la que se indica con el valor 0xFFFFFFFF el fin del archivo. Esta es la lógica que se sigue para localizar a cualquier archivo.

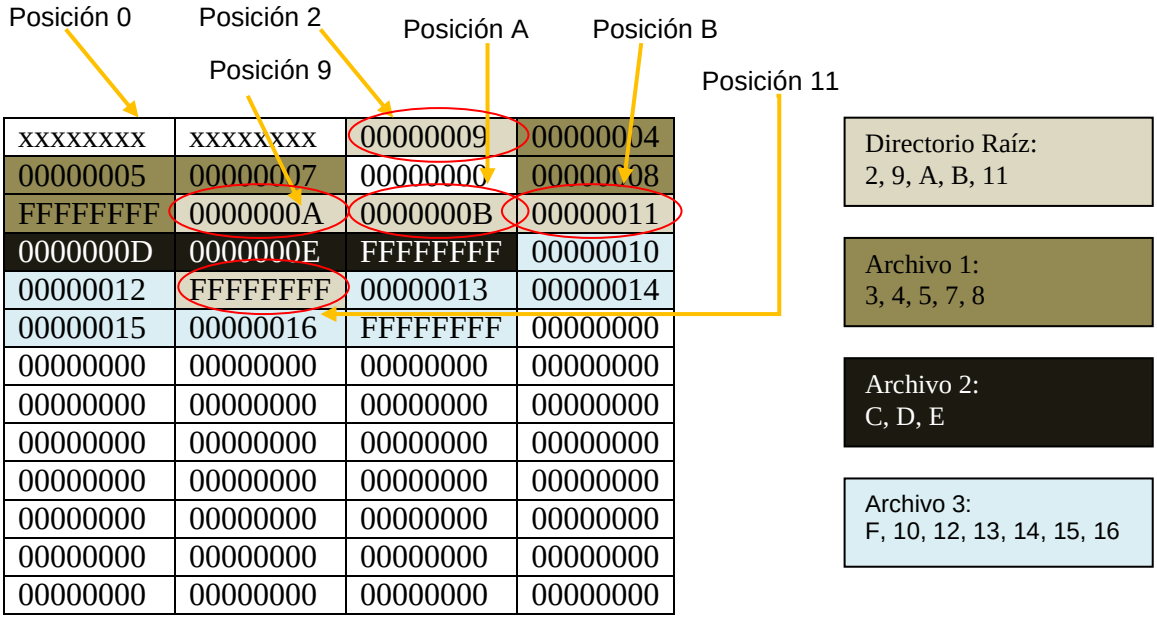


Figura 2.23. Seguimiento de los archivos en una tabla FAT.

La Región del Directorio raíz (***R2***) fue definida por *Microsoft* sólo en el sistema de archivos FAT16, en la cual se tiene una sección determinada y de tamaño fijo para dicho fin. Para sistemas con FAT32 el Directorio Raíz es de tamaño variable, dependiendo de la cantidad de archivos y directorios que se vayan creando. Por lo general se encuentra en el primer clúster del área de datos.

El Directorio Raíz guarda la información relativa a la disposición de los datos contenidos en el medio de almacenamiento; los datos son almacenados en una estructura de archivos y carpetas, en donde un directorio es un archivo con el atributo de directorio y su contenido serán las entradas de los archivos que lo conforman. El único directorio especial y que siempre debe estar presente es el directorio raíz.

Cada archivo o carpeta creada tiene una entrada de 32 bytes en el Directorio Raíz (ver fig. 2.24); se utilizan ocho bytes para el nombre del archivo, tres para la extensión, un byte de atributos, ocho bytes reservados, dos bytes para la parte alta del número del primer clúster, cuatro bytes reservados, dos bytes para la parte baja del número del primer clúster y cuatro bytes para el tamaño en bytes. Esta información es utilizada por el SO para localizar al archivo deseado; obteniendo el número del primer clúster de un archivo encontrará el resto, entrando a la tabla FAT. Las entradas son iguales para los distintos tipos de FAT.

En la figura 2.22 se muestra un ejemplo de la organización de cualquier entrada en el directorio raíz, en este ejemplo corresponde a un archivo de nombre ARCHIVO1, con extensión TXT, el atributo indica que se trata de un archivo, el primer clúster es el número 0x0000 0003 y el tamaño de este archivo es 512 bytes (0x0200).

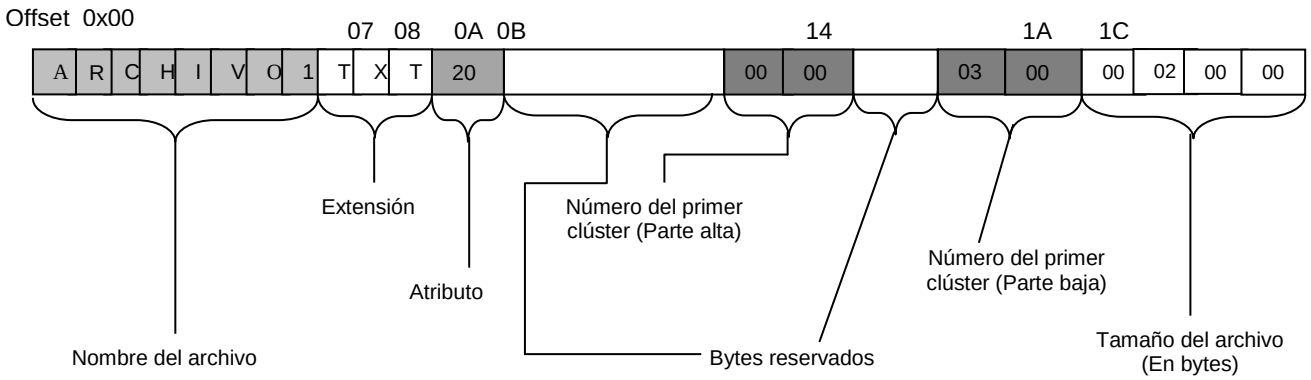


Figura 2.24. Estructura de las entradas en el Directorio Raíz.

La última región corresponde al área de datos (***R3***), en la que son registrados los archivos y carpetas del usuario. Por lo general, la región de datos se encuentra a partir del clúster 2.

Determinación del tipo de FAT

El dispositivo que intente leer un medio de almacenamiento y localizar los archivos y carpetas contenidos, debe conocer el tipo de sistema de archivos que tiene dicho medio, para realizar las operaciones de lectura y escritura de manera correcta.

En el caso de los sistemas de archivos FAT, la única manera como se puede conocer el tipo de FAT que se tiene es determinando el parámetro *Count_Of_Clusters*, que se refiere a la cantidad de clústers que tiene la unidad de almacenamiento en la región de datos. El procedimiento es el siguiente:

Primero, se determina la cantidad de sectores ocupados por el directorio raíz, realizando la siguiente operación, con algunos de los campos del BPB mostrados en la tabla 2.6:

$$RootDirSectors = ((RootEntCnt * 32) + (BytsPerSec - 1))/BytesPerSec$$

En un sistema con FAT32 el parámetro *RootDirSectors* es siempre 0, porque se ha definido al campo *RootEntCnt* como cero.

El siguiente paso consiste en determinar la cantidad de sectores en la región de datos. Teniendo en cuenta las siguientes consideraciones:

Si el sistema es FAT16: $FATsz = FATSz16$
$TotSec = TotSec16$

Si el sistema es FAT32: $FATsz = FATSz32$
$TotSec = TotSec32$

A partir de lo anterior se realiza la siguiente operación:

$$DataSec = TotSec - (ResvdSecCnt + (NumFATs * FATSz) + RootDirSectors)$$

Por último, se obtiene el valor de de *CountOfClusters* de la siguiente manera:

$$CountOfClusters = DataSec \,/\, SecPerClus$$

Una vez determinado este valor se puede conocer qué tipo de FAT tiene el sistema, tomando en cuenta que siempre se cumple lo siguiente:

- En FAT12 $\boldsymbol{CountOfClusters} < 4{,}084$
- En FAT16 $\boldsymbol{CountOfClusters} \leq 65{,}524$
- En FAT32 $\boldsymbol{CountOfClusters} > 65{,}524$

2.4. Unidad sísmica SR04

La unidad sísmica SR04 es un instrumento diseñado para convertir una señal sísmica analógica a su forma digital y grabarla haciendo uso de una computadora. Esta unidad es producida por la compañía italiana *SARA electronic instruments*. Fue pensada, originalmente, para trabajar con un equipo de cómputo en el cual se tenga instalado algún *software* de registro de datos. Esta unidad está formada por dos tarjetas electrónicas (SADC20 y GPSDCF) y tres sensores sísmicos (geófonos), integrados dentro de un gabinete de uso industrial, ver fig. 2.25.

Figura 2.25. Unidad sísmica SR04.

Los sensores son del tipo geófono modelo GS-11D, que permiten medir las tres componentes de movimiento (x,y,z). Un geófono es un sensor de tipo electromagnético que tiene la forma de un cilindro metálico y que contiene un imán y una bobina sostenida por un

resorte. Este sensor entrega una medición proporcional a la velocidad del movimiento de la tierra. En la figura 2.26 se muestra la estructura y la apariencia de estos sensores.

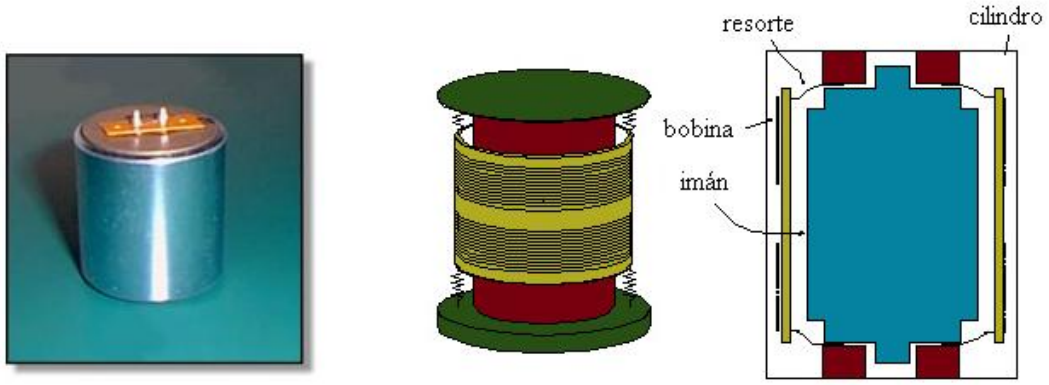


Figura 2.26. Geófono GS – 11D.

2.4.1. Tarjeta digitalizadora SADC20

La SADC20 es una tarjeta de conversión analógica – digital de 24 bits para tres canales, diseñada para ofrecer altas prestaciones en aplicaciones sísmicas; acepta señales provenientes de un amplio catálogo de sensores, aunque ha sido optimizada para trabajar directamente con geófonos, de manera que no necesita amplificadores externos. A continuación se describen las partes que componen a la tarjeta, ver figura 2.27:

1) Una entrada para la recepción de la referencia de tiempo, proveniente de la tarjeta GPSDCF.

2) Tres canales analógicos de entrada. Cada canal puede ser configurado para aceptar señales diferenciales unipolares o señales diferenciales bipolares.

3) Un arreglo de filtros RC paso-bajas por canal, cuyas frecuencias de corte se pueden ajustar a 8.8, 19 y 41 Hz, reemplazando los capacitores de entrada.

4) Un conversor sigma-delta de 24 bits por canal, cuya frecuencia de muestreo se puede ajustar a 10, 20, 25, 50, 100 y 200 muestras por segundo.

   Cada conversor acepta un rango de voltaje de ±1 V, por lo que el valor más pequeño detectable es 119 nV, equivalente a una cuenta.

5) Un conector para programación de circuitos (ISP: In System Programming) para poder actualizar el *firmware* del microcontrolador, esta actualización únicamente la puede realizar el fabricante.

6) Un microcontrolador, que está programado para ejecutar las siguientes funciones:

(1) Recibir la señal de tiempo enviada por la tarjeta GPSDCF, para mantener actualizado el reloj en tiempo real.

(2) Recibir los datos enviados por los conversores, estamparlos con su referencia de tiempo y organizarlos de acuerdo al protocolo de comunicación del fabricante.

(3) Establecer la comunicación con la computadora a través del estándar RS232, para el envío de datos y la recepción de comandos de configuración.

7) Una entrada protegida contra inversión de polaridad para la conexión de la alimentación de energía, que acepta de 8 a 12 V.

8) Un conector DB9 para la comunicación RS232.

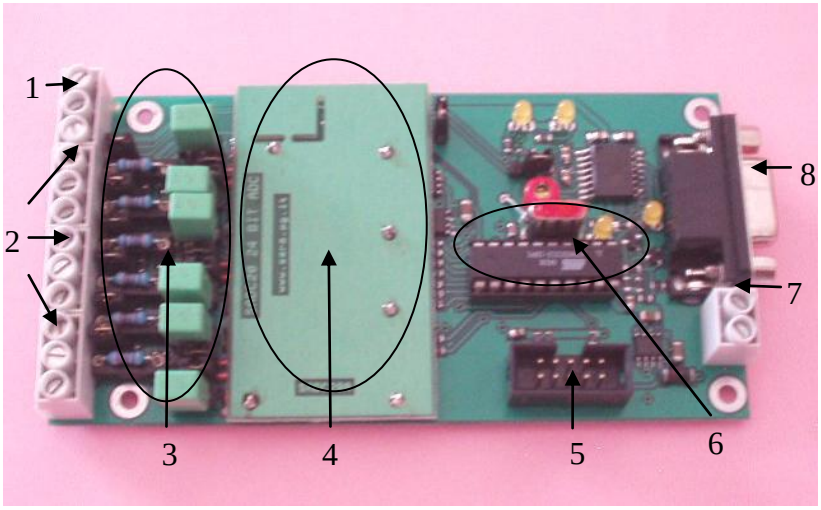


Figura 2.27. Tarjeta SADC20.

2.4.2. La tarjeta GPSDCF

La tarjeta GPSDCF (GPS: Global Positioning System; DCF: **D** significa Alemania (*Deutschland*), **C** significa señal de onda larga, **F** significa Frankfurt y **77** se refiere a la frecuencia de 77.5kHz) convierte las señales de tiempo de un receptor GPS en pulsos codificados de acuerdo con el estándar DCF77.

DCF77 es un transmisor de radio de onda larga situado en Mainfield (aproximadamente 25 km al sureste de Frankfurt, Alemania) que transmite las señales de tiempo generadas por un reloj atómico en una señal portadora de 77.5 kHz con una potencia de 50 kW. La señal de este transmisor puede recibirse en gran parte del territorio europeo abarcando hasta 2000 km de distancia de Frankfurt.

El estándar DCF77 utiliza 60 bits para transmitir los datos de tiempo: del bit 0 al bit 58 se utilizan para enviar cada minuto la fecha y el tiempo completo. El bit 59 se utiliza como marcador para detectar el comienzo de cada minuto. En la figura 2.28 se muestra la forma en que se codifican los bits usados para transmitir el tiempo.

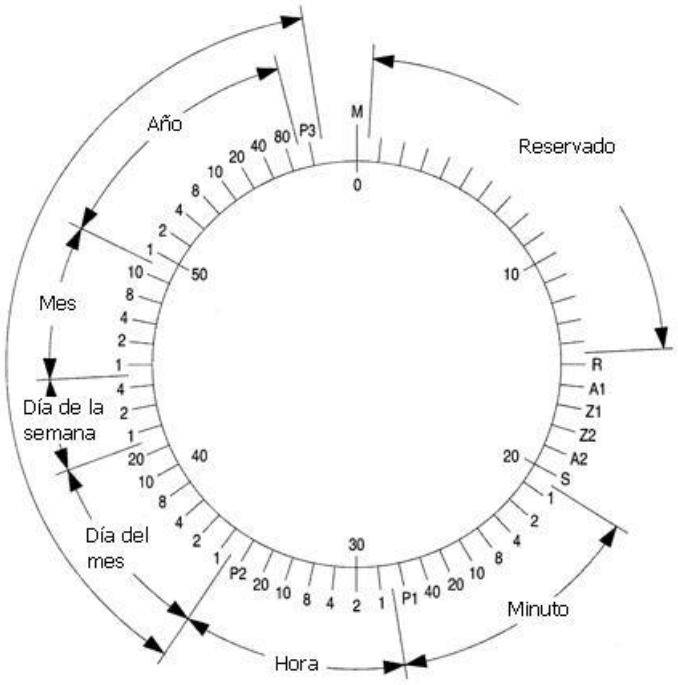


Figura 2.28. Codificación de los datos de tiempo enviados por un transmisor DCF77.

Las partes que en general constituyen la GPSDCF (ver fig. 2.29) son:

1) Una salida para el envío de pulsos en formato DCF77. Esta salida puede manejar como máximo una carga de 5 mA.

2) Un receptor GPS, utilizado para recoger las señales provenientes de una red de satélites y determinar la localización y el tiempo exacto. Este receptor tiene una entrada para la conexión de una antena y un conector de siete terminales hacia la tarjeta. Las siete terminales del conector son: datos del PPS (*Precise Positioning Service*), salida de datos, entrada de datos, tierra, batería de respaldo, fuente de energía de la antena y fuente de energía del receptor (5V).

   El consumo de potencia de la tarjeta cuando el receptor GPS está en operación es de 1W y cuando la tarjeta está en modo de ahorro de energía el consumo es de 300mW.

3) Un conector de tres terminales para la alimentación de una carga externa a 5 V.

4) Un conector ISP para la actualización del *firmware* del microcontrolador, dicha actualización únicamente la puede realizar el fabricante.

5) Una salida de voltaje dependiente del voltaje de alimentación de la tarjeta. Esta salida sirve para alimentar a otro dispositivo y está protegida con un fusible de 1A.

6) Una fuente conmutada de 5V a 2A, para la alimentación del receptor GPS y de alguna carga externa.

7) Una entrada para la alimentación de la tarjeta. El rango de voltajes que acepta la tarjeta es de 8 a 16 V.

8) Otro conector ISP para la conexión de la tarjeta a una computadora o a un receptor GPS externo usando la interfaz RS232.

9) Una batería de respaldo de 5V.

10) Un grupo de *jumpers* para definir el modo de operación de la tarjeta:

J1	1-2= entrada de PPS normal	2-3= entrada de PPS invertida
J2	1-2= salida de PPS normal	2-3= salida de PPS invertida
J3	1-2= receptor GPS interno	2-3= receptor GPS externo

11) Un microcontrolador para efectuar la conversión de las sentencias NMEA al estándar DCF77.

12) Un conjunto de *leds* para indicar el estado de la tarjeta:

D2	Está encendido cuando está habilitada la salida de 5V.
D4	Está encendido cuando el GPS está energizado.
D5	Enciende y apaga cuando se energiza la tarjeta.
D6	Está encendido cuando el receptor GPS está captando información de los satélites.

13) Un dispositivo con cuatro *switches* para configurar las siguientes opciones de la tarjeta:

S1-1	OFF = operación normal	ON = modo de ahorro de energía
S1-2	OFF = PPS modulado DCF77	ON = PPS 50% ciclo de trabajo
S1-3	OFF = señal de PPS de bajada	ON = señal de PPS de subida
S1-4	OFF = operación normal	ON = voltaje de entrada bajo, apaga salida de 5V

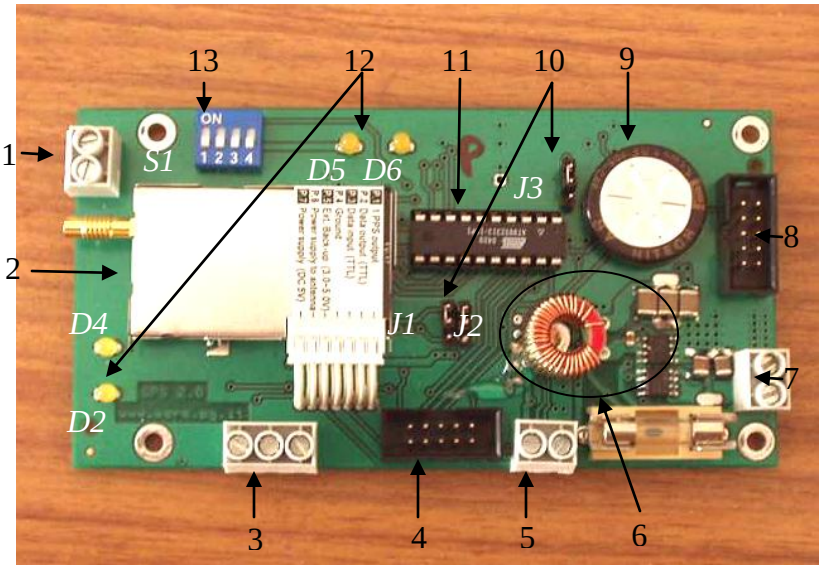


Figura 2.29. Tarjeta GPSDCF.

2.4.3. Protocolo de comunicación de la tarjeta SADC20

La comunicación con la tarjeta SADC20 es bidireccional, se realiza por un puerto serie RS232, que debe ser configurado a 38 400 baudios, sin señal de *handshake*, 8 bits de datos, 1 bit de paro y sin paridad.

Los datos digitalizados por la SADC20 son enviados en paquetes de 5 bytes para cada canal, y los datos correspondientes a la marca de tiempo son enviados en paquetes de 9 bytes. En estos paquetes, el primer byte corresponde a un identificador, cuyo valor es mayor o igual a 128 decimal (0x80 hexadecimal), y sirve para indicar el canal al que corresponden los próximos bytes transmitidos o si se trata del paquete de tiempo. El último byte indica el fin del paquete correspondiente, teniendo un valor mayor a 240 (0xF0). Los datos digitalizados son enviados en el resto del paquete y tienen un valor menor a 128 (0x80 hexadecimal). En la figura 2.30 podemos observar la forma en que son enviados los paquetes de información, suponiendo ***n*** muestras por segundo.

El paquete de tiempo es identificado por el byte 0x81, los siguientes bytes corresponden al año, mes, día, segundo, minuto y hora, en ese orden. La información es enviada en forma binaria, cada segundo. El byte extra es utilizado para enviar información de las líneas auxiliares L1 y L2 provenientes de la tarjeta GPSDCF y que son usadas para detectar la información del GPS. El byte 0xFF indica el fin de la marca de tiempo.

Para los canales los bytes de identificación tienen valores 0x82, 0x83 y 0x84 perteneciendo a los canales 1, 2 y 3, respectivamente. Los tres bytes siguientes (bajo, medio y alto) de cada paquete, contienen el valor de la muestra correspondiente.

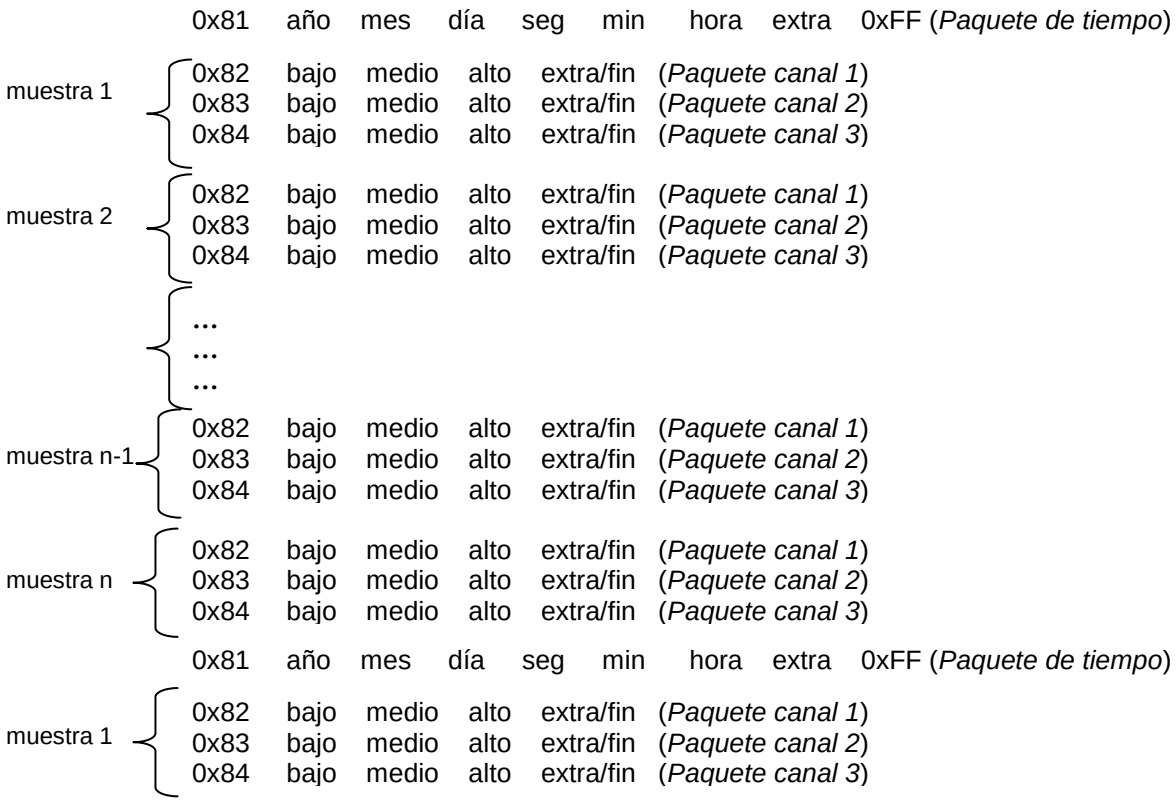


Figura 2.30. Formato de los paquetes de información provenientes de la tarjeta SADC20.

De acuerdo al protocolo del fabricante, los bytes de cada muestra necesitan ser completados porque se transmiten sin el bit número 7. El byte extra/fin contiene la información para completar el bit siete de los bytes bajo, medio y alto. El byte extra/fin tiene el siguiente formato:

Bit0	bit número 7 del byte bajo
Bit1	bit número 7 del byte medio
Bit2	bit número 7 del byte alto
Bit3	siempre 1
Bit4	siempre 1
Bit5	siempre 1
Bit6	siempre 1
Bit7	siempre 1

Decodificación de los datos

Para obtener el valor de cada muestra se debe decodificar cada paquete de información, esto a partir del algoritmo que se presenta a continuación:

Primero se completan los bytes bajo y medio:

$\boldsymbol{low = low + (extra \,\&\, 0x01) * 128;}$ // completa el byte más bajo *low*
$\boldsymbol{middle = middle + (extra \,\&\, 0x02) * 64;}$ // completa el byte *middle*

Calcula el valor absoluto de la muestra:

$\boldsymbol{tmp = (high * 65536) + (middle * 256) + low}$;

Por último, se le agrega el signo al valor obtenido:

```
if (extra & 0x04 )
{
  valor = -8388608 + tmp;
}

Else
valor = tmp;
```

En el código anterior está escrito en C, y se tiene que:

$\boldsymbol{low}$ = byte más bajo de los 24 bits
$\boldsymbol{middle}$ = byte medio de los 24 bits
$\boldsymbol{high}$ = byte más alto de los 24 bits
$\boldsymbol{extra}$ = bits transmitidos en el byte extra/fin
$\boldsymbol{tmp}$ = valor temporal
$\boldsymbol{valor}$ = variable que contiene el valor decodificado

Comandos aplicables a la unidad SR04

La unidad sísmica acepta una serie de comandos que sirven para obtener el contenido de la EEPROM, para establecer parámetros como el tiempo, la fecha, la frecuencia de muestreo, la compensación del reloj y para realizar correcciones al tiempo GMT.

Los comandos están formados por paquetes de 6 bytes, donde el primer byte es el identificador de la operación a realizar y siempre es mayor a 128 en decimal (0x80), mientras que los bytes restantes son los parámetros propios de cada comando. A continuación se presentan los comandos soportados por la unidad SR04:

- Obtención de la versión del firmware

Sintaxis del comando: 0x81 0x00 0x00 0x00 0x00 0x00

Este comando es enviado si se desea conocer la versión del firmware de la tarjeta. Para la SADC20 la versión actual es la 2.00, por lo que al enviar el comando se espera que la respuesta sea V200 (en código ASCII).

- Corrección del tiempo GMT

Sintaxis del comando: 0x82 GMT 0x00 0x00 0x00 0x00

En el envío del comando, el parámetro GMT se debe dar en formato complemento a dos y con valores que van de -23 a +23. Si el comando ha sido recibido correctamente, la tarjeta responde enviando el byte 248 decimal (0xF8).

- Ajuste del tiempo

Sintaxis del comando: 0x83 seg min hora 0x00 0x00

Este comando ajusta el tiempo del reloj de la SADC20. Los parámetros que ajusta son los segundos, minutos y horas, los cuales deben mandarse en binario y no en BDC o ASCII. Si se recibió el comando correctamente, la tarjeta responde enviando un byte con valor decimal de 248 (0xF8).

- Ajuste de la fecha

Sintaxis del comando: 0x87 año mes día 0x00 0x00

El comando de ajuste de fecha permite modificar el año, el mes y el día del reloj de la SADC20. Las consideraciones para el envío de los parámetros de ajuste son las mismas que para el tiempo e igualmente se recibe el byte 248 decimal (0xF8) cuando se tiene éxito en la operación.

- Ajuste de la frecuencia de muestreo

Sintaxis del comando: 0x84 sps1 sps2 sps3 0x00 0x00

Este comando es de los más importantes, permite establecer la frecuencia a la que se muestrearán los sensores conectados a los canales de esta unidad sísmica. Los bytes sps1, sps2 y sps3 del comando corresponden a la especificación de la frecuencia de muestreo que se programará en los canales 1, 2 y 3, respectivamente.

La frecuencia de muestreo se especifica utilizando la siguiente expresión:

$$\boldsymbol{spsX = 200/(frecuencia\ requerida)}$$

De tal forma que si se necesitara una frecuencia de 100 Hz el valor de spsX sería:

$$spX = 200/100 = 2$$

Y el comando que se envía para establecer dicha frecuencia tiene la siguiente forma:

0x84 0x02 0x02 0x02 0x00 0x00

Cabe comentar que en esta versión de la unidad sísmica no es posible programar los canales a diferentes frecuencias y que una vez establecido este parámetro, se mantiene en la memoria EEPROM de la tarjeta. Sin embargo, el fabricante reserva los bits 2 y 3 de este comando para futuras versiones, en las que se soportará el muestreo a distintas frecuencias.

- Compensación del error del cristal

Sintaxis del comando: 0x85 baja media alta dirección 0x00

El error del cristal de la tarjeta ocurre principalmente por la variación de temperatura, lo que puede provocar errores en la frecuencia de muestreo, lo que implica que se tome una cantidad mayor o menor de muestras. Dicho error se puede compensar al aplicar el comando apropiado.

Este comando programa a la tarjeta para que compense 100 segundos cada X cientos de segundos. Si se recibe el comando adecuadamente la tarjeta contesta enviando el byte 248 decimal (0xF8). La compensación debe realizarse antes de realizar la adquisición de datos.

El fabricante indica que en futuras versiones de la unidad sísmica utilizará cristales más precisos, lo que evitará utilizar este comando.

- Lectura de la EEPROM

Sintaxis del comando: 0x86 dirección 0x00 0x00 0x00 0x00

Con este comando se pueden obtener algunos de los parámetros que se han establecido en la tarjeta y que se almacenan en la EEPROM. Los parámetros almacenados son la compensación del cristal, la corrección del tiempo GMT y la frecuencia de muestreo de los canales. La dirección de cada uno de ellos se indica en el siguiente mapa de memoria:

0x00	Corrección del tiempo GMT
0x01	Compensación baja
0x02	Compensación media
0x03	Compensación alta
0x04	Compensación dirección
0x05	Frecuencia de muestreo del canal 1
0x06	Frecuencia de muestreo del canal 2
0x07	Frecuencia de muestreo del canal 3

Si el comando se reconoce correctamente, la tarjeta responde con un solo byte conteniendo el dato encontrado en la dirección especificada de la EEPROM.

2.5. Microcontrolador

Un microprocesador puede entenderse como una Unidad Central de Procesamiento (*CPU: Central Processing Unit*), formada por una Unidad de Aritmética y Lógica (*ALU: Arithmetic Logic Unit*), una unidad de control y algunos registros de transferencia, todo encapsulado en un solo circuito integrado.

Un microcontrolador es un sistema que incluye en un solo circuito integrado a un microprocesador y a un conjunto de subsistemas que le proveen memoria, puertos de entrada y salida, comunicación serie, convertidor analógico – digital, unidad de tiempo, oscilador, temporizador y otros. De manera general, a continuación se describen los dispositivos internos con que cuentan los microcontroladores:

- *Reloj del sistema*: El microcontrolador ejecuta las instrucciones de un programa determinado a cierta velocidad, la cual está determinada por la frecuencia del oscilador (frecuencia del reloj o reloj del sistema), dicha frecuencia puede ser generada por un circuito interno de tipo RC o por circuitos externos del tipo RC, LC o a través de osciladores basados en cristal de cuarzo.

- *Memoria de programa*: La memoria de programa es el espacio destinado para almacenar las instrucciones que constituyen al código del programa. Es de tipo no-volátil, por lo general de tipo *EEPROM* o *FLASH*.

- *Memoria de lectura y escritura*: Es la memoria utilizada por el microcontrolador para el almacenamiento de datos y variables. Es memoria tipo RAM.

- *Módulo MSSP* (*Master Synchronous Serial Port*): Este módulo permite la comunicación a través del bus SPI y también por I^2C.

- *Módulo USART* (*Universal Synchronous Asynchronous Receiver Transmitter)*: El modulo *USART* proporciona al microcontrolador la capacidad para comunicarse a través de las interfaces seriales RS–485 y RS–232.

- *Puerto E/S digital*: El puerto de entrada y salida digital provee terminales que permiten controlar y recibir información de dispositivos periféricos, como teclados, *display's*, relevadores, motores y algunos otros.

- *Puerto de conversión analógico – digital*: Este puerto genera un número binario proporcional al voltaje de entrada detectado. Es utilizado para manipular la señal proveniente de algunos sensores.

- *Temporizador* (*Timmer*): El temporizador es implementado como un contador que establece el tiempo preciso para algún evento o rutina. También es útil como contador de eventos.

2.5.1. Arquitecturas de microcontroladores

Los microcontroladores pueden ser clasificados de acuerdo a las características del conjunto de instrucciones que soportan:

- CISC (Complex Instruction Set Computer): Computadora con un set de instrucciones complejas.
- RISC (Reduced Instruction Set Computer): Computadora con un set de instrucciones reducido.
- MISC (Minimal Instruction Set Computer): Computadora con un set de instrucciones mínimo.

Otra forma en la que son clasificados los microcontroladores es por la arquitectura que siguen, es decir, la manera en que el microcontrolador accede a los recursos con los que dispone. Las arquitecturas utilizadas son las de Princeton o Von Newmann y la de Harvard:

Arquitectura Princeton o Von Newmann

En la arquitectura Von Newmann, la Unidad Central de Procesamiento está conectada a una sola memoria, en la que se tiene tanto la memoria de programa como la memoria de datos. En la figura 2.31, se muestra la arquitectura mencionada.

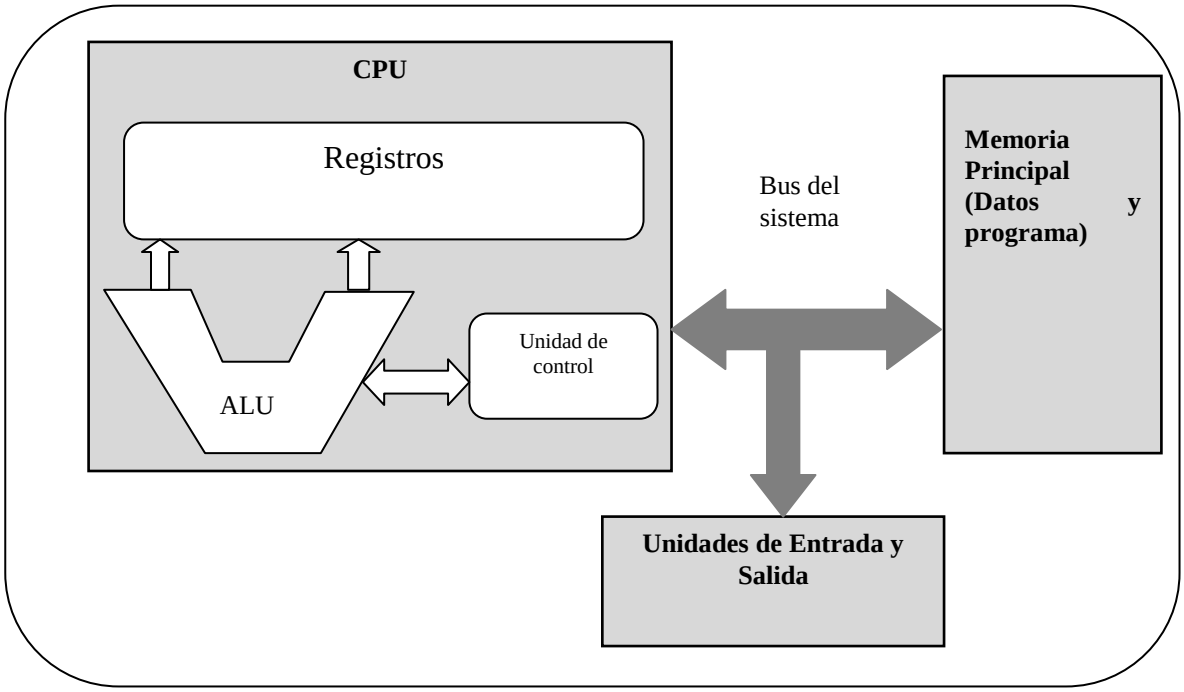


Figura 2.31. Arquitectura Von Newmann.

Arquitectura Harvard

En esta arquitectura se cuenta con dos memorias independientes: memoria de datos y memoria de programa, a las cuales se accede a través de dos grupos de buses separados. Dichos buses son independientes y pueden ser de distinto ancho, lo que permite acceder al mismo tiempo a la memoria de datos y a la memoria de programa, obteniendo un mayor desempeño en la ejecución de instrucciones.

En la figura 2.32 se muestra la estructura de un microcontrolador en la arquitectura Harvard.

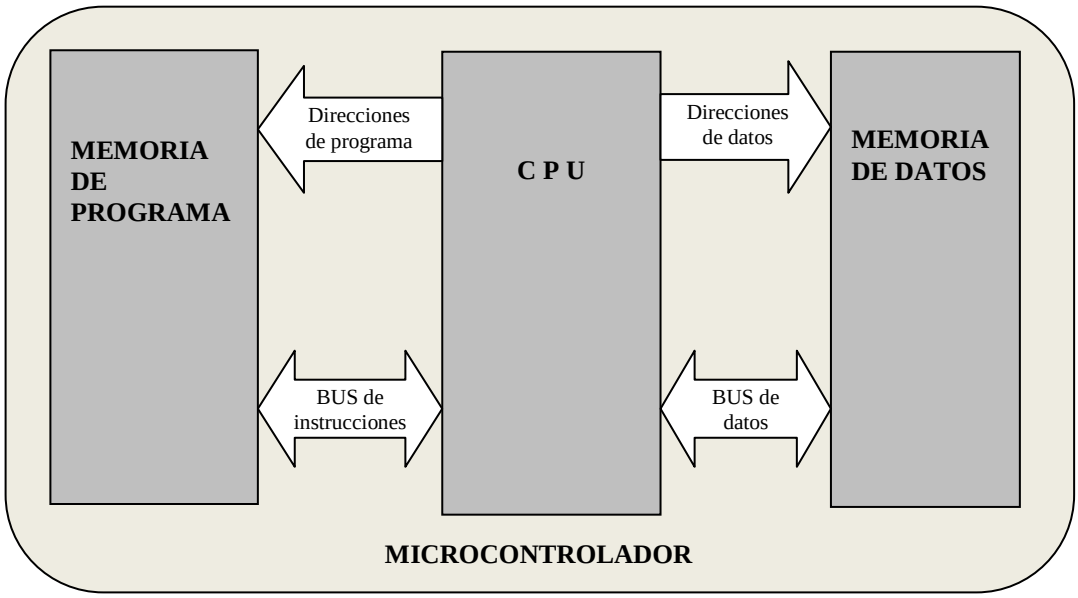


Figura 2.32. Arquitectura Harvard.

2.5.2. Familias de microcontroladores

Dentro del mercado de los microcontroladores existen muchas marcas y modelos, cada fabricante cuenta con dispositivos tanto de propósito general como para uso específico y con características muy diversas. A continuación se comenta en forma general los principales fabricantes de microcontroladores, dando una breve descripción de sus modelos y mencionando las principales familias que producen.

ARM: El núcleo de los microcontroladores ARM es utilizado por distintos fabricantes como base para sus propios diseños de microcontroladores. Los microcontroladores ARM manejan tanto la arquitectura Harvard como la Von Neumann, dependiendo de la familia a la que pertenezca dicho dispositivo.

Los periféricos incorporados en microcontroladores ARM incluyen: Convertidores Analógico – Digital (*ADC: Analog to Digital Converter*) y Digital – Analógico (*DAC: Digital – Analog Converter*), comunicación USB (*Universal Serial Bus*), comunicación serie SPI e

I2C, módulo UART, módulo CAN (*Control Area Network*), modulación por ancho de pulso (*PWM: Pulse Wide Modulation*) y Ethernet. Además, cuentan con memoria flash y SRAM.

Las principales familias de microcontroladores ARM son:

- Cortex – A
  - ARMv7 – A
- Cortex – R
  - ARMv7 – R
- Cortex – M
  - ARMv7 – M
  - ARMv7 – ME
  - ARMv6 – M
- ARM11
  - ARMv6
- ARM9
  - ARMv5T
  - ARMv5TJ
- ARM7
  - ARMv4
- SecureCore
  - ARMv4T
  - ARMv7 – M
  - ARMv6 – M

Existen distintas herramientas de varias compañías para el desarrollo de sistemas con estos dispositivos, están, por ejemplo, el *ARM RealView Development Suite,* que es un ambiente de desarrollo completo (*IDE: Integrated Development Environment*), con el que se puede programar el software para todos los procesadores ARM, dado que integra un compilador C/C++ y un depurador, que son compatibles con herramientas de otros fabricantes.

***INTEL*:** Dentro de los microcontroladores de esta compañía se encuentra familia denominada 8051 (MCS 51, MCS 52). Estos dispositivos, al igual que los ARM, han sido utilizados como base para otros microcontroladores por parte de otros fabricantes, mejorando algunas de sus características e incorporando otras.

Los 8051 son de arquitectura Harvard y cuentan con convertidores ADC y DAC, comunicación USB, I2C y SPI, UART, temporizadores y contadores, módulo PWM, módulo CAN, y reloj de tiempo real (*RTC: Real Time Clock*), entre algunas otras funcionalidades.

Existen varios compiladores de lenguaje C y ensamblador para esta familia de microcontroladores como el *Small Device C Compiler*, *BASCOM-8051* y ambientes de desarrollo como el *Reads51*, que además permite simular y depurar el código escrito en C o en ensamblador.

ANALOG DEVICES: Esta empresa produce microcontroladores basados en los núcleos ARM7 (de arquitectura Von Neumann) y 8052 (de arquitectura Harvard), ambos con un conjunto de instrucciones RISC.

Estos dispositivos cuentan con amplias prestaciones analógicas, tales como: Convertidor *ADC* de 16 bits, Conversor *DAC* de 12 bits, referencia de voltaje, módulo PWM, sensor de temperatura y memoria flash, básicamente.

Las principales familias de estos dispositivos son:

- ADuC7xx (Nucleo ARM7)
- ADuC8xx (Nucleo 8052)

Las familias de microcontroladores mencionadas tienen dispositivos con 8 y hasta 126 KB de memoria de programa. Los dispositivos de la familia ADuC7xxx son capaces de ejecutar hasta 40 millones de instrucciones por segundo (*MIPS*: *Million instructions per second*). En el caso de la familia ADuC8xxos se tiene la capacidad para ejecutar hasta 20 *MIPS.* Las características mencionadas van dirigidas a aplicaciones médicas, de instrumentación y comunicaciones.

ATMEL: Los microcontroladores AVR de ATMEL son dispositivos de arquitectura Harvard y conjunto de instrucciones RISC. Las características más comunes de estos microcontroladores incluyen a convertidores ADC, comunicación SPI, módulo UART, módulo PWM, módulo de captura y comparación y temporizadores, además que utilizan memoria flash (hasta 128 Kbytes). En general, los microcontroladores de ATMEL ejecutan hasta 16 *MIPS*.

Las principales familias de microcontroladores AVR son:

- Tiny AVR
- Mega AVR
- Low Power AVR
- LCD AVR

Existen varias herramientas de desarrollo como el compilador winAVR.

Los AVR se encuentran en aplicaciones como son la automotriz, el control de displays, el control de motores e iluminación y en aplicaciones de propósito general.

FREESCALE: Esta corporación fabrica microcontroladores de 8, 16 y 32 bits, de arquitectura Von Newmann y conjunto de instrucciones CISC. Estos microcontroladores cuentan con módulo SPI, I2C, PWM, temporizadores, ADC, entre otros. Algunos de estos dispositivos cuentan, además, con interfaz USB 2.0 y memoria flash, con la capacidad de ejecutar hasta 60 MIPS.

Las principales familias de microcontroladores FREESCALE son:

- Microcontroladores de 8 bits:
  - RS08
  - HCS08
  - HC08
  - HC05
  - HC11

- Microcontroladores de 16 bits:
  - S12 and S12X
  - 56800/E DSC
  - HC16
  - HC12

- Microcontroladores de 32 bits:
  - 68K/ColdFire
  - Basados en nucleo ARM
  - i.MX
  - Power Architecture
  - PowerQUICC
  - QorIQ

Para desarrollar sistemas basados en estos microcontroladores existe la herramienta llamada *Code Warrior*, que dispone de una versión de evaluación.

Las aplicaciones más comunes de estos dispositivos son el control de displays, control de motores, comunicaciones, entre otras.

FUJITSU: Los microcontroladores de Fujitsu son de arquitectura Harvard y conjunto de instrucciones RISC. Estos dispositivos cuentan con amplia memoria de programa, módulos *CAN* (*Controller Area Network*), convertidores ADC, módulo PWM, comunicación por SPI e I2C, entre otros.

Las principales familias de microcontroladores Fujitsu son:

- Microcontroladores de 8 bits:
  - F^2MC – 8FX

- Microcontroladores de 16 bits:
  - F^2MC – 16FX
  - F^2MC – 16FXS
  - F^2MC – 16LX

- Microcontroladores de 32 bits
  - FR60Lite
  - FR60/70

- FR80/81

La gama de microcontroladores producidos por Fujitsu están enfocados principalmente en aplicaciones industriales y automotrices.

La empresa proporciona el software SOFTUNETM IDE, que permite desarrollar proyectos con sus microcontroladores.

INFINEON: Los microcontroladores que provee esta empresa están basados en el núcleo del procesador 8051 de arquitectura Harvard y conjunto de instrucciones RISC, con capacidad para ejecutar hasta 10 MIPS. Cuentan con módulos de comunicación I2C, UART, temporizadores, contadores, módulos PWM y *ADC*.

Las principales familias de microcontroladores Infineon son:

- Microcontroladores de 8 bits:
  - C500
  - XC800
- Microcontroladores de 16 bits:
  - C100
  - XC100
  - XE100
- Microcontroladores de 32 bits:
  - T1130
  - TC116x
  - TC1167x

Estos dispositivos están diseñados para aplicaciones automotrices, industrial, control de motores, audio e iluminación.

El fabricante provee herramientas de desarrollo para compilar, programar, emular y depurar, además, cuenta con tarjetas de desarrollo para cada familia mencionada.

JENNIC: Los microcontroladores de Jennic son de arquitectura Harvard y conjunto de instrucciones RISC, cuentan con transceptores IEEE 802.15.4. de 2.4 GHz, un módulo de administración de energía, regulador de voltaje, convertidores DAC y ADC, temporizadores, UART, SPI , entre otras características.

Las principales familias de microcontroladores *JENNIC* son:

- JN5148
- JN5139
- JN5121

Estos dispositivos están diseñados para aplicaciones inalámbricas y pueden ejecutar hasta 32 MIPS.

Jennic provee distintas herramientas de desarrollo para sus dispositivos, dentro de las que encontramos al software *Eclipse Integrated Development Environment*. Esta herramienta permite programar, editar y depurar el programa del microcontrolador.

MAXIM: De arquitectura Harvard, los microcontroladores de MAXIM cuentan con cuatro familias principales de acuerdo a la función principal que desempeñan.

- Microcontroladores RISC de 16 bits
  - MAXQ xx
- Microcontroladores de seguridad (*Tamper Resistant*)
  - ZA9L xx
- Microcontroladores con interfaz de internet (*Networked*)
  - DS80C4xx
  - DSTINIS4xx
  - DS80Cxx
- Microcontroladores con núcleo 8051
  - DS89C4xx
  - DS87C5xx
  - DS80C3xx

Dentro de las características que incorporan estos dispositivos las más comunes son módulos PWM, temporizadores, convertidores ADC y DAC, comunicación SPI e I2C.

MAXIM provee distintas herramientas para crear proyectos, para compilar y grabar código, ya sea en lenguaje C o en ensamblador, por ejemplo *Keil Software*.

MICROCHIP: Los microcontroladores de Microchip PIC (*Peripheral Interface Controller*) son de arquitectura Harvard y contienen un conjunto de instrucciones RISC. Cuentan con varios modelos que incorporan periféricos básicos, como son: los módulos de comunicación serie *MSSP*, módulo *USART* y *PWM*, convertidores *ADC*, temporizadores, etcétera. En dispositivos de última generación se cuenta con interfaces USB (*Universal Serial Bus*), CAN, Ethernet y memoria *FLASH*.

Los PIC se clasifican en tres familias o gamas de acuerdo a las características incorporadas:

- Microcontroladores base
  - PIC10 C/F xxx
- Microcontroladores de gama Media
  - PIC12 C/F xxxx

- PIC16 C/F xxxx

- Microcontroladores de gama Alta
  - PIC18 F/LF xxxx
  - PIC24 F/H (16 bits)
  - PIC32 (32 bits)

Para estos dispositivos se dispone de una gran variedad de herramientas de desarrollo y de programación, tanto comercial como gratuita, que son de fácil adquisición o construcción. Todo esto con notas de aplicación y documentación que se puede encontrar en la página del fabricante, en foros y en páginas dedicadas a proyectos con este tipo de microcontroladores.

NATIONAL SEMICONDUCTOR: Esta corporación produce los microcontroladores conocidos como COP (*Control Oriented Processor*) que son de arquitectura Harvard en familias de 8 y 16 bits, estos últimos no recomendados por el mismo fabricante para nuevos diseños.

Los microcontroladores más comunes de *National Semiconductor* son los COP8, que se subdividen en cuatro familias:

- Familia Básica, para aplicaciones sencillas.

- Familia Característica, orientada a comunicaciones.

- Familia OTP (*One Time Programmable*), de programación única.

- Familia S, microcontroladores de última generación.

  - COP8SA
  - COP8SG
  - COP8AC
  - COP8SB/CB

Las principales características que integran estos dispositivos son: convertidores *ADC*, módulos de captura y comparación, módulo PWM y temporizadores, entre otras prestaciones.

National Semiconductor provee software de desarrollo (*IDE: Integrated Development Environments*) como lo es el WCOP8 IDE.

NEC: Los microcontroladores de la empresa NEC cuentan con prestaciones que permiten realizar diseños tanto para aplicaciones de propósito general, como para aplicaciones especificas, entre las que encontramos el control de *display's*, el control de motores, comunicación USB, comunicación Ethernet y control remoto, principalmente.

Las principales familias de microcontroladores NEC se enlistan a continuación:

- Microcontroladores de 8 bits
  - 78KO
- Microcontroladores de 16 bits
  - 78KOR
  - R8C
  - M16C

- Microcontroladores de 32 bits
  - SuperH para display
  - V850 para control de motores
  - SH – Ether para Ethernet

Estos dispositivos se pueden manejar con herramientas de desarrollo y programación como lo es el software Webench.

RABBIT: Esta empresa provee tarjetas de desarrollo basadas en sus microprocesadores Rabbit de 8 bits. Dichas tarjetas incorporan una gran cantidad de puertos de E/S, puertos serie, Ethernet, TCP/ IP (*Transmission Control Protocol/Internet Protocol*) y están diseñadas para proveer un alto rendimiento en la ejecución de funciones matemáticas.

Los sistemas Rabbit se clasifican en las siguientes familias, de acuerdo al microprocesador utilizado:

- Rabbit 5000
  - RCM5400W RabbitCore

- Rabbit 4000
  - RCM4510W Mesh RabbitCore
  - RCM4300 RabbitCore

- Rabbit 3000
  - RCM3900 RabbitCore

- Rabbit 2000
  - RCM2300 RabbitCore

TEXAS INSTRUMENTS (TI): Los microcontroladores de TI están basados en los núcleos ARM. Son de arquitectura Von Newmann y conjunto de instrucciones RISC. Existen con distintos tipos de memoria como la tipo flash, flash-ROM, OTP, por ejemplo. También se incorporan periféricos tales como convertidores *ADC*, módulos de captura y comparación, módulo PWM, temporizadores, contadores, osciladores, comparadores analógicos, sensor de

temperatura, comunicación I2C y SPI, UART, Ethernet, USB, módulo CAN e interfaz IEEE 1588, entre algunos otros.

Las principales familias de microcontroladores TI de 16 bits son:

- MPS430
  - MPS430x1xx
  - MSP430F2xx
  - MSP430x3xx
  - MSP430x4xx

- C2000
  - 28x Delfino
  - 28x Picolo

- Stellaris ARM Cortex – M3
  - Series 1000 – 9000

Estos microcontroladores son de propósito general en su mayoría, aunque existen con características que facilitan el control de displays y el control de motores. Otras aplicaciones de estos microcontroladores incluyen sistemas de iluminación, control digital de motores, sistemas automotrices, aplicaciones médicas, entre otras.

Algunos otros fabricantes de microcontroladores que podemos mencionar son:

- RENESAS
- ST Microelectronics
- Silicon LABS
- NXP
- Toshiba
- OKI
- Silog
- Sanyo
- CIAN

2.6. Dispositivos Periféricos

El microcontrolador cuenta con terminales designadas para la entrada y salida de datos (Puertos de E/S). Dichas terminales son de tipo digital, es decir, detectan o entregan los niveles lógicos ‘1’ y ‘0’.

Los puertos de Entrada/Salida son utilizados para conectar dispositivos auxiliares, que permitirán al microcontrolador obtener información del exterior, si el puerto es configurado como entrada, o para manipular algún indicador o visualizador, si el puerto se configura como salida. Estos dispositivos auxiliares son llamados periféricos.

Los dispositivos periféricos pueden clasificarse como:

- Periféricos de entrada
- Periféricos de salida
- Periféricos de almacenamiento
- Periféricos de comunicación

Dentro de los dispositivos periféricos, el teclado (periférico de entrada) y el display (periférico de salida) son de los más utilizados en conjunto con microcontroladores.

2.6.1. Teclado

El teclado es un conjunto de interruptores, teclas o botones que permitirán introducir datos al sistema. Una tecla es un contacto mecánico que genera un estado lógico (‘1’ y ‘0’) cerrando o abriendo un circuito eléctrico, dependiendo de la posición en la que se encuentre dicha tecla.

De manera ideal, las teclas entregarán una forma de onda como la mostrada en la figura 2.33.

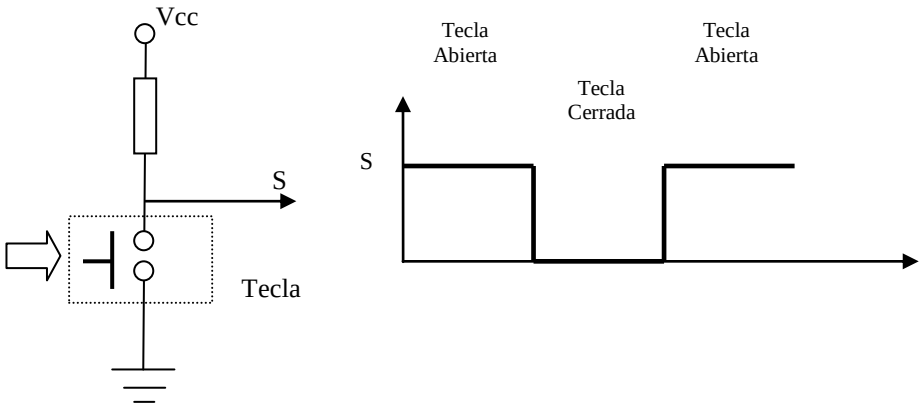


Figura 2.33. Comportamiento ideal de una tecla o botón.

Debido a las características elásticas de los elementos que constituyen a las teclas o botones, al ser accionados éstos se generan vibraciones que alteran la forma de onda entregada, lo que es conocido como efecto de rebote. En la figura 2.34 se muestra el efecto de rebote mecánico producido en los botones de un teclado.

El efecto de rebote mecánico debe ser evitado, debido a que el microcontrolador puede tomarlo como si se pulsara varias veces la tecla, generando errores en las rutinas programadas.

Existen varias maneras de eliminar el efecto de rebote, algunas formas incluyen filtros o compuertas lógicas, esto es, a través de *hardware*. Una solución por medio de *software* implica agregar retardos en la rutina que lee el estado del botón, con el fin de esperar a que se estabilice la señal proporcionada por el interruptor.

A través del uso rutinas de retardo implementadas en el microcontrolador, el programa descarta el ruido producido al accionar alguna tecla y sólo leerá el valor lógico cuando ya se encuentra estable, como se puede observar en la figura 2.35.

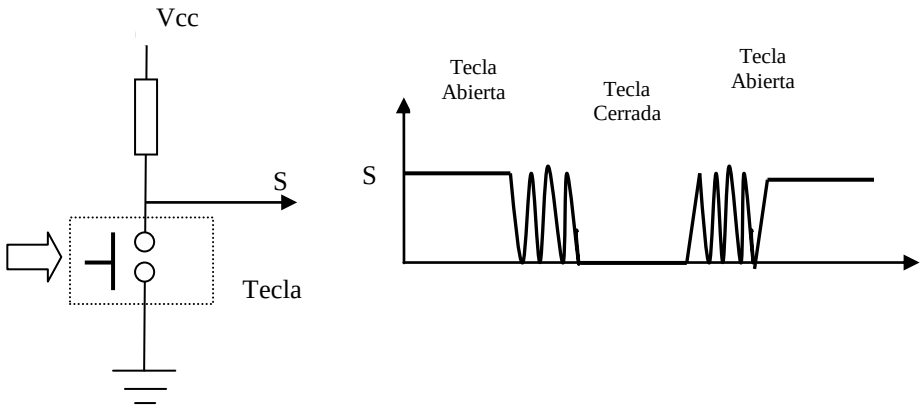


Figura 2.34. Efecto de rebote mecánico al pulsar una tecla.

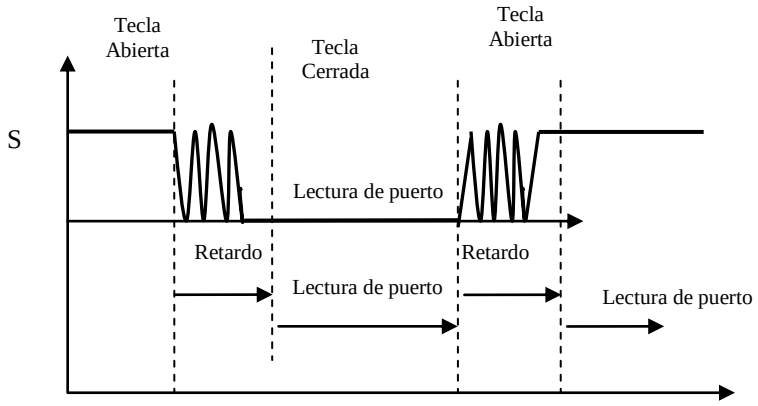


Figura 2.35. Uso de retardos para evitar el efecto de rebote mecánico.

2.6.2. Display de cristal líquido

Dentro de los dispositivos de salida que permiten la visualización de datos podemos encontrar al llamado display de cristal líquido (LCD: *Liquid Crystal Display*).

Como se puede observar en la figura 2.36, un display LCD se compone de dos placas de vidrio paralelas separadas por una capa de cristal líquido. Cuenta, además, con electrodos y filtros de polarización y una capa reflejante.

Los visualizadores LCD no emiten luz para mostrar el mensaje deseado, sino que trabajan con la luz que incide sobre ellos. Para ello aprovechan las propiedades de algunos materiales conocidos como *cristales líquidos*, que absorben o reflejan la luz dependiendo de la alineación que presentan sus moléculas.

El cristal líquido es transparente en estado natural, en el que sus moléculas se encuentran alineadas de forma simétrica. Al aplicar un campo eléctrico a estos materiales se provocará un cambio en la alineación de su estructura, provocando que se vuelvan opacas a la luz, lo que crea un efecto de contraste. En un LCD se aplica el campo eléctrico a través de electrodos que tienen la forma de caracteres o de una matriz de pixeles en el caso del display gráfico (*GLCD Graphic Liquid Crystal Display*). En la figura 2.37 se muestra la apariencia de algunos *display's* de cristal líquido.

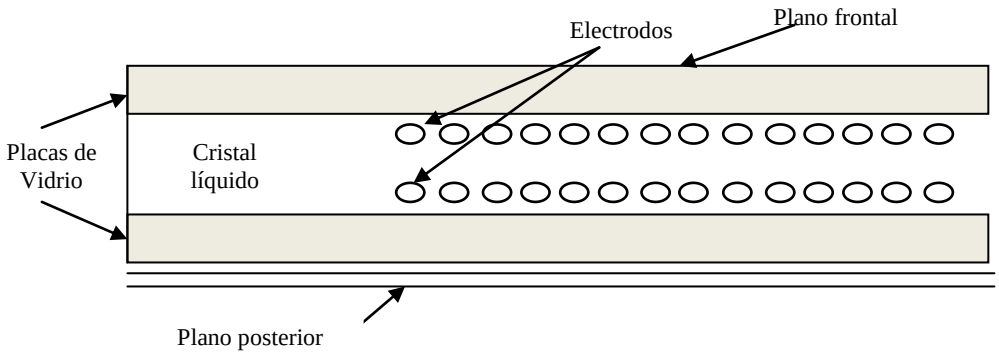


Figura 2.36. Estructura básica de un display LCD.

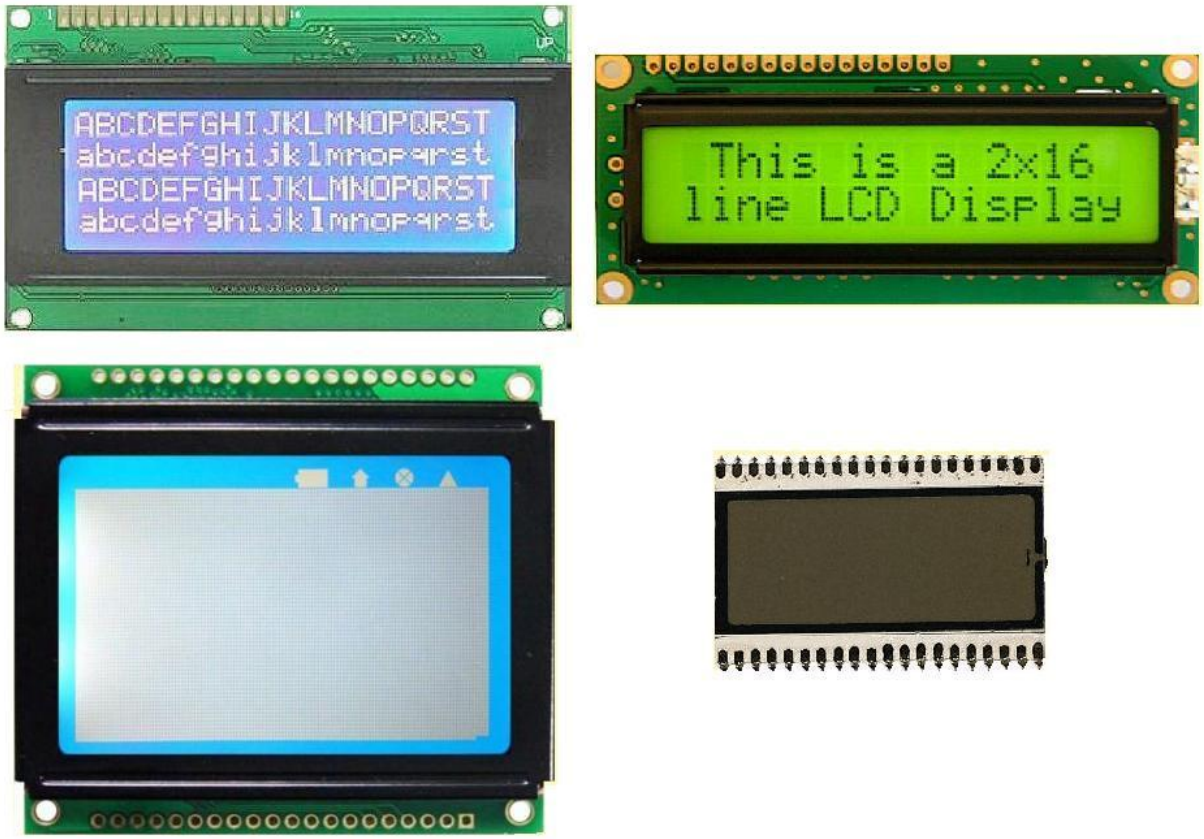


Figura 2.37. Display's de cristal líquido en distintas presentaciones.

2.7. Comunicación serie RS - 232

El estándar de comunicación serie RS232 está basado en la norma RS-232-C, que fue creada en conjunto por la Asociación de Fabricantes Electrónicos (*EIA: Electronic Industries Alliance*), los Laboratorios Bell y algunos fabricantes de equipos de comunicaciones, en 1969.

En dicha norma se definen las características mecánicas, la distribución y disposición de terminales, las señales y los protocolos que debe cumplir este estándar.

En la comunicación serie RS232 se establecen los márgenes de voltaje para los valores lógicos mostrados en la tabla 2.7:

Valor lógico	Margen de voltaje transmisor	Margen de voltaje receptor
1	-15 a -5 V	-15 a -3 V
0	5 a 15 V	3 a 15 V

Tabla 2.7. Márgenes de voltaje para los valores lógicos en el estándar RS 232.

En la figura 2.38 se puede observar el ejemplo de la transmisión en RS 232 del paquete de datos correspondiente al carácter A en ASCII, que en binario se representa como 100 0001.

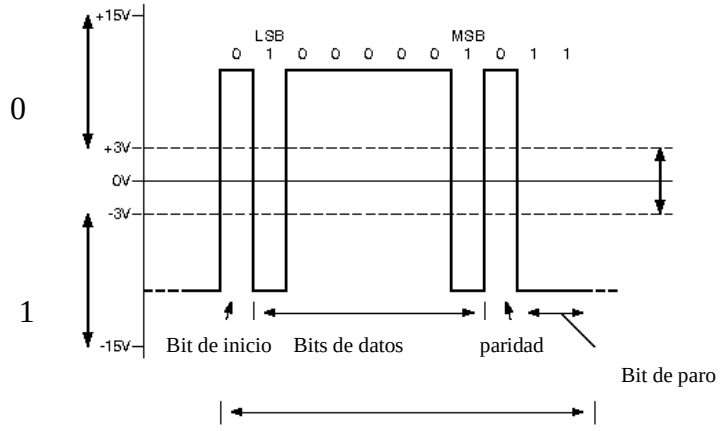


Figura 2.35. Envío del carácter 'A' por RS 232.

En este estándar se puede realizar la transmisión de datos con velocidades de hasta 20 kbps y en una distancia máxima entre transmisor y receptor de 15 metros.

Los conectores designados para realizar la comunicación RS 232 son conocidos como DB25 y DB9 que se muestran en la figura 2.39.

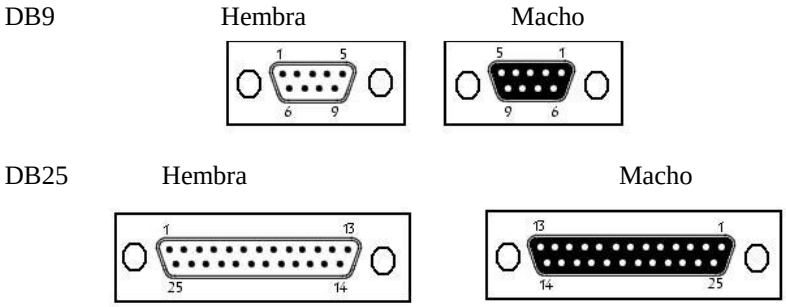


Figura 2.39. Conectores DB9 y DB25 para la comunicación serial RS 232.

En la tabla 2.8 se describe la función de las terminales de los conectores DB9 y en la tabla 2.9 lo referente al conector DB25:

TERMINAL	NOMBRE	FUNCIÓN
1	CD	(*Carrier Detect*) Detector de transmisión
2	RxD	Recepción de datos
3	TxD	Transmisión de datos
4	DTR	(*Data Terminal Ready*) Terminal de datos lista
5	GND	Terminal para Tierra
6	DSR	(*Data Set Ready*) Conjunto de datos listo
7	RTS	(*Request To Send*) Permiso para transmitir
8	CTS	(*Clear To Send*) Listo para enviar
9	RI	(*Ring Indicator*) Indicador de llamada

Tabla 2.8. Descripción de las terminales del conector DB9.

TERMINAL	NOMBRE	FUNCIÓN
1	NC	No conectado
2	TxD	Transmisión de datos
3	RxD	Recepción de datos
4	RTS	(*Request To Send*) Permiso para transmitir
5	CTS	(*Clear To Send*) Listo para enviar
6	DSR	(*Data Sheet Ready*) Conjunto de datos listo
7	GND	Terminal para Tierra
8	CD	(*Carrier Detect*) Detector de transmisión
9 – 19	NC	No conectado
20	DTR	(*Data Terminal Ready*) Terminal de datos lista
21	NC	No conectado
22	RI	(*Ring Indicator*) Indicador de llamada
23 – 25	NC	No conectado

Tabla 2.9. Descripción de las terminales del conector DB25.

Para realizar correctamente la transmisión de datos entre dos dispositivos utilizando el estándar RS 232, los valores lógicos deben estar dentro de los márgenes de voltaje definidos. En algunos casos los niveles de voltaje no son compatibles para ser conectados directamente, como al requerir comunicar una PC con un microcontrolador. Este problema puede resolverse utilizando dispositivos conocidos como transceptores (*transceiver*).

Uno de los transceptores utilizados en la comunicación RS 232 es el *MAX 232*. Este dispositivo permite adaptar la señal con niveles TTL de un microcontrolador a los niveles RS 232 mencionados.

En la figura 2.40 se muestra al transceptor *MAX* 3222 (variante del *MAX* 232) y el circuito propuesto en las hojas de especificaciones correspondientes, adicionalmente se muestra una gráfica en donde se puede observar la forma de onda obtenida en la terminal T1OUT, al introducir una señal cuadrada de 3.3 V de amplitud por la terminal T1IN.

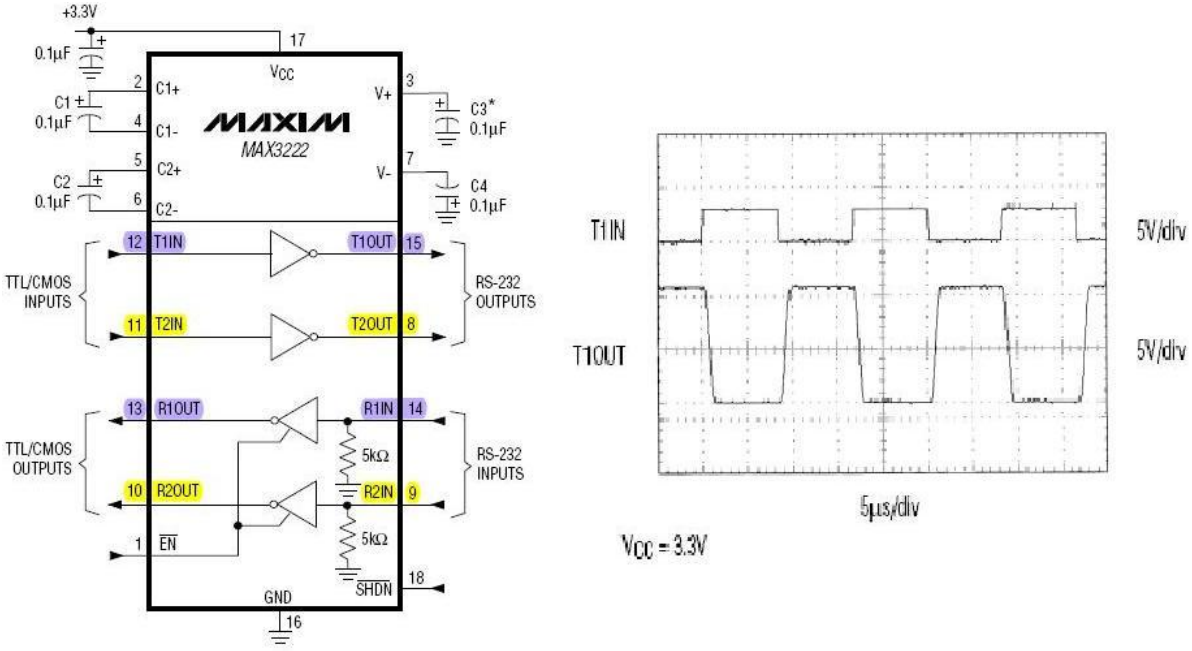


Figura 2.40. Transceptor MAX 3222 para comunicación serial RS 232.

En este capítulo se ha dado un panorama general sobre los temas necesarios para el desarrollo del proyecto en cuestión. En el siguiente capítulo se presentará el proceso de diseño y desarrollo de la interfaz de registro para la unidad sísmica SR04.

3. DISEÑO Y DESARROLLO DE LA INTERFAZ

En el presente capítulo se describe la forma en la que se realizó la interfaz propuesta en el capítulo 1. Comenzaremos por plantear formalmente las necesidades del proyecto. Después, se presentará el *hardware* propuesto para la interfaz mencionada y por último el *software* desarrollado para el microcontrolador.

3.1. Evaluación de las necesidades del usuario

Los especialistas en sismología del Instituto de Ingeniería cuentan con equipos para realizar estudios relativos a fenómenos sísmicos. Uno de estos equipos es la llamada Unidad Sísmica SR04, ampliamente descrita en el capítulo 2. Esta unidad tiene como propósito obtener datos de ruido sísmico y transmitirlos a una computadora, donde se efectúa un procesamiento que permite determinar la estructura de las capas superficiales del terreno donde se realice la prueba. Cabe comentar que para poder realizar cualquier tipo de prueba con dicha unidad, ésta debe estar conectada a un equipo de cómputo, PC o *laptop* y, como se comentó anteriormente, las pruebas muchas veces deben llevarse a cabo en lugares remotos o de difícil acceso. Lo anterior dificulta realizar pruebas en las que se requiera un muestreo por grandes periodos de tiempo.

El propósito principal de la presente propuesta es permitir la ejecución de pruebas a largo plazo, es decir, posibilitar a los especialistas en sismología el registro de las muestras generadas por la unidad sísmica SR04 por largos periodos de tiempo, dígase días, semanas e inclusive hasta de un mes. Lo anterior le otorgará autonomía de funcionamiento al conjunto de la unidad sísmica y la interfaz de registro, sólo se requerirá que se le programe el tiempo de registro y se deje al equipo registrando durante el tiempo indicado.

Con base en las características de la mencionada unidad sísmica y tomando en consideración los detalles de las pruebas que se realizan, se determinó que el sistema a desarrollar deberá cumplir con los siguientes requerimientos:

- Mostrar el funcionamiento de los tres sensores sísmicos
- Mostrar la estampa de tiempo que genera el GPS
- Registrar los datos generados por la unidad sísmica en memorias flash SD

    - Configurar la frecuencia de muestreo de la unidad SR04
    - Establecer el tiempo de registro
    - Crear archivos que cumplan con un sistema de archivos FAT

- Configurar la frecuencia de muestreo de la unidad SR04 con una PC a través de un puerto USB manejado por el microcontrolador

3.2. Desarrollo del Hardware

Para cumplir el propósito señalado en párrafos anteriores se deben integrar una serie de componentes electrónicos y algunos otros elementos que conformarán el *hardware* del sistema. En la figura 3.1 se puede observar un esquema general de los elementos que serán parte del sistema.

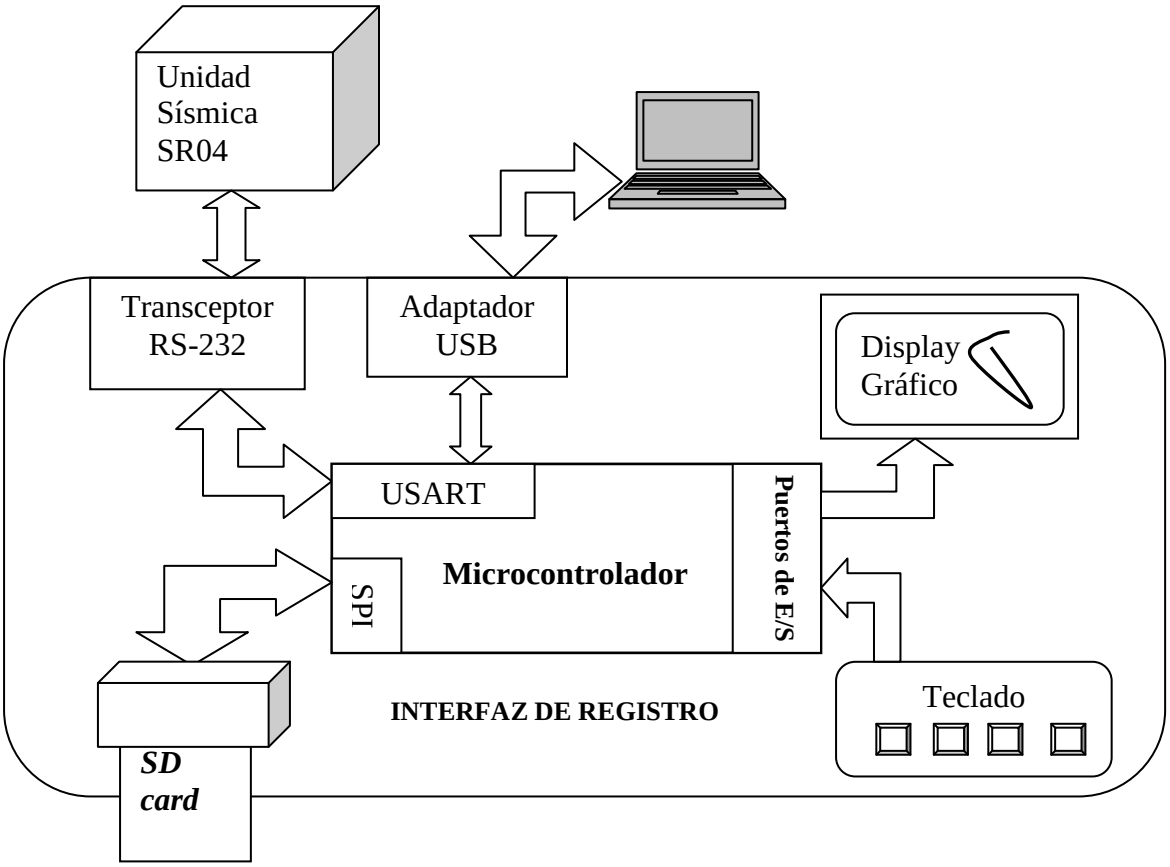


Figura 3.1. Elementos que conformarán el Hardware del sistema.

En seguida se presentaran cada una de las partes que se muestran en el esquema de la figura anterior.

3.2.1. Microcontrolador

El núcleo de todo el sistema a desarrollar es un microcontrolador. La elección del dispositivo a utilizar se basó en dos factores principales:

El primer factor a considerar es la infraestructura con que cuenta el laboratorio de Instrumentación del Instituto de Ingeniería, que es donde se desarrolló este trabajo. En dicho laboratorio se cuenta con equipo para programar microcontroladores de las familias AVR de ATMEL y PIC de Microchip.

Otro aspecto a considerar es la experiencia que se tiene al trabajar con microcontroladores, que ha sido principalmente con los microcontroladores PIC.

Tomando en cuenta los dos puntos mencionados y después de analizar las necesidades del proyecto se optó por utilizar microcontroladores PIC en el desarrollo de la interfaz. Las prestaciones con las que cuentan estos dispositivos permiten integrar todas las necesidades que se requieren para implementar la interfaz de registro propuesta.

3.2.2. Microcontroladores *Microchip PIC*

Los PIC son dispositivos muy populares en el ramo de los microcontroladores de propósito general, debido a diversos factores como su fácil adquisición, su bajo costo, las herramientas de desarrollo, gran documentación y fácil utilización.

Las características generales de los microcontroladores PIC son:

- La arquitectura de los microcontroladores PIC sigue el modelo Harvard.
- Son microcontroladores tipo RISC.
- Se aplica la técnica de segmentación (*pipe-line*) en la ejecución de las instrucciones. Esto quiere decir que el procesador ejecuta una instrucción mientras busca el código de la siguiente instrucción, de manera que se puede ejecutar una instrucción en sólo un ciclo de instrucción (CI). Un CI se lleva a cabo en cuatro ciclos de reloj. En la figura 3.2 se puede observar un ejemplo de esta técnica de ejecución de instrucciones.

Programa ejemplo:	CI 0	CI 1	CI 2	CI 3	Ciclo 4	Ciclo 5
1. MOVLW 55h	Búsqueda de 1	Ejecuta 1				
2 .MOVWF PORTB		Búsqueda de 2	Ejecuta 2			
3. BRA SUB_1			Búsqueda de 3	Ejecuta 3		
4. BSF PORTA, BIT3				Búsqueda de 4	Ejecución de 4	
5. MOVLW 00h					Búsqueda de 5	Ejecución de 5

Figura 3.2. Técnica de ejecución de instrucciones Pipeline.

- Todas las instrucciones son ortogonales, es decir, pueden utilizar cualquier elemento de la arquitectura como fuente o destino.

- El formato de todas las instrucciones tiene la misma longitud: 12 bits para la gama base, 14 para la media y 16 para la gama alta.

- Existen modelos con distintas prestaciones, como son: puertos E/S, ADC, temporizadores, módulos de comunicaciones y memoria, entre otras, facilitando al usuario escoger el dispositivo adecuado a su diseño.

- El fabricante provee algunas herramientas de *hardware* y *software* para el desarrollo y experimentación de sistemas basados en sus microcontroladores. Cabe mencionar que las herramientas son de bajo costo e inclusive algunas, como el ambiente de desarrollo MPLAB son gratuitas.

Microcontrolador PIC 18LF452

En el diseño del sistema en cuestión se propuso el uso del microcontrolador PIC18LF452 como núcleo del mismo. Dicho modelo de microcontrolador permite integrar las necesidades del proyecto, comentadas con anterioridad.

Las características generales del PIC18LF452 se presentan en la tabla 3.1:

Características PIC18F452	
Frecuencia del reloj	Hasta 40 MHz (Modo PLL)
Memoria de programa (bytes)	32 768
Memoria de programa (instrucciones de 16 bits)	16384
Memoria de datos RAM (bytes)	1536
Memoria EEPROM (bytes)	256
Puertos de entrada/salida	Puertos A, B, C, D, E
Temporizadores	4
Módulos de captura/ comparación/ PWM	2
Módulo *Master Synchronous Serial Port*	SPI y I^2C
Comunicación serie (modulo USART)	Comunicación serial RS-485 y RS- 232
Conversor analógico – digital de 10 bits	8 canales
Detección de bajo voltaje programable	Sí
Set de instrucciones RISC (16 bits por instrucción)	75 instrucciones
Voltaje de alimentación	2.0 – 5.5 V
Número de terminales	40

Tabla 3.1. Características generales del microcontrolador PIC 18LF452.

En la figura 3.3 se muestra el diagrama de terminales y el aspecto del PIC18LF452, el empaque mostrado es conocido como DIP (*Dual In Line Package*).

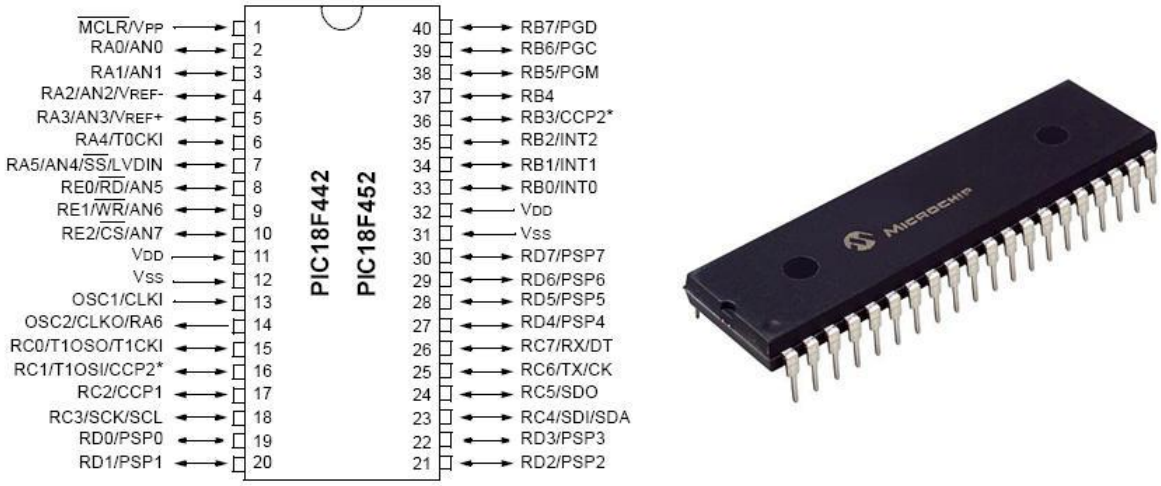


Figura 3.3. Aspecto del microcontrolador PIC18LF452 en empaque DIP.

Oscilador del microcontrolador

La señal de reloj para este microcontrolador será generada por un circuito oscilador externo basado en cristal de cuarzo. El circuito es conectado en las terminales OSC1 y OSC2 del microcontrolador como se muestra en la figura 3.4.

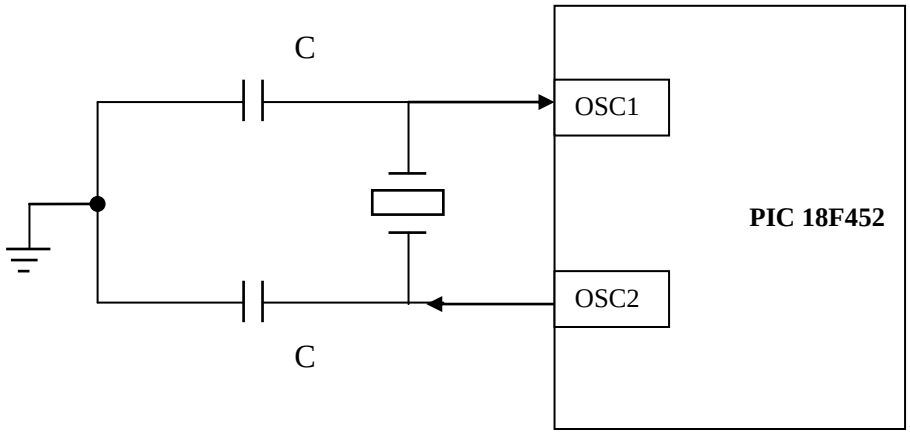


Figura 3.4. Circuito oscilador externo basado en cristal de cuarzo.

El valor de los capacitores C1 y C2 puede ser escogido a partir de la tabla 3.2, en donde se muestran las frecuencias de reloj más comunes y los valores de los capacitores correspondientes que han sido probados. La frecuencia máxima (40 MHz) se logra utilizando el oscilador en modo PLL, que multiplicará por 4 la frecuencia del cristal, en este caso para dicha frecuencia el cristal deberá ser de 10 MHz.

Modo	Frecuencia de reloj	C1	C2
HS (Oscilador por cristal, alta velocidad)	4.0 MHz	15 pF	15 pF
	8.0 MHz	15 – 33 pF	15 – 33 pF
	20 MHz	15 – 33 pF	15 – 33 pF
	25 MHz	15 – 33 pF	15 – 33 pF

Tabla 3.2. Valores recomendados de capacitores para el circuito oscilador.

3.2.3. Tarjeta de memoria SD

Para interactuar con las tarjetas de memoria SD (leer y escribir datos) se requiere conocer sus características principales, que incluyen: la disposición de terminales, voltaje de alimentación, protocolos y topologías de comunicación. Toda esta información se puede encontrar en las especificaciones técnicas (*simplified Version of phisical layer specification*), que fueron comentadas ampliamente en el capítulo anterior.

En el apartado 2.2.2 se explican las características más importantes de estas tarjetas de memoria flash, como son: los distintos modos de comunicación, el Bus SPI, el formato de comandos, los tipos de respuesta, el algoritmo de inicialización, las operaciones de lectura y las operaciones de escritura. Esta información es utilizada en este apartado para realizar la comunicación entre el microcontrolador y la tarjeta de memoria SD, y de esta manera utilizar a dicha tarjeta para almacenar la información proveniente de la unidad sísmica SR04.

En la figura 3.5 se puede observar la disposición de terminales, la forma y las dimensiones de las tarjetas SD. Adicionalmente, en la tabla 3.3 se describen dichas terminales.

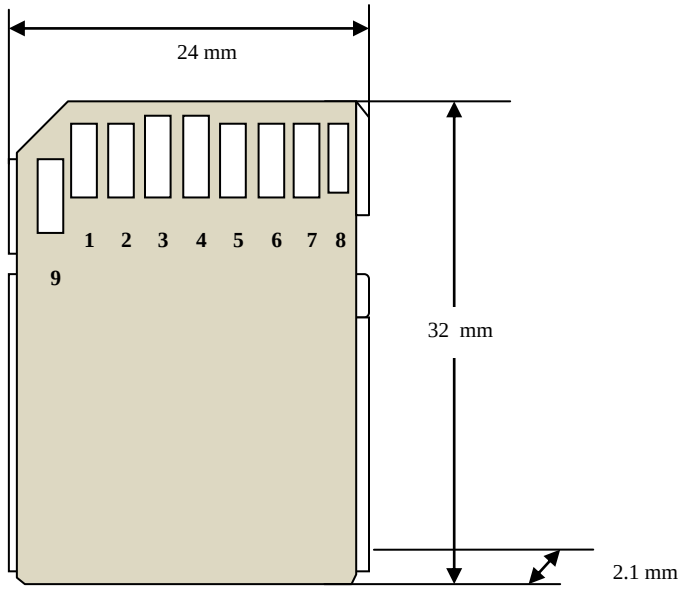


Figura 3.5. Disposición de terminales en una tarjeta de memoria SD.

Terminal	Nombre	Tipo	Función en modo SPI
1	CS	Entrada	Señal de control *Chip Select.*
2	DataIn	Entrada	Comandos y datos hacia la tarjeta.
3	VSS1	Alimentación	Tierra.
4	VDD	Alimentación	Fuente de voltaje.
5	CLK	Entrada	Señal de reloj.
6	VSS2	Alimentación	Tierra.
7	DataOut	Salida	Datos y estado desde la tarjeta.
8	RSV	----------	Reservado.
9	RSV	----------	Reservado.

Tabla 3.3. Función de las terminales de la tarjeta de memoria SD.

Alimentación y habilitación de la tarjeta SD

La tarjeta SD será *encendida* a través de un transistor que funcionará como interruptor, que al ser habilitado le proporcionará el voltaje y la corriente necesaria para funcionar. Dicha habilitación se llevará a cabo conectando la base del transistor a una terminal E/S del microcontrolador designada para dicha tarea, como puede observarse en la figura 3.6.

Cabe comentar que las tarjetas de memoria SD trabajan en un voltaje que va de 2.6 a 3.7 V y que los microcontroladores PIC pueden ser alimentados de 2.0 a 5.5 V, por lo que todo el sistema será alimentado con 3.3 V, asegurando compatibilidad de voltajes entre los distintos dispositivos que serán interconectados.

En el microcontrolador se hará uso de las terminales designadas para la conexión SPI como se muestra en la figura 3.6.

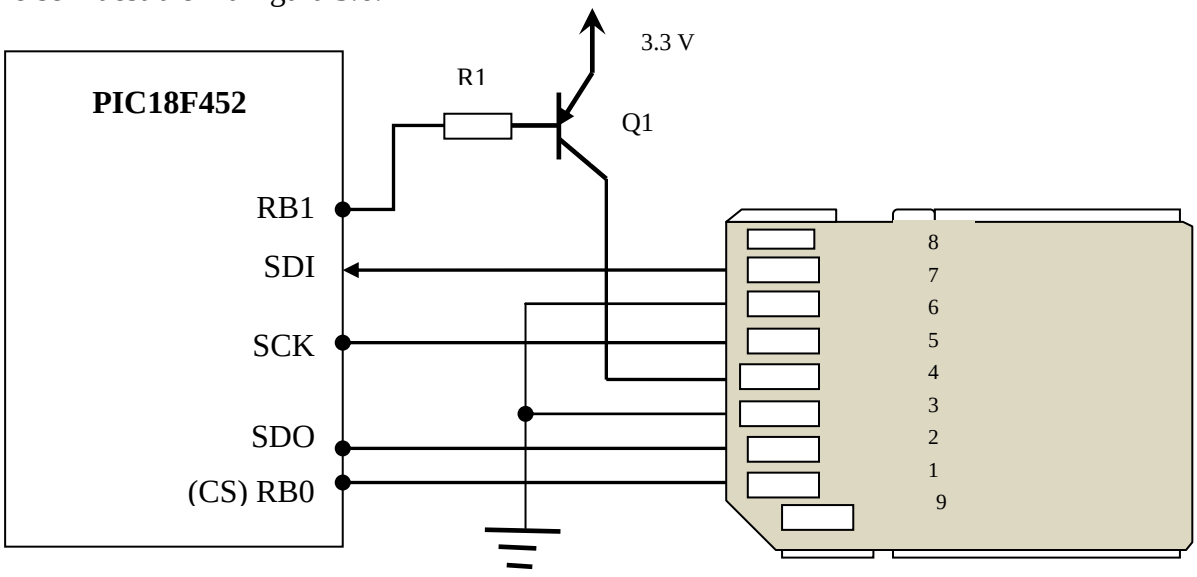


Figura 3.6. Conexión de la tarjeta SD con el microcontrolador.

3.2.4. Transceptor MAX3222

La comunicación del microcontrolador PIC con la unidad SR04 es de tipo serie, con base en estándar RS 232. En la interconexión física de estos dos elementos se deberá cuidar que los niveles de voltaje sean compatibles, para ello se hará uso del transceptor MAX 3222. Este circuito integrado sólo requiere para su operación de cuatro capacitores de 0.1 µF y ser polarizado en un voltaje que esté en el margen de 3.0 a 5.0 V.

En la figura 3.7 se puede observar la conexión que se realiza entre la salida serie del microcontrolador y el conector DB9, a través del cual se logrará la conexión a la unidad sísmica. Del conector DB9 sólo se utilizan las terminales RxD y TxD, además de la terminal de tierra.

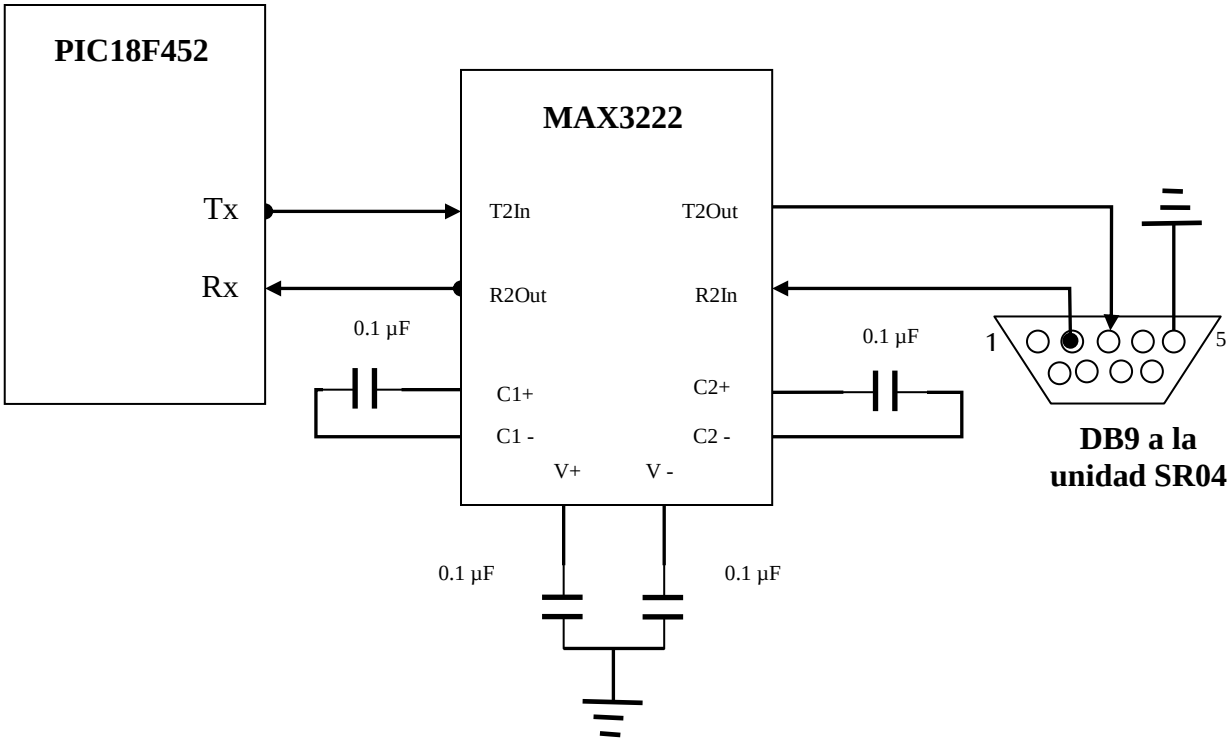


Figura 3.7. Conexión del transceptor con el microcontrolador.

3.2.5. Adaptador RS 232 – USB

Los fabricantes de equipos de cómputo o de comunicaciones actuales han suprimido el puerto serie RS 232 (conectores DB9 ó DB25) y los han ido sustituyendo por puertos USB, provocando que muchos dispositivos se vuelvan obsoletos o que requieran de un adaptador USB – RS232.

En el caso de los microcontroladores con comunicación serie, se puede hacer uso de circuitos integrados que funcionan como interfaz entre el puerto serie del microcontrolador y

un puerto USB. Estos dispositivos manejan todo el protocolo serie–USB y adaptan los niveles de voltaje para las señales involucradas.

Un módulo que integra uno de estos circuitos adaptadores USB - RS232 es el identificado como UB232R de la empresa *Future Tecnology Devices International*, ver figura 3.8. Este módulo recibe directamente la señal generada por el microcontrolador y la adapta a los niveles y al protocolo USB, transmitiendo por un conector mini USB la información generada.

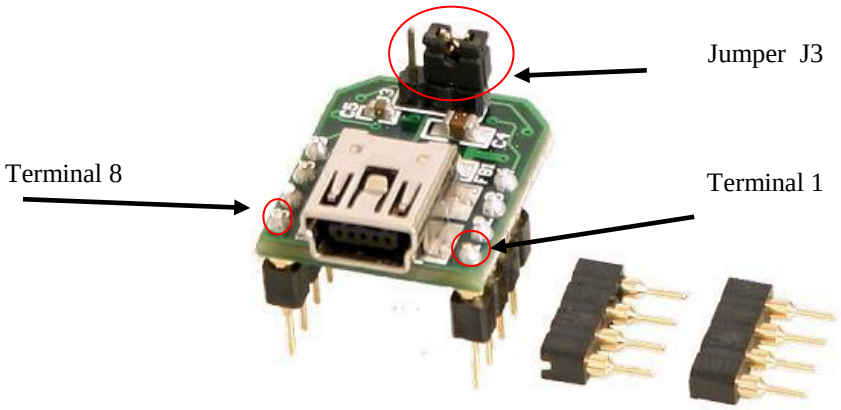


Figura 3.8. Módulo UB232R.

En la tabla 3.4 se tiene la descripción de las terminales del módulo UB232R:

Terminal	Nombre	Descripción
1	GND	Voltaje 0 V
2	Vcc	+ 5 V del puerto USB
3	CTS #	(*Clear To Send*) Listo para enviar
4	RTS #	(*Request To Send*) Permiso para transmitir
5	CBUS1	Terminal para LED indicador Tx
6	CBUS0	Terminal para LED indicador Rx
7	Rx	Recepción de datos
8	Tx	Transmisión de datos
Jumper J3	VCCIO	Selección de voltaje + 3.3V ó + 5V

Tabla 3.4. Descripción de terminales del módulo UB232R.

Este módulo se alimenta directamente del puerto USB donde se conecte. Por medio del jumper J3 se puede elegir si el voltaje al que se trabajará con este módulo es directamente el del puerto USB o si se utilizará el voltaje proporcionado por un regulador interno de 3.3 V.

En la figura 3.9 se muestra la conexión que se realiza entre el PIC18F452 y el módulo adaptador RS 232 – USB.

Cabe comentar que, aunque el propósito de la interfaz es liberar a la unidad sísmica del uso de una computadora para la configuración y el registro de datos, se agregará la conexión USB para poder configurar la frecuencia de muestreo de dicha unidad directamente con una computadora, en caso de ser necesario. Más adelante se comentará con detalle sobre la función que desempeñará el módulo UB232R.

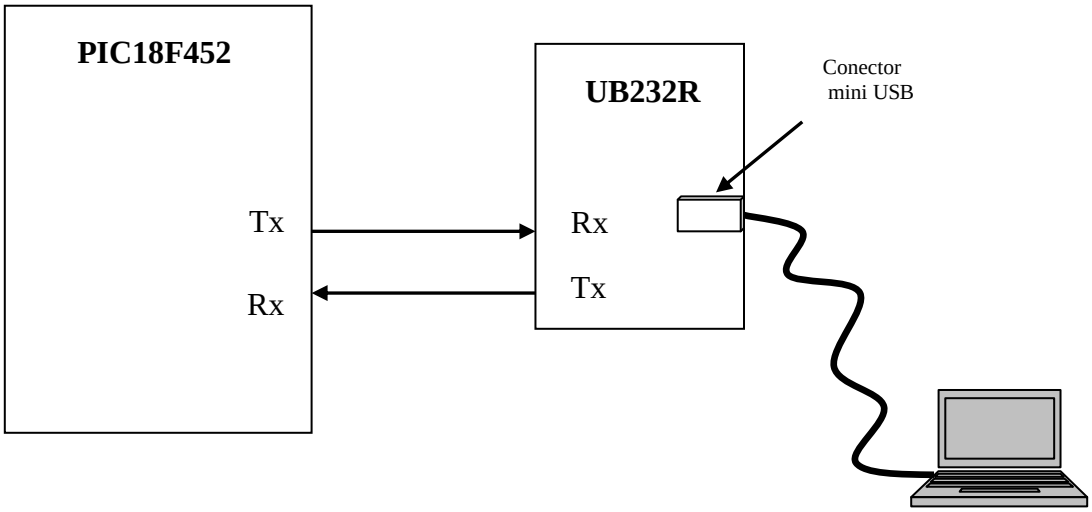


Figura 3.9. Conexión entre el microcontrolador y una PC a través de un adaptador RS -232 a USB.

3.2.6. Display gráfico JHD12864E

El display gráfico JHD12864E de 128x64 puntos está basado en el controlador KS0108. En la figura 3.10 se muestra el diagrama de bloques del display y las señales que se requieren para su control.

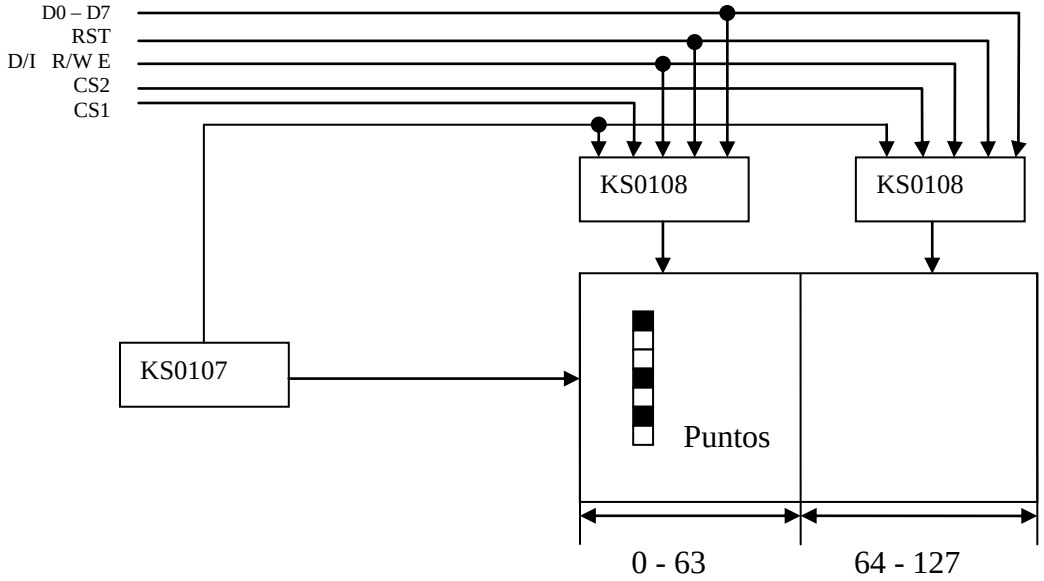


Figura 3.10. Bloques internos del display gráfico JHD12864E.

El display está dividido en dos secciones, que son controladas por dos circuitos KS0108 y por un circuito KS0107. Cada sección comprende un bloque de 64x64 puntos que conforman a la matriz de 128x64 puntos.

En la figura 3.11 se muestra la apariencia del display en cuestión, en donde se puede observar la disposición de las terminales, y en la tabla 3.5 se presenta la descripción de dichas terminales para este display y su correspondencia con el microcontrolador.

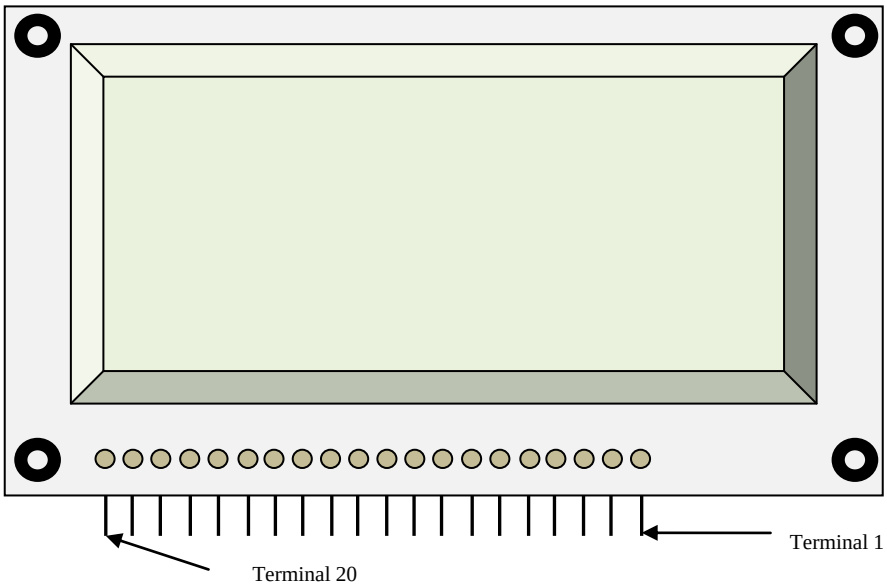


Figura 3.11. Apariencia del display JHD12864E.

Display JHD12864E			**PIC18F452**
Terminal	**Nombre**	**Función**	**Terminal**
1	Vss	0 V	------
2	Vdd	5 V	------
3	Vo	Ajuste del contraste	------
4	D/I	Control de comando o dato	A0
5	R/W	Lectura o escritura de comando/dato	A1
6	E	Señal de habilitación	C0
7 – 14	D0 – D7	Líneas de Datos	Puerto D
15	CS1	Habilitación parte izquierda del display	A3
16	CS2	Habilitación parte derecha del display	A2
17	RST	Señal Restablecer display	C1
18	Vee	10 V internos	------
19	LED +	Voltaje V+ para el LED interno	------
20	LED -	Tierra para el LED interno	------

Tabla 3.5. Descripción de las terminales del display JHD12864E.

El display será integrado al microcontrolador utilizando el puerto D de éste, para enviar los datos y algunas otras terminales para la parte de control.

Para la polarización de este dispositivo se utilizan las terminales **Vss** y **Vdd**. El display cuenta con un circuito interno que genera un voltaje **Vee** de 10 V en la terminal 18.

La terminal **Vo** se utiliza para ajustar el contraste de la pantalla, para lo cual se utiliza un potenciómetro entre esta terminal y **Vee**, como se muestra en la figura 3.12.

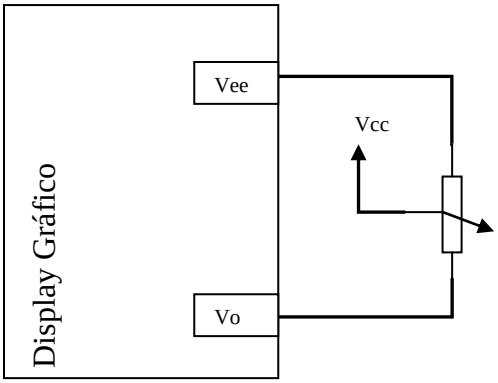


Figura 3.12. Circuito de ajuste del contraste del display gráfico.

Con la terminal de control **D/I** se le indica al display si se le están enviando datos o comando: si esta terminal está en alto se indica envío de datos y, si la terminal está en bajo, se indica envío de comando.

La terminal de control **R/W** permite indicarle al display lectura o escritura de datos. La escritura de datos hacia el display se indica poniendo en alto a esta terminal y la lectura con la terminal en bajo. La lectura tiene el propósito de obtener el estado del display, es decir, si está listo para recibir datos o si se encuentra ocupado.

La terminal de control **E** habilita al display si está en alto, de lo contrario éste se encontrará deshabilitado.

DB0:DB7 son las terminales de datos, a través de las cuales se envía el mensaje que se quiere desplegar en la pantalla.

Cada sección del display es habilitada por las líneas **CS1** y **CS2**, cabe comentar que sólo se puede tener habilitada una sección a la vez, de lo contrario el display podría dañarse o no mostrar el mensaje deseado.

La terminal **RST**, de control, es utilizada para restablecer el display, es decir, para efectuar un *reset*, que es efectivo al tener un cero lógico en esta terminal.

Las terminales **LED +** y **LED –** corresponden al ánodo y al cátodo del LED de iluminación de fondo del display.

3.3. Desarrollo del *software*

En relación al microcontrolador, Microchip pone a disposición de sus usuarios un entorno de programación para sus microcontroladores, se trata de la herramienta MPLAB IDE. Esta herramienta es un *software* que permite programar, depurar, simular y grabar el código en el microcontrolador de la interfaz.

MPLAB puede integrar un compilador de lenguaje C llamado *MPLAB C Compiler for PIC18 microcontrollers*, también conocido como MPLAB C18. Este compilador será usado para desarrollar el programa del microcontrolador de la interfaz.

Dentro de las características principales del compilador MPLAB C18 se cuentan a las siguientes:

- Diseñado para la familia de microcontroladores PIC18
- Cumple con la norma ANSI C
- Librerías para:
    - Funciones matemáticas
    - Control de Periféricos
    - Comunicaciones
    - Conversor Analógico – Digital
    - Temporizadores

3.3.1. Estructura del programa del microcontrolador

De acuerdo a las necesidades del usuario y con base en el *hardware* propuesto, se plantean las tareas que deberán programarse en el microcontrolador a las que llamaremos programa o *firmware*:

- Manejo de la tarjeta de memoria SD:
    - Comunicación por SPI
    - Inicialización
    - Lectura de información
    - Determinación del sistema de archivos de la memoria SD
    - Implementación del sistema de archivos FAT correspondiente
    - Escritura de información en la tarjeta
    - Registro de los datos de la unidad sísmica.

- Comunicación con la unidad sísmica SR04:
    - Comunicación serie RS 232
    - Adquisición de los datos provenientes de dicha unidad
    - Decodificación de los datos mencionados
    - Comunicación entre la unidad sísmica y una PC por USB
    - Configuración de la frecuencia de muestreo

- Manejo del teclado:
    - Detectar la tecla pulsada
    - Manejar la correspondencia entre tecla pulsada y mensaje en el display
    - Asignar una tarea de acuerdo a la tecla pulsada

- Manejo del display gráfico:
    - Mostrar menús de opciones
    - Generar gráficas a partir de los datos adquiridos de la unidad sísmica
    - Indicar a través de mensajes el estado de algún proceso en curso
    - Mostrar resultados de los distintos procesos

El programa del microcontrolador está desarrollado de manera que cada una de las tareas o proceso mencionados pueden ser llamados como una subrutina. Al usuario se le presenta un menú de opciones, donde cada una de éstas lleva a un proceso o a un submenú.

En seguida se presentará el desarrollo de cada una de las tareas mencionadas, comenzando con lo relacionado al manejo de la memoria SD, posteriormente se explicará el desarrollo de la comunicación entre el microcontrolador y la unidad sísmica. Con estos dos puntos se expondrá el desarrollo de los menús de opciones y cómo es que se llaman a las subrutinas mediante éstos.

3.3.2. Inicialización de la tarjeta de memoria SD

La comunicación entre el microcontrolador y la tarjeta SD se lleva a cabo por medio del protocolo SPI, para el cual se utilizará la librería disponible en el compilador C18.

Como se comentó en el capítulo de generalidades, la memoria deberá seleccionar el modo de operación SPI a través de la secuencia que se muestra en la figura 2.6 de dicho capítulo.

El primer paso consiste en energizar a la tarjeta SD, donde se debe considerar un tiempo de encendido y un tiempo de rampa para que el voltaje de operación alcance un valor admitido para los comandos. En la figura 3.13 se muestra el diagrama de encendido.

En las especificaciones se define al tiempo de encendido como el tiempo necesario para que el voltaje de alimentación vaya de 0 V a V^+ min (2.7 V). El tiempo de rampa se define como el tiempo necesario para que la tarjeta alcance el voltaje de operación, que deberá encontrarse entre 2.7 y 3.6 (en nuestro caso 3.3 V).

Siguiendo las especificaciones de las tarjetas SD, el microcontrolador deberá energizar a la tarjeta y ésta tendrá que alcanzar el voltaje V^+ min dentro de los primeros 250 ms. Después de este tiempo se deberán enviar 74 pulsos de reloj, que corresponden al tiempo de rampa requerido para alcanzar el voltaje de operación, esto se ejecuta manteniendo la línea **CS** en estado lógico alto.

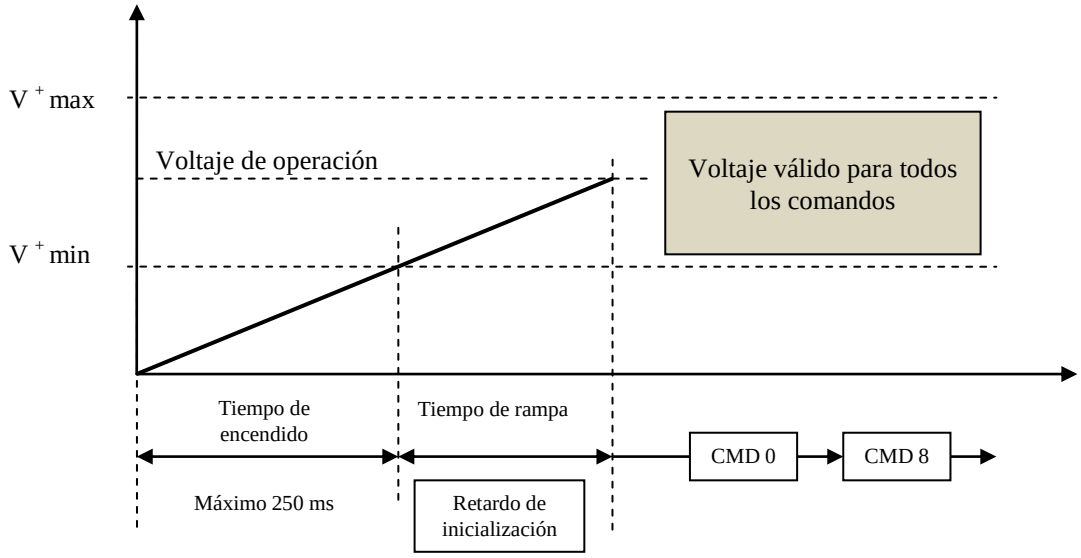


Figura 3.13. Secuencia de encendido de la tarjeta de memoria SD.

Una vez cumplido los tiempos requeridos, la tarjeta estará lista para aceptar los comandos para inicializar su operación en el modo bus SPI.

El primer comando que se envía es el de reset (CMD0), que consiste en un bloque de seis bytes con los valores en hexadecimal que se muestran en la figura 3.14.

Byte 5					Byte 0
0x40	0x00	0x00	0x00	0x00	0x95

Figura 3.14. Estructura del comando de reset CMD0.

Si el comando ha sido aceptado por la tarjeta, ésta envía la respuesta R1 con valor 0x01, lo que significa que la tarjeta ya está trabajando en el modo SPI.

El siguiente comando que se envía es el CMD8, con el cual se comprobará si la tarjeta puede trabajar dentro del margen de voltaje con el que es alimentado el sistema, en nuestro caso 3.3 V. En la figura 3.15 se muestra el bloque de 6 bytes que corresponde a este comando.

Byte 5					Byte 0
0x48	0x00	0x00	0x01	0xAA	0x87

Figura 3.15. Estructura del comando CMD8.

El voltaje que se le provee a la tarjeta se especifica con la parte baja del byte 2 de este comando, de acuerdo a la tabla 3.6.

Valor de los bits	Voltaje suministrado
0000	No definido
0001	2.7 – 3.6 V
0010	Reservado para bajo voltaje
0100	Reservado
1000	Reservado
Otros	No definido

Tabla 3.6. Definición del argumento del CMD8.

Cuando la recepción del comando CMD8 es correcta y además el voltaje suministrado es soportado por la tarjeta, ésta envía la respuesta R7. En esta respuesta en los bits 8 a 11 se regresa el valor del voltaje aceptado por la tarjeta.

Si el comando no es soportado, la tarjeta sólo enviará el primer byte de la respuesta R7, indicando error por comando ilegal. Esto significa que la tarjeta no cumple con la versión 2.00 de las especificaciones técnicas o que es otro tipo de tarjeta de memoria. Así mismo, si el patrón de verificación no corresponde al que fue enviado en el argumento, la comunicación no será válida y habrá que repetir el envío de este comando.

Continuando con la secuencia de inicialización, se envía el comando CMD58, que obtiene el registro OCR (ver tabla 2.3). En la figura 3.16 se observa el formato de este comando.

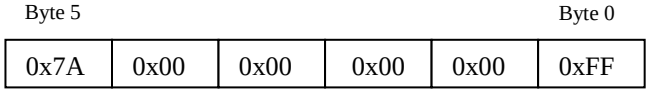


Figura 3.16. Estructura del comando CMD58.

Para este comando la tarjeta envía la respuesta R3, que contiene a los 32 bits del registro OCR, que con los bits 0 al 29 se indican los márgenes de voltaje que soporta la tarjeta, (referirse a fig. 2.13). Con el bit 30 (CCS) de dicho registro se conoce el tipo de capacidad de la tarjeta, es decir, si este bit es 1 la tarjeta es de alta capacidad (*SDHC: Secure Digital High Capacity*) y si este bit es 0 la tarjeta es de capacidad estándar. El bit 31 del OCR indica si el proceso de inicialización ha terminado cuando tiene un valor de 1, de lo contrario aún se encuentra en dicho proceso.

El próximo comando a enviar es el ACMD41, que al ser un comando de aplicación requiere enviar previamente al comando CMD55. Este último comando tiene la estructura que se muestra en la figura 3.17.

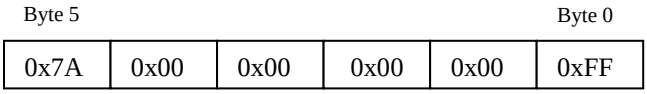


Figura 3.17. Estructura del comando CMD55.

Para el comando CMD55 la tarjeta envía la respuesta R1, que tendrá el valor en hexadecimal 0x01 si el comando ha sido correctamente aceptado, indicando que la tarjeta se encuentra en estado de espera o inactivo (*idle*).

Una vez aceptado el comando anterior, se procede a enviar el comando ACMD41, que permitirá conocer si la tarjeta ha completado satisfactoriamente el proceso de inicialización. Este comando, además, servirá para indicar si el microcontrolador soporta a las tarjetas de alta capacidad (*SDHC*). Para lo anterior, se utiliza el bit HCS en el argumento de este comando como se puede observar en la figura 3.18.

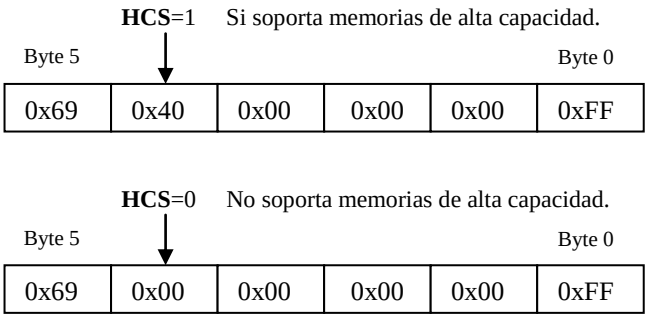


Figura 3.18. Argumentos para el comando ACMD41.

El *host* repetirá el envío de este comando si la respuesta, del tipo R1, indica estado *idle*, equivalente a que el proceso de inicialización aún no se ha completado.

Para terminar con el proceso de inicialización se enviará de nuevo el comando CMD58 y se analizará el estado del bit ***CCS*** de la respuesta R3.

Cuando el bit CCS está en cero indica que la tarjeta recién inicializada es de capacidad estándar y si este bit está en 1 se trata de una tarjeta de alta capacidad (**tarjeta SDHC**), como se puede observar en el diagrama de flujo de la figura 2.4.

Una vez terminado este proceso la tarjeta estará lista para recibir comandos de lectura o escritura por parte del microcontrolador.

3.3.3. Lectura y escritura de datos de las memorias SD

Las operaciones de lectura y escritura de datos de las memorias SD se realizan de acuerdo al esquema presentado en la sección 2.2.4, que se resumen en las figuras 2.7 a 2.10 de dicha sección. En la figura 3.19 y 3.20 se muestran los diagramas de flujo para la función de escritura y de lectura, respectivamente.

Cabe comentar que la dirección especificada en el argumento de los comandos de lectura y escritura tiene dos formatos, de acuerdo al tipo de capacidad de la tarjeta en cuestión. Si la tarjeta es de capacidad estándar, la dirección corresponderá al número del primer byte del bloque de datos. Por el contrario, si la tarjeta es de alta capacidad, la dirección se especificará por el número de bloque, recordando que cada bloque es de 512 bytes.

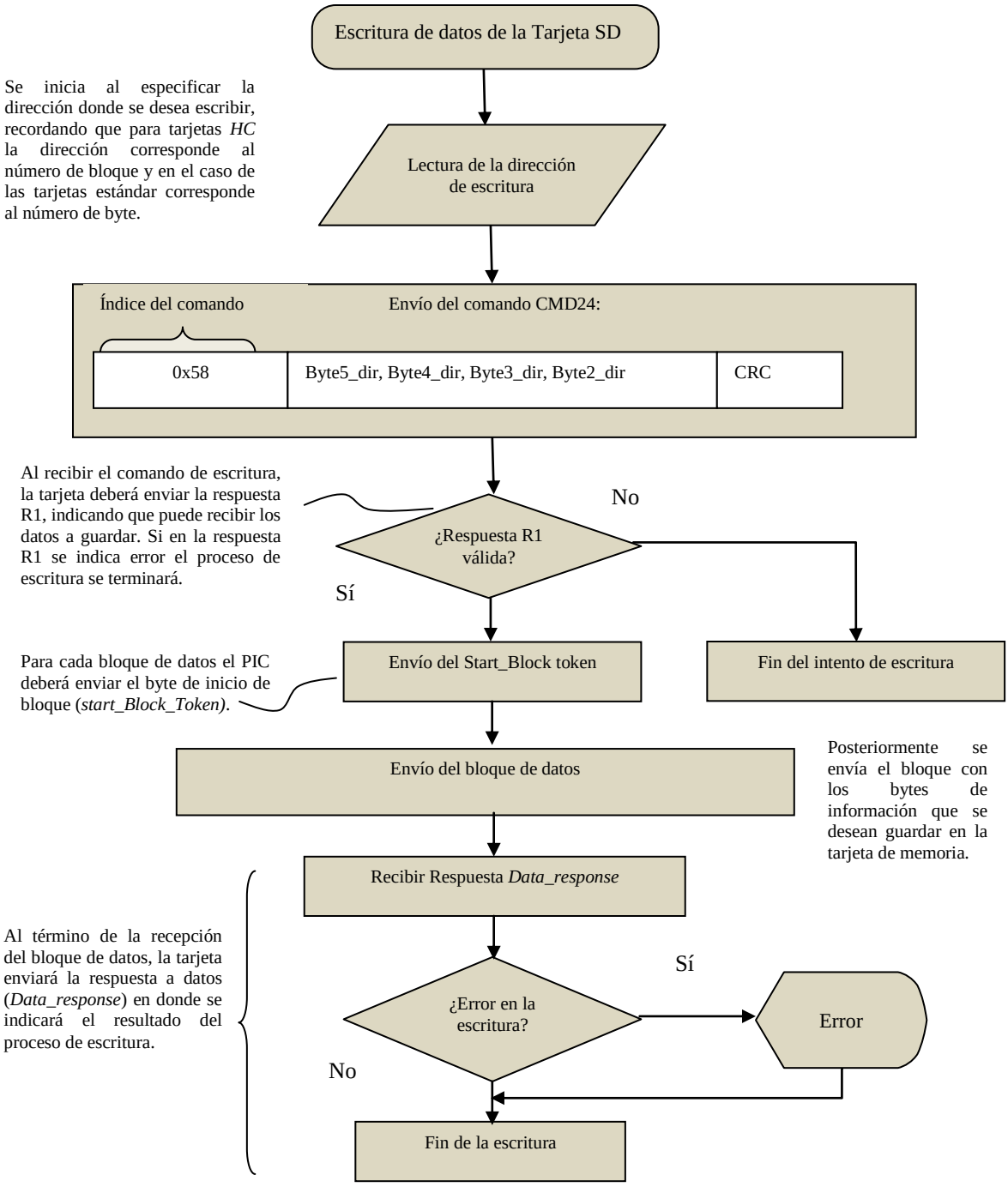


Figura 3.19. Diagrama de flujo para la escritura de datos en la memoria SD.

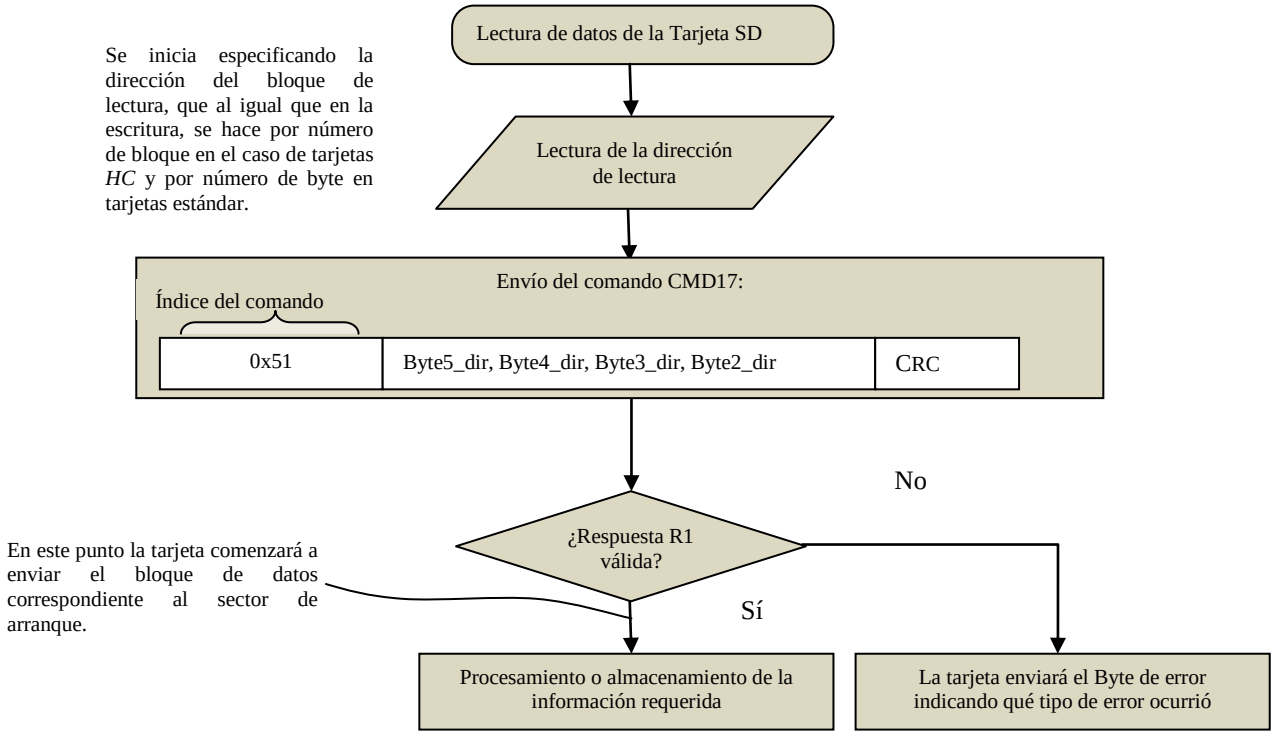


Figura 3.20. Diagrama de flujo para la lectura de datos de la memoria SD.

3.3.4. Implementación del sistema de archivos FAT en tarjetas de memorias SD

Con el fin de que la información de la unidad sísmica, almacenada en las tarjetas SD, sea leída por casi cualquier computadora, dicha información será guardada en archivos que cumplirán con el sistema de archivos FAT (FAT16 ó FAT32).

La implementación del sistema de archivos requiere de un proceso que consiste en los siguientes pasos:

A. Determinación del sistema de archivos de la tarjeta de memoria SD.

B. Cálculo de la posición de las regiones que componen al sistema de archivos (ver fig. 2.22).

C. Creación de archivos:

- **i.** Registro de la entrada en el directorio raíz
- **ii.** Asignación de espacio en disco a través de la tabla FAT

Hay que aclarar que la interfaz de registro requiere que las tarjetas sean previamente formateadas, ya sea en FAT16 o en FAT32.

Determinación del sistema de archivos

Para determinar el sistema de archivos de las tarjetas SD se requiere leer el *BS* o sector de arranque, localizado en el primer sector lógico de las tarjetas. Este sector es un bloque de 512 bytes que contiene varios campos con información sobre el sistema de archivos con el que ha sido formateada la tarjeta. El microcontrolador sólo requerirá algunos de estos campos para identificar al mencionado sistema de archivos.

La lectura del *BS* consiste en el envío del comando CMD17 cuyo argumento será la dirección cero. El esquema de la lectura y almacenamiento de los campos requeridos se muestra en el diagrama de flujo de la figura 3.21. Los datos obtenidos serán almacenados en una estructura tipo arreglo que tiene el formato de la tabla 3.7.

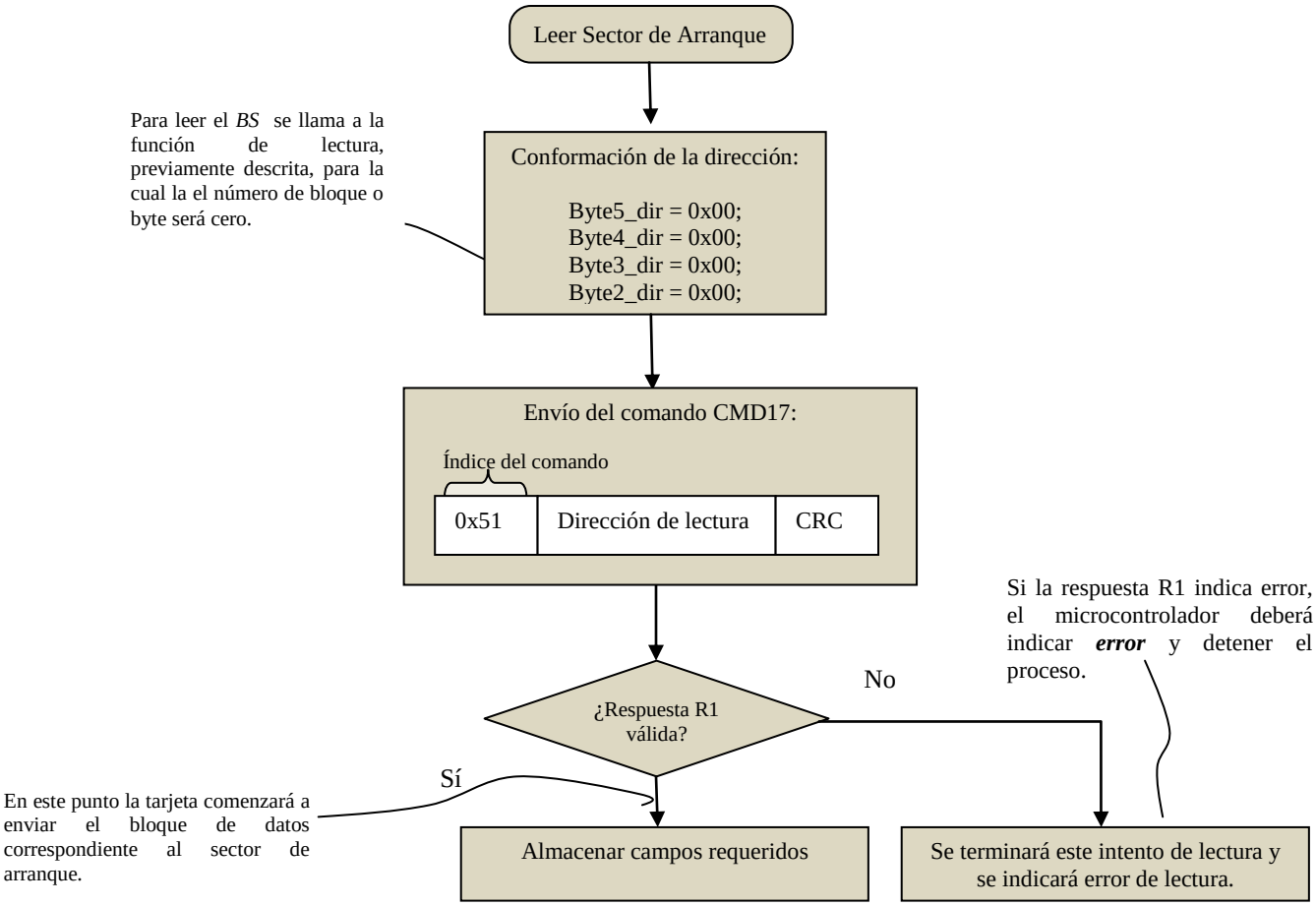


Figura 3.21. Secuencia de lectura del sector de arranque.

Con la información obtenida se realiza el procedimiento descrito en la sección 2.3.1, en lo referente a la determinación del sistema de archivos. El primer paso consiste en determinar la cantidad de sectores ocupados por la región del directorio raíz. En la figura 3.22 se muestran

los pasos para obtener el parámetro *RootDirSecs* a partir de los campos leídos anteriormente, de acuerdo a las especificaciones de *Microsoft*.

Nombre	BS[i]	Offset (bytes)	Tamaño (Bytes)
BytesPerSec	BS[0]	11	2
SecsPerClus	BS[2]	13	1
RsvdSecCnt	BS[3]	14	2
NumFATs	BS[5]	16	1
RootEntCnt	BS[6]	17	2
TotSec16	BS[8]	19	2
FATSz16	BS[10]	22	2
TotSec32	BS[12]	32	4
FATSz32	BS[16]	36	4
RootClus	BS[20]	44	4

Tabla 3.7. Estructura del arreglo BS con los campos requeridos del sector de arranque.

Se toma el valor de BS[6] correspondiente al parámetro *RootEntCnt*. Si dicho parámetro es diferente de cero, se calculará el número de sectores que ocupa el directorio raíz, almacenando dicho valor en RootDirSecs. De lo contrario se asignará un valor de cero a esta última variable.

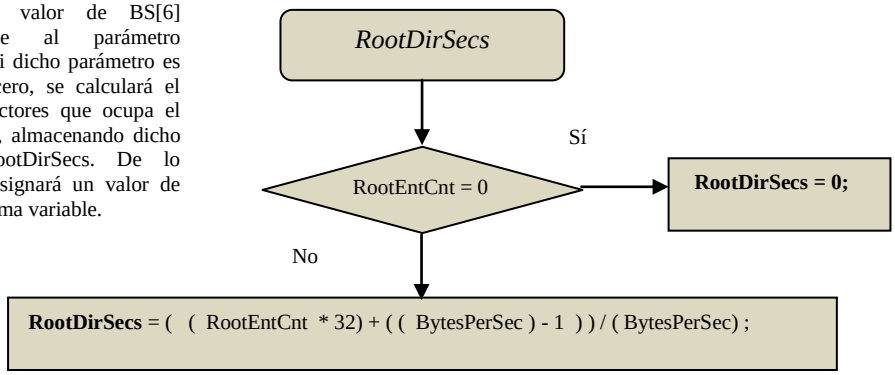


Figura 3.22. Obtención del parámetro RootDirSecs.

El siguiente paso de este proceso consiste en obtener la cantidad se sectores que ocupa una de las tablas de la región FAT, y que se asignará a la variable FATSz. Esto requiere de los campos FATSz16 y FATSz32, operados conforme al diagrama de flujo de la figura 3.23.

De a acuerdo al valor del elemento BS[10], que corresponde a la cantidad de sectores ocupados por una tabla FAT en FAT16, se asignará el valor de este elemento a la variable FATSz, de lo contrario tomará el valor del elemento BS[16] correspondiente a FAT32.

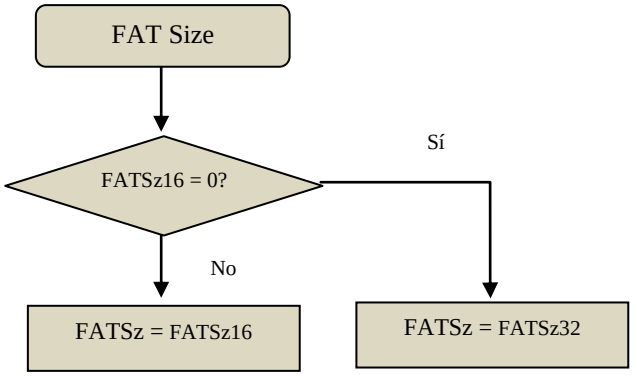


Figura 3.23. Determinación de la cantidad de sectores ocupados por la región FAT.

El siguiente cálculo a realizar es para determinar la cantidad de clústers de la región de datos, esto es, a partir de la cantidad de sectores en la región de datos (R4). En la figura 3.24 se visualiza el proceso para obtener el parámetro *DataSecs* y la conversión de esta cantidad a clústers.

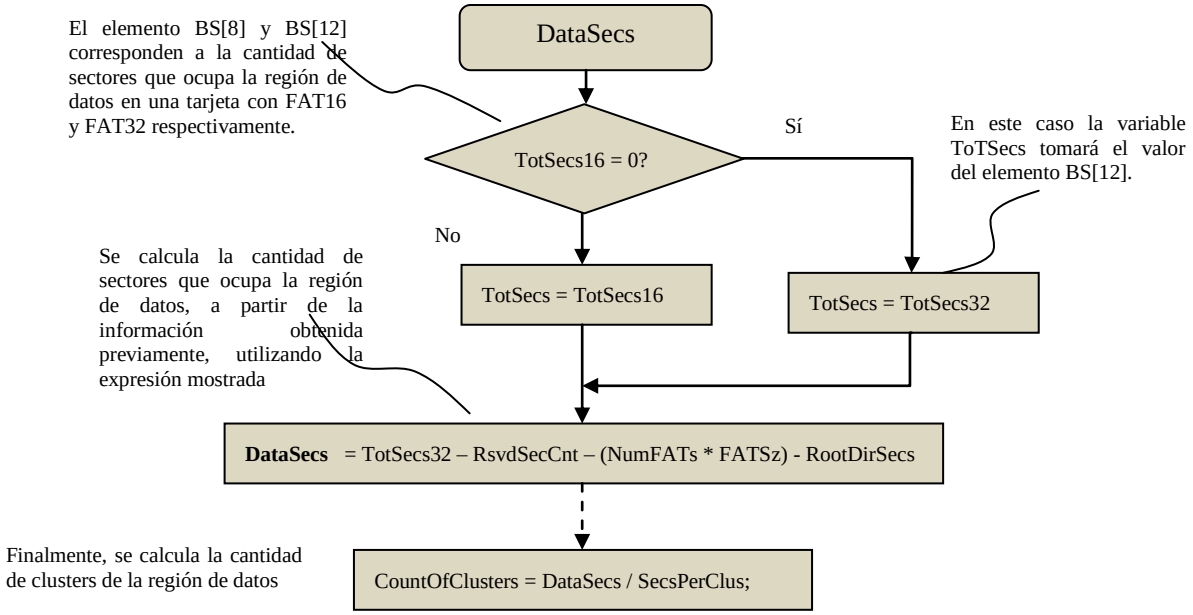


Figura 3.24. Cálculo de la cantidad de sectores en la región de datos.

Por último, de acuerdo a la cantidad de clústers en la región de datos, previamente calculada, se determina el tipo de FAT, tomando en cuenta el criterio presentado en el apartado 2.3.1 y, recordando que el sistema en cuestión solo soportará a los sistemas FAT16 y FAT32, indicando error si la tarjeta tiene sistema de archivos FAT12 (ver figura 3.25) .

Es sólo al terminar los pasos indicados en dicha figura como se pude tener certeza del tipo de sistema de archivos con el que se cuenta en la tarjeta de memoria SD.

Posición de las regiones del sistema de archivos

Una vez que se conoce el sistema de archivos se pude determinar la posición de cada una de las regiones que componen a dicho sistema (ver figura 2.22), que implica obtener el número del primer sector de cada región.

Se pude intuir que el primer sector de la región cero es a su vez el sector cero.

En lo que se refiere a la región FAT (R1), la obtención del número de su primer sector (*FATBegin*) se determina a partir del campo *RsvdSecCnt*, que corresponde al número de sectores que componen a la región reservada, conforme a la siguiente expresión, Ec. 3.1:

$$FATBegin = RsvdSecCnt \qquad \textbf{Ec. 3.1.}$$

Para la región del directorio raíz (R2) hay que recordar que existe una sección de tamaño fijo, en el caso de FAT16, en cambio, para FAT32 el directorio raíz se encuentra en el primer sector de la región de datos y su tamaño dependerá de la cantidad de archivos o carpetas. Adicionalmente el parámetro *RootClus* del *Boot Sector* establece el número de cluster del directorio raíz, por lo general es el cluster 2 pero no necesariamente. En la figura 3.26 se muestra la forma de determinar el primer sector del directorio raíz.

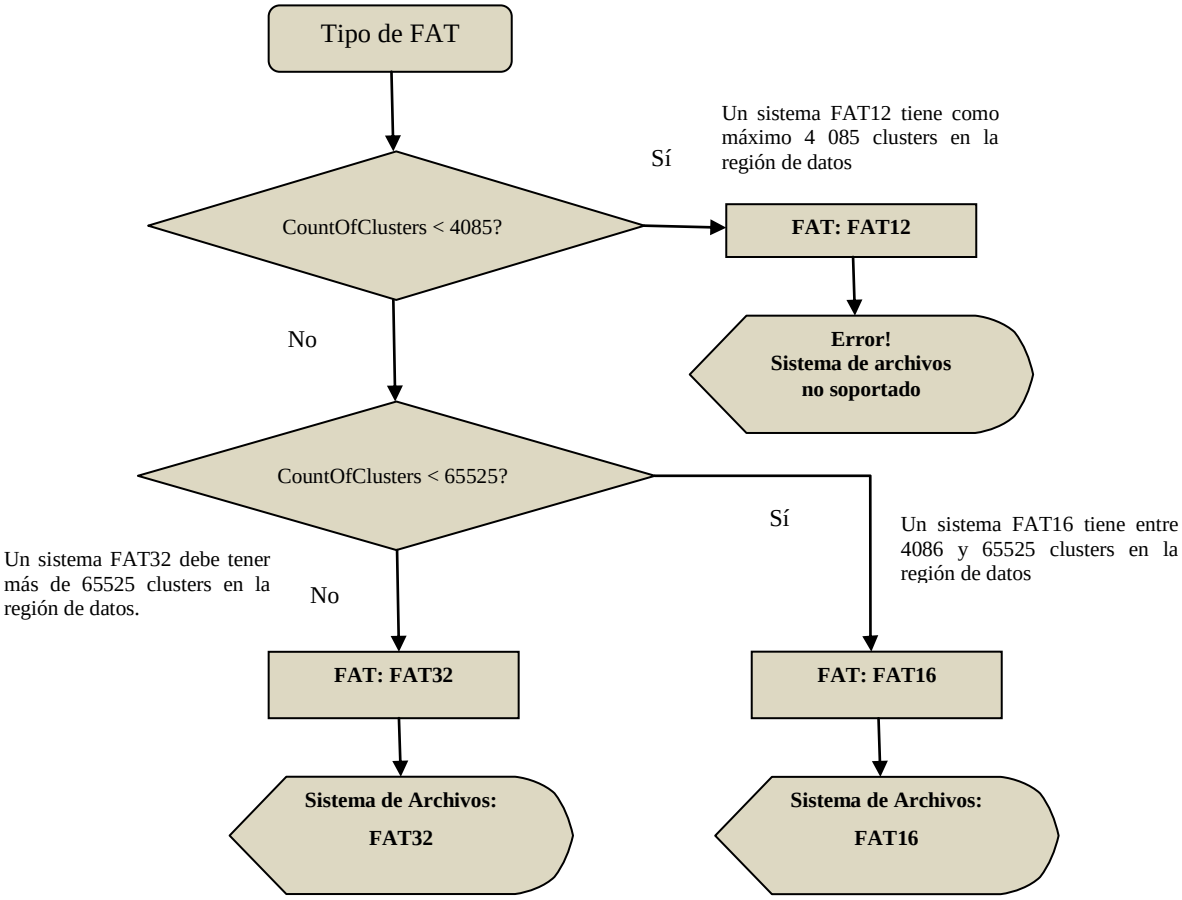


Figura 3.25. Determinación del tipo de sistema de archivos.

Por último, se calcula el primer sector de la región de datos. Cabe señalar que las especificaciones de *Microsoft* establecen que la región de datos para sistemas de archivos FAT32 se encuentra a partir del clúster 2. En caso de que el sistema de archivos sea FAT16 se debe realizar el cálculo de la expresión 3.2:

$$\boldsymbol{FirstDataSector = RsvdSecCnt + (NumFATs * FATSz) + RootDirSecs} \qquad \textbf{Ec. 3.2}$$

Después de este último cálculo se conoce la posición exacta de cada una de las regiones del sistema de archivos. Esta información permitirá crear archivos que podrán ser leídos en una computadora que soporte el sistema de archivos FAT implementado.

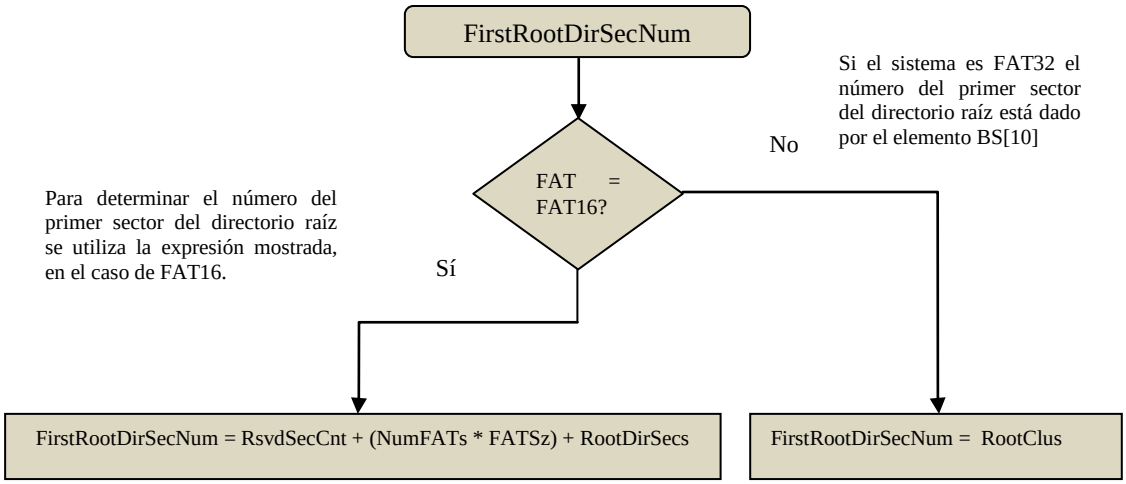


Figura 3.26. Determinación del primer sector del directorio raíz.

Creación de archivos

La creación de archivos consiste en registrar las entradas correspondientes en el directorio raíz, en la tabla FAT, y grabar los datos en el área asignada de la región de datos, básicamente. Este proceso requiere de los siguientes pasos:

- Registro de la entrada en el directorio raíz
- Cálculo de las entradas en la tabla FAT
- Escritura de las entradas de la región FAT
- Registro de datos dentro del área designada

Registro de entradas en el directorio raíz

En la sección 2.3.1 se comentó la estructura del directorio raíz, misma que está sintetizada en la figura 2.24. Conforme a esto, se crearán las entradas para los archivos necesarios. Los primeros ocho bytes de cada entrada corresponden al nombre del archivo, cada byte corresponde al código ASCII de las letras de dicho nombre, como se ejemplifica en la figura 3.27. Los tres byte siguientes corresponden a la extensión, que indica de qué tipo de archivo se trata, y como en el caso del nombre, se especifica cada letra con el código ASCII correspondiente.

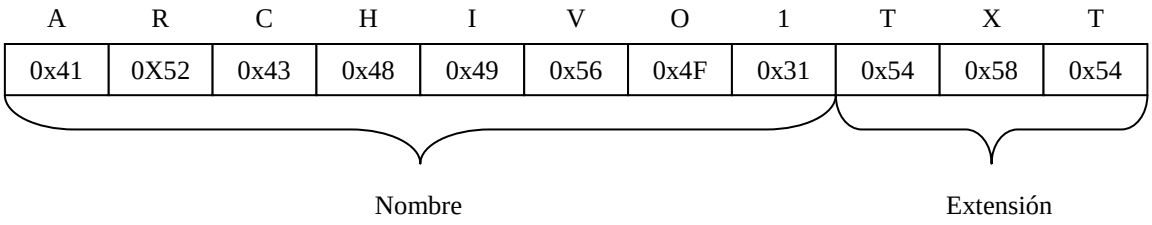


Figura 3.27. Valor de los bytes en hexadecimal del nombre y extensión de un archivo.

Después del nombre y la extensión del archivo se encuentra un byte de atributo a través del cual se indicarán las propiedades del archivo. En la tabla 3.8 se presentan los posibles valores que pueden tomar el byte de atributo y su significado.

Valor hexadecimal	Atributo
0x01	Archivo de sólo lectura
0x02	Archivo oculto
0x04	Archivos de sistema
0x08	*Volume_ID*
0x10	Directorio
0x20	Archivo general

Tabla 3.8. Descripción del byte de atributo.

Siguiendo al byte de atributo se encuentran ocho bytes reservados y que para fines de este proyecto se escribirán como ceros (0x00), ver figura 3.28.

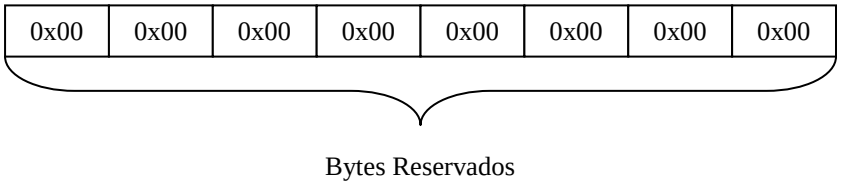


Figura 3.28. Estructura de los bytes reservados.

Inmediatamente después de estos bytes se encuentra la parte baja del número de primer clúster del archivo en cuestión. Hay que recordar que este parámetro tiene un tamaño de 4 bytes que se divide en dos partes: alta y baja. La parte baja del número del primer clúster viene después de otros cuatro bytes reservados. En la figura 3.29 se muestra un ejemplo de la asignación del número del primer clúster, dividiéndolo en parte baja y parte alta e incluyendo el byte reservado que los separa. Debe notarse que estos valores se encuentran en *Little Endian.*

Por último, se encuentran los cuatro bytes correspondientes al tamaño del archivo. Este valor también tiene formato en *Little Endian*.

Cada entrada del directorio raíz consta de 32 bytes por lo que en un sector sólo se pueden registrar 16 entradas.

El primer registro del primer sector del directorio raíz corresponde al llamado *VolumenID.* Este registro contiene el nombre del medio de almacenamiento, además, se debe indicar en el campo de atributo que es el *Volume_ID* y como tamaño del archivo se le asigna el valor cero. Cabe notar que sólo debe haber un registro con dicho atributo.

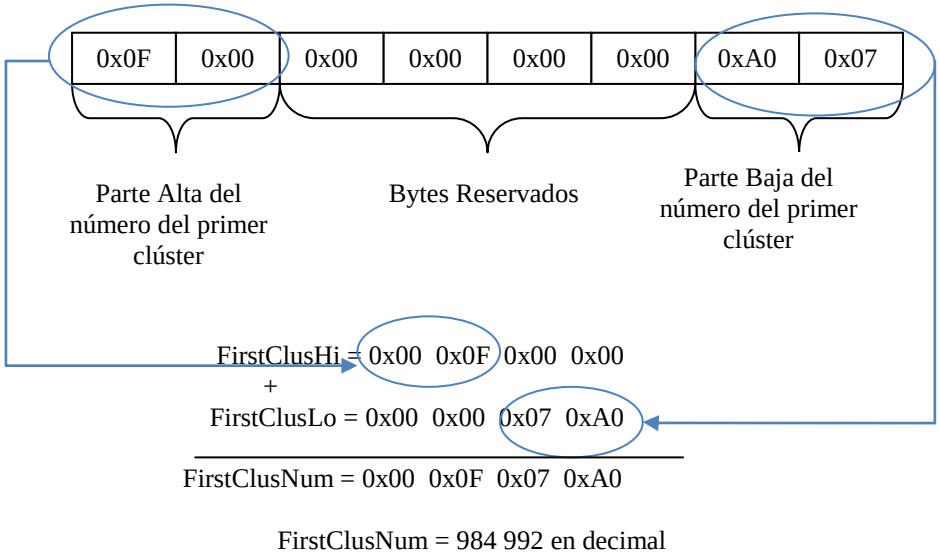


Figura 3.29. Conformación del número del primer clúster.

Para la interfaz de registro, el nombre en las entradas para cada archivo dependerá de la información proporcionada por la unidad sísmica. Se tomara en cuenta el día, el mes y el año para conformar el nombre del archivo. Si existe más de un archivo, el último byte del nombre servirá para diferenciarlos, ver figura 3.30.

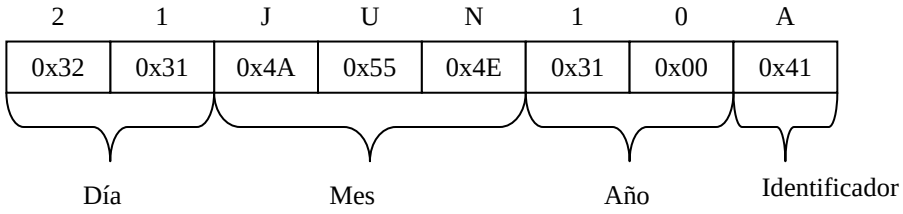


Figura 3.30. Disposición del nombre del archivo en la entrada del directorio raíz.

Con respecto al número del primer clúster, para el primer archivo será fijado por conveniencia en 0x23 y para los siguientes se agregará un offset de acuerdo a la cantidad de clústers del archivo que se está escribiendo.

En cuanto al tamaño, éste será determinado por el usuario, al especificar el tiempo de registro y la frecuencia de muestreo de acuerdo a la expresión 3.3:

$$\boldsymbol{FileSize = \big((15 * m.p.s.) + 9\big) * segs + Num} \qquad \boldsymbol{Ec. 3.3}$$

En la expresión anterior:

- **15** = (3 canales) * (5 Bytes por muestra)
- **9 =** son los bytes por segundo de la marca de tiempo
- **mps** = Frecuencia de muestreo

- **segs** = Cantidad de segundos: 3600 para registro en horas, 86 400 por dia
- **Num** = Cantidad de días u horas a registrar (parámetro introducido por el usuario)

En una sección posterior se presentarán mas detalles de la creación de las entradas del directorio raíz, esto es, cuando se presente lo relativo a la interacción del usuario con el teclado y el display a través de un menú en pantalla.

Cálculo de las entradas en la tabla FAT

En el directorio raíz sólo se registra el número de primer clúster de cada archivo, sin embargo, para que el sistema operativo de la computadora que intente leer al archivo lo haga con éxito, debe conocer el resto de los clústers que conforman al archivo, además del orden en que se encuentran.

En el caso de nuestra interfaz de registro la asignación de clústers será de forma continua, lo que facilitará el registro.

El registro de las entradas FAT comienza por determinar la cantidad de clústers que serán asignados al archivo, de acuerdo al tiempo que el usuario requiere registrar datos, lo que determinará el tamaño del archivo.

En la figura 3.31 se ilustra el proceso para determinar la cantidad de clústers asignados al archivo, recordando que la unidad mínima de almacenamiento en un sistema de archivos FAT es un clúster. Cabe comentar que en las operaciones de división sólo se toma en cuenta la parte entera, por lo que se utiliza la función "**%**" para obtener el residuo de la división de dos números, esto, para saber si quedan bytes por asignar.

El parámetro *CntOfClus* se refiere a la cantidad de clústers que ocupará un archivo.

El número de entradas en la tabla FAT es equivalente al número de clústers del archivo, por ejemplo, un archivo que ocupa 4 clústers tendrá 4 entradas.

El número de entradas en un sector dependerá del sistema de archivos, recordando que en un sistema de archivos FAT**16** cada entrada será de **16** bits y en FAT**32** las entradas son de **32** bits, por lo que se debe realizar la operación de la ecuación 3.4, para conocer la cantidad de entradas que se registrarán en cada sector de la región FAT:

$$FATCntOFEntPerSec = BytesPerSec/FATSizeOfVal \qquad \textbf{Ec. 3. 4.}$$

En la expresión anterior:

- BytesPerSec: Número de bytes por sector.

- FATSizeOfVal: Número de bytes por entrada; 16 para FAT16 y 32 para FAT32.

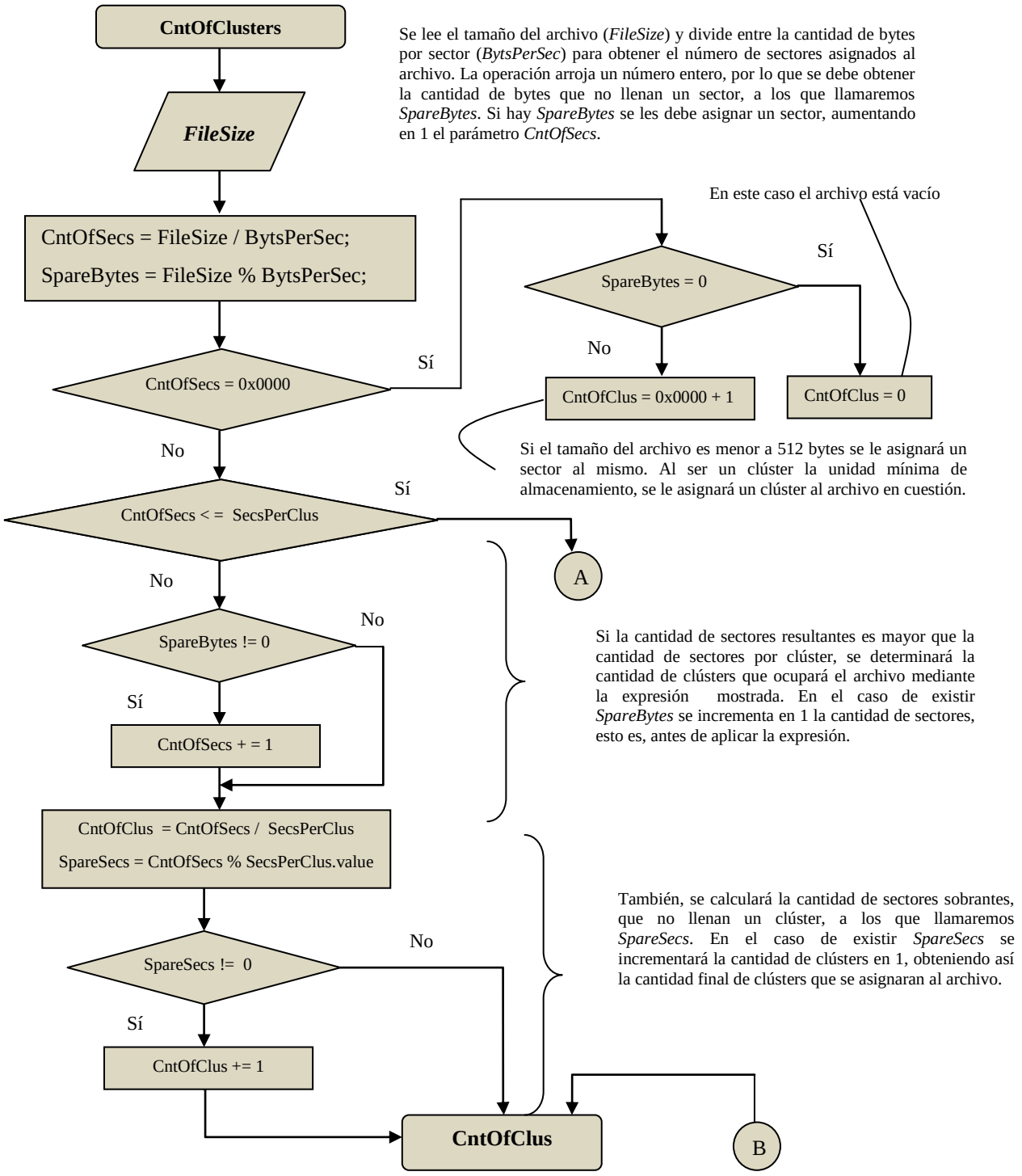


Figura 3.31. Determinación de la cantidad de clústers para un archivo (1 de 2).

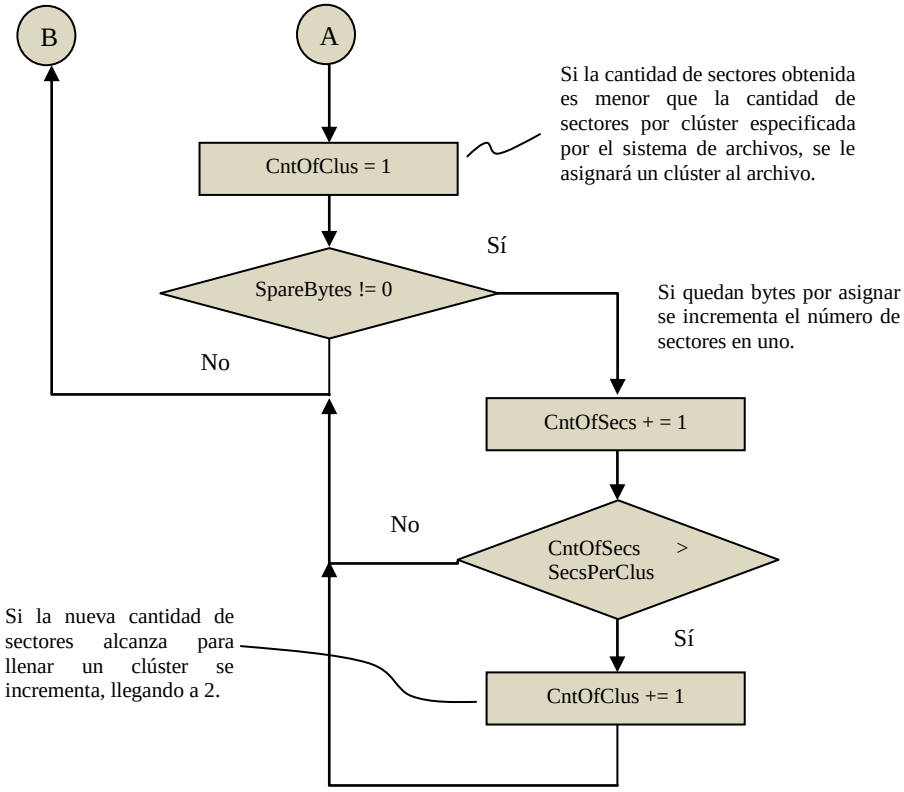


Figura 3.32. Determinación de la cantidad de clústers para un archivo (2 de 2).

Para iniciar el registro de las entradas se determina la posición dentro de la tabla FAT para el primer clúster, siguiendo el proceso de la función **ClusNumToFAT** descrito en el diagrama de la figura 3.33.

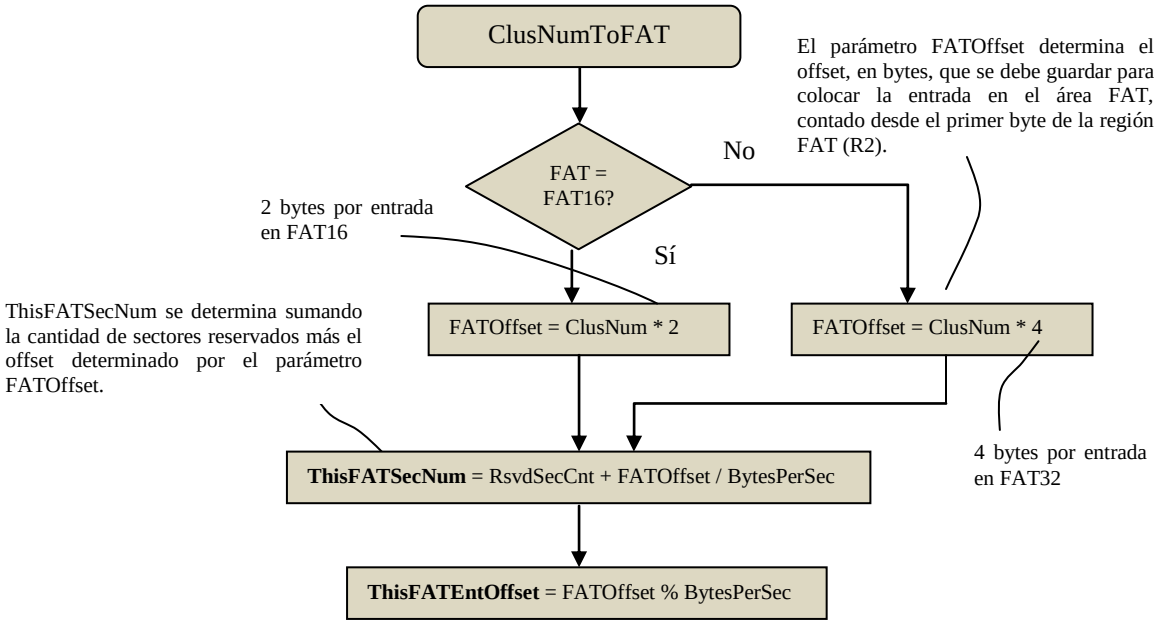


Figura 3.33. Determinación de la posición de la entrada de un clúster en la tabla FAT.

El parámetro *ThisFATSecNum* proporciona el número del sector donde se encontrará la entrada para el clúster de interés (*ClusNum*), en este caso el primero del archivo. *ThisFATEntOffset* es el número de byte dentro de este sector donde inicia la entrada.

El contenido de la entrada será el número del siguiente clúster del archivo en cuestión. Con base en la primera entrada se calcula la posición de la siguiente, de acuerdo al tipo de FAT, recordando que los clústers serán asignados de manera continua. La última entrada tendrá el valor 0xFFFF indicando el fin del archivo.

Las entradas se escribirán en el área FAT de la tarjeta SD utilizando la función de escritura de un bloque o de múltiples bloques, en el caso de que el número de entradas requiera más de un sector.

Una vez que se han escrito todas las entradas en la tabla FAT, el archivo estará listo para usarse. En este punto si la tarjeta es leída por una PC, verá un archivo con el nombre y tamaño asignados pero sin contenido.

Registro de datos dentro del área designada

El registro de datos consiste en adquirir la información que se desea almacenar, conformando bloques de 512 bytes y enviándolos a la tarjeta. Este proceso se compone de los siguientes pasos:

- Lectura de la entrada del directorio raíz: Tamaño del archivo y número de primer clúster.
- Conformar la dirección para el comando de escritura, a partir del número del primer clúster.
- Obtener la cantidad de sectores a registrar.
- Enviar el comando de escritura.
- Adquirir datos de la unidad sísmica y enviarlos a la tarjeta SD.

3.3.6. Comunicación con la unidad sísmica SR04

La recepción de los datos provenientes de la unidad sísmica SR04 y la transmisión de comandos hacia la misma se llevan a cabo a través de la salida serie del microcontrolador y haciendo uso de un transceptor y un conector DB9 (ver figura 3.7).

Para implementar las funciones que permiten la comunicación serie RS 232 entre el microcontrolador y la unidad SR04 se hará uso de la librería de funciones de MPLAB C18 para el manejo de la USART del PIC18F452.

La parte del *software* del microcontrolador que se refiere a la comunicación con la unidad sísmica tendrá dos propósitos principales:

- Adquirir los datos generados
- Enviar comandos

El puerto de comunicación serie de la unidad sísmica está configurado para una velocidad de transmisión de 34800 baudios, sin *Handshake*, con 8 bits de datos, 1 bit de paro y sin paridad. La configuración del puerto serie del microcontrolador se realiza utilizando la función ***OpenUSART ( )*** de la siguiente manera:

```
OpenUSART (USART_TX_INT_OFF      &
           USART_RX_INT_OFF      &
           USART_ASYNCH_MODE     &
           USART_EIGHT_BIT       &
           USART_CONT_RX         &
           USART_BRGH_LOW, SPBRG);
```

En esta función los parámetros del argumento tienen el siguiente sentido:

- USART_TX_INT_OFF: Interrupción por transmisión deshabilitada.
- USART_RX_INT_OFF: Interrupción por recepción deshabilitada.
- USART_ASYNCH_MODE: Modo de transmisión de la USART asíncrono.
- USART_EIGHT_BIT: Ocho bits de datos.
- USART_CONT_RX: Habilita la recepción continua de datos.
- USART_BRGH_LOW: Modo de transmisión a baja velocidad.
- SPBRG: Velocidad de transmisión.

El parámetro SPBRG (*USART Baud Rate Generator*) indicará la velocidad de transmisión, cuyo valor se calcula con la expresión 3.5:

$$\boldsymbol{SPBRG = \frac{Fosc}{64 * BaudRate} - 1} \qquad \textbf{Ec. 3.5}$$

En la expresión anterior, *Fosc* es la velocidad del reloj del sistema y *BaudRate* la velocidad a la que se desea trabajar con el puerto serie del microcontrolador, en nuestro caso requerimos 34800 baudios utilizando un cristal de 7.3728 MHz, por lo que *SPBRG* = 2.

Adquisición de los datos

El proceso de adquisición de los datos provenientes de la unidad sísmica hace uso de las funciones ***DataRdyUSART ( )*** y ***ReadUSART ( )***. La función ***DataRdyUSART ( )*** arroja el valor **'1'** si hay un dato disponible para leer en el buffer de la terminal de recepción serie del microcontrolador (Rx). La función ***ReadUSART ( )*** leerá el byte del buffer puerto serie, este valor deberá ser asignado a una variable para su posterior procesamiento. Adicionalmente, se debe verificar el bit de error por desbordamiento (*overflow*). Al detectar que el *buffer* se ha desbordado se tendrá que limpiar tanto el *buffer* como el bit que indica este error.

En la figura 3.34 se ilustra el proceso de adquisición de un bloque de 512 bytes que serán almacenados en el arreglo del mismo tamaño **Data [i]**.

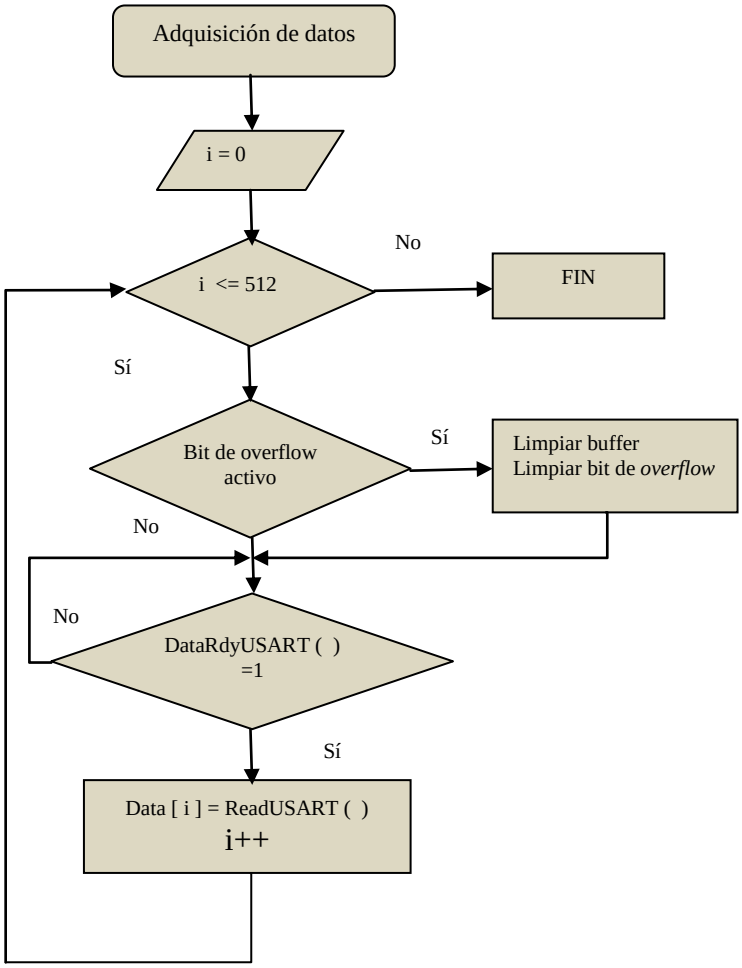


Figura 3.34. Diagrama de flujo para la adquisición de datos con el puerto serie del PIC18F452.

Decodificación de los datos

Una vez que se tiene una función para adquirir datos por el puerto serie del microcontrolador, se tiene la capacidad para almacenar y decodificar estos datos. En el caso de la interfaz propuesta, la decodificación de los datos se utilizará sólo para graficar el comportamiento de los sensores conectados a los canales de la tarjeta SADC20.

La decodificación de los datos que arroja la unidad SR04 se implementa utilizando el algoritmo descrito en la sección 2.4.3 del capítulo anterior. En el diagrama de la figura 3.35 y 3.36 se esquematiza el proceso para la decodificación del canal 1, que aplica para los dos canales restantes.

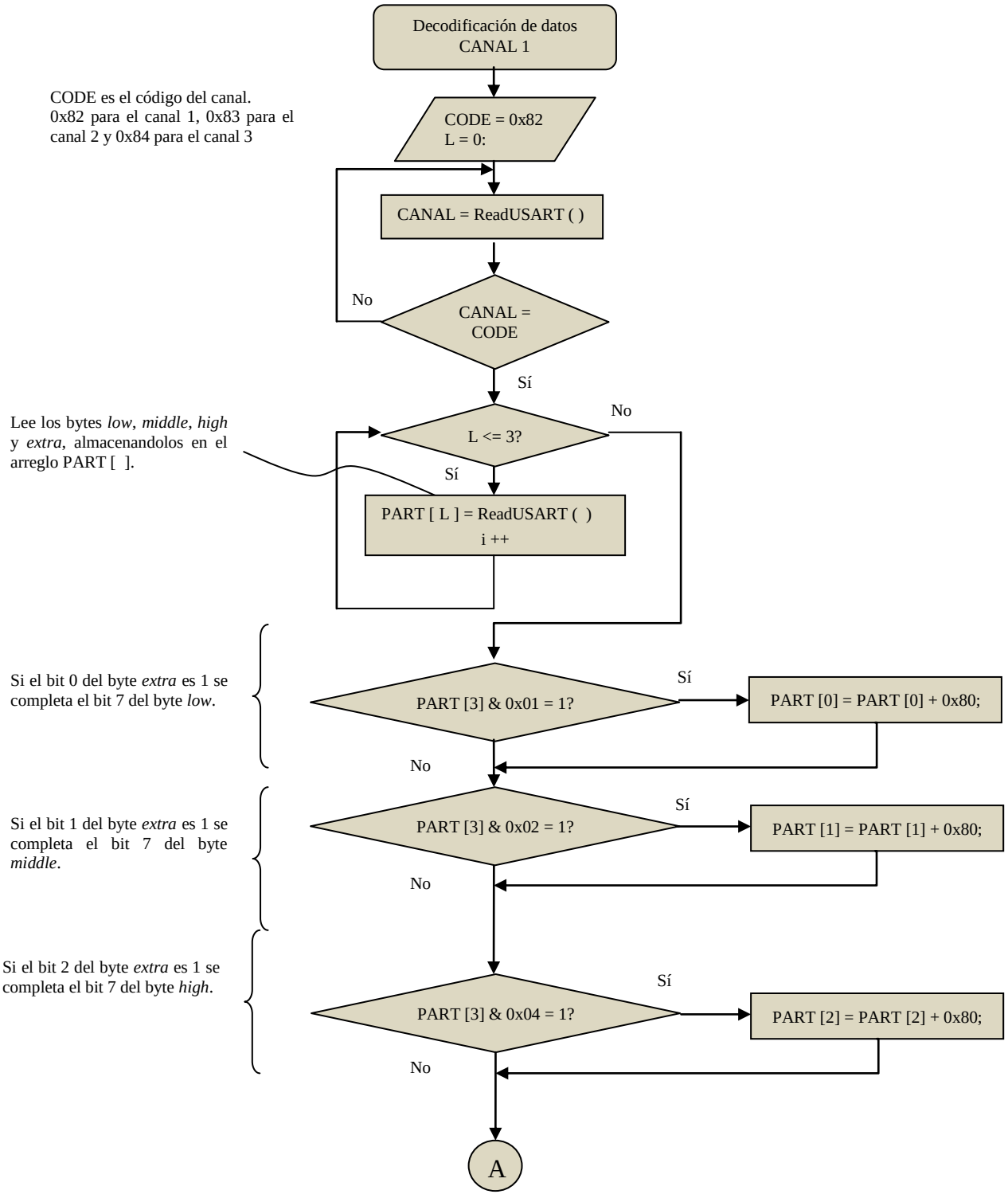


Figura 3.35. Diagrama de flujo para la decodificación de los datos del canal 1 de la unidad SR04 (1 de 2).

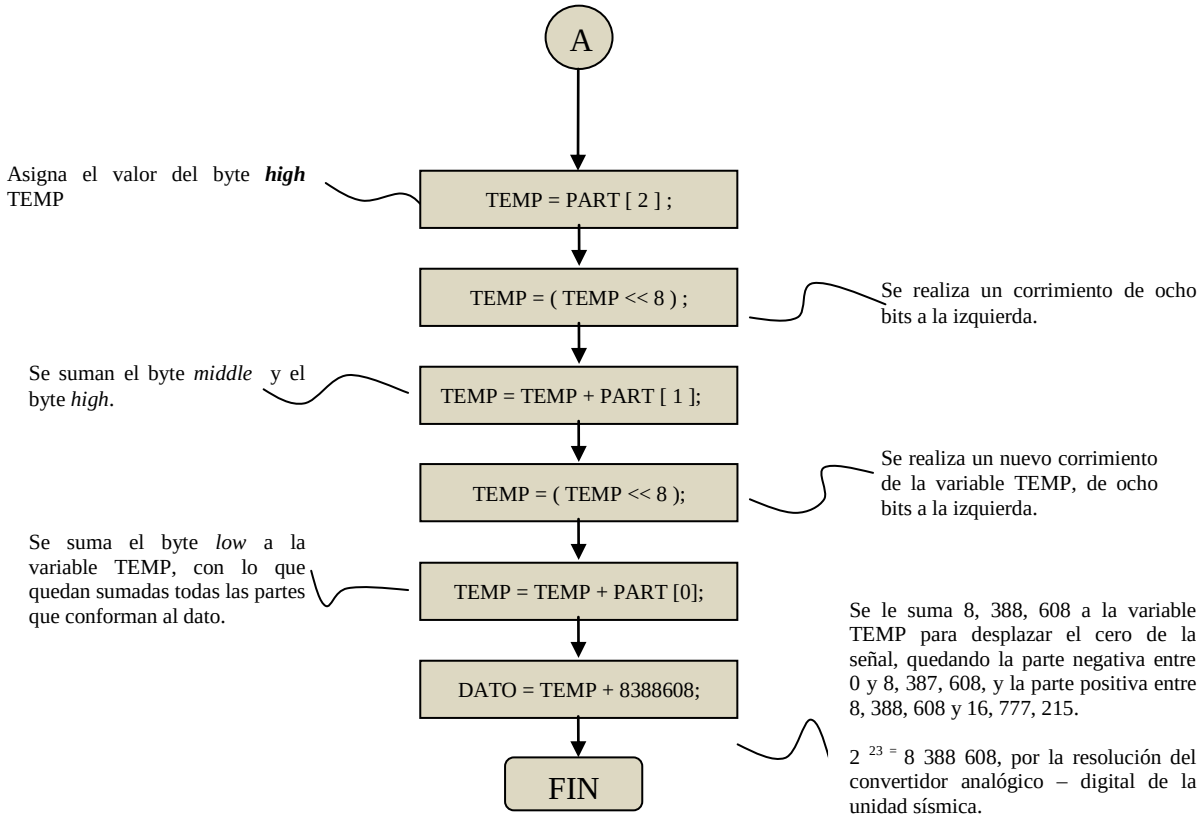


Figura 3.36. Diagrama de flujo para la decodificación de los datos del canal 1 de la unidad SR04 (2 de 2).

Comunicación entre la unidad sísmica y una PC por USB

Como se comentó anteriormente, la interfaz propuesta tendrá como una función secundaria el servir de enlace entre la unidad sísmica SR04 y un equipo de cómputo. En la figura 3.37 se muestra la conexión que se realiza entre la PC, la interfaz de registro y la unidad SR04.

En un principio y para los fines del presente trabajo, el enlace sólo servirá para configurar la frecuencia de muestreo de la mencionada unidad sísmica.

Al conectar el módulo UB232R a un equipo de cómputo, éste lo detectará como un dispositivo USB nuevo y buscará los controladores necesarios. Los controladores para esta aplicación son para crear un puerto COM virtual (*Virtual COM Port Drivers*), que permitirán manejar a este dispositivo USB como un puerto de comunicaciones COM. De esta manera se puede crear *software* para PC que maneje al módulo UB232R como si fuera un puerto serie estándar. En la figura 3.38 se muestran los mensajes que envía la computadora al conectar por primera vez al módulo UB232R y el nuevo puerto en el administrador de dispositivos.

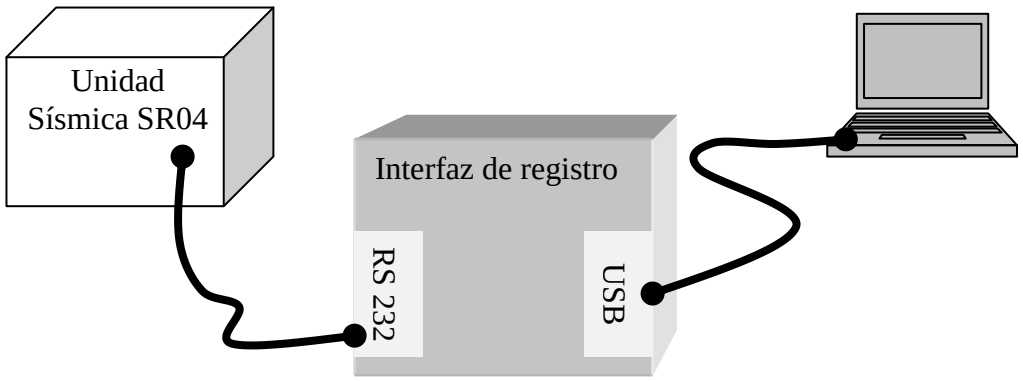


Figura 3.37. Conexión entre la Unidad SR04 la Interfaz de registro y una PC.

Figura 3.38. Instalación de los controladores del módulo UB232R en una PC.

El equipo de cómputo requiere, además, de una terminal virtual para visualizar la información que se genere en el puerto serie conectado a la interfaz de registro, en este caso es la llamada *RealTerm: serial capture*, que permite configurar y visualizar los puertos del equipo de cómputo.

Después de instalar los controladores del módulo USB, se podrá realizar la comunicación entre dicho módulo y la computadora, esto, a través de *RealTerm*, con el cual se configuran los parámetros de velocidad de transmisión, bits de datos y formato de visualización de los datos del puerto virtual, principalmente, ver figura 3.39.

Del lado del microcontrolador, la comunicación se realizará con el puerto serie del mismo, al cual se conecta el módulo UB232R. Las funciones necesarias para realizar la comunicación (*software*) requieren de la misma librería que se usa para la comunicación con el estándar RS – 232, es decir la librería *USART* de MPLAB C18.

En primer lugar, el microcontrolador envía un mensaje a la PC, donde pregunta si se desea configurar la frecuencia de muestreo, ver figura 3.40.

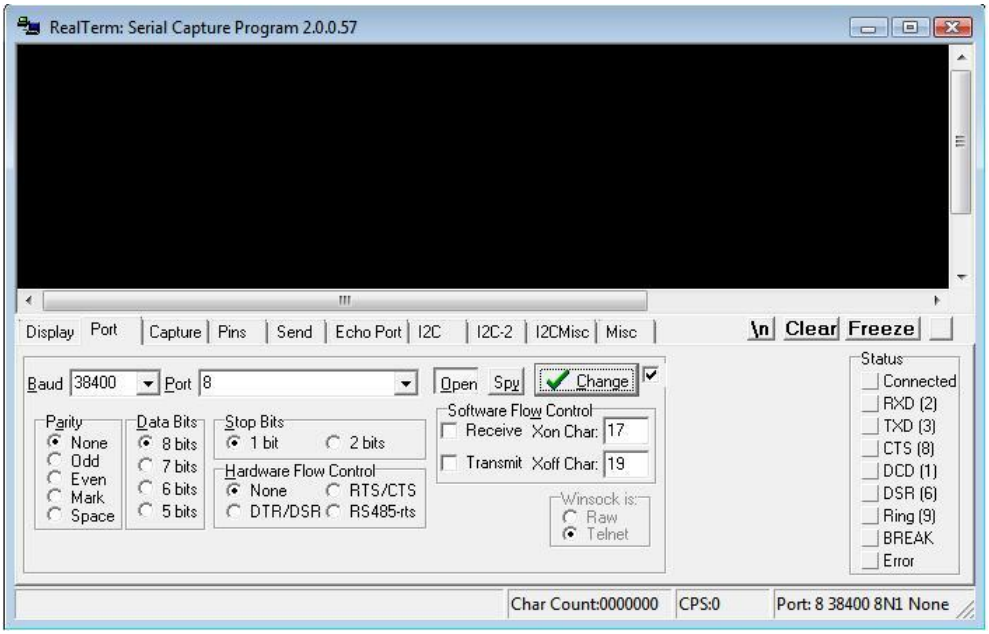


Figura 3.39. Panel de opciones del software RealTerm Serial Capture.

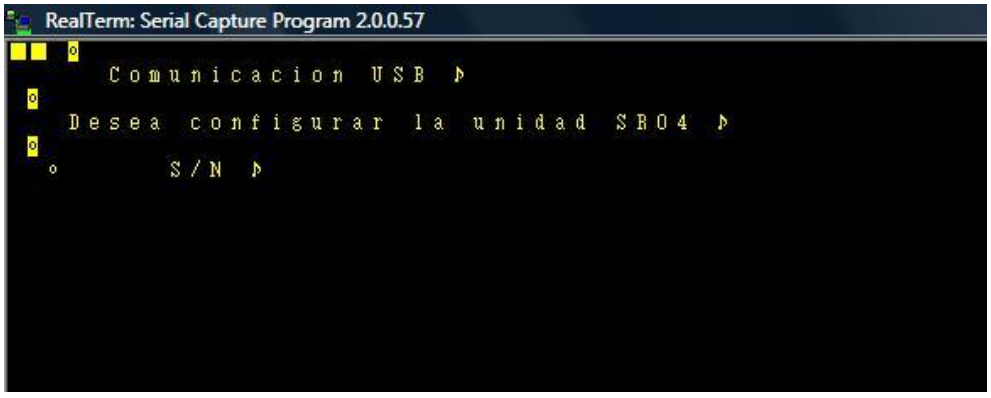


Figura 3.40. Mensaje inicial de la interfaz de registro hacia la PC por medio del adaptador USB.

La selección de las opciones que se mostrarán en los mensajes enviados, y que se pueden visualizar por medio de la terminal, se realiza pulsando la tecla correspondiente, en este primer mensaje se podrá pulsar **S** o **N**.

Si se pulsa S, el microcontrolador lo interpretará como un ***Sí*** y enviará como mensaje las opciones de frecuencia de muestreo, ver figura 3.41. De lo contrario, si se pulsa N el microcontrolador dará por terminada la comunicación con la computadora.

Cabe señalar que al pulsar una tecla diferente a las que corresponden a las opciones mostradas, se enviará un mensaje que indicará que la opción escogida no es válida y se enviarán de nuevo las opciones permitidas.

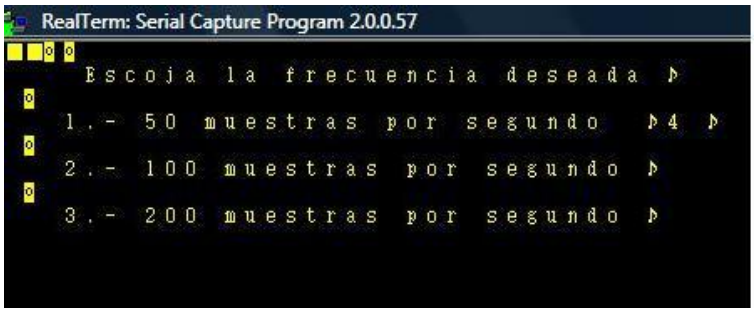


Figura 3.41. Opciones de frecuencias a configurar en la unidad SR04.

Para escoger una de las opciones mostradas en la figura 3.38 se debe pulsar las teclas 1, 2 ó 3 del teclado de la PC, correspondientes a las frecuencias de 50, 100 y 200 m.p.s, respectivamente. Al pulsar cualquiera de estas opciones la computadora está enviando al microcontrolador la orden para que se establezca la configuración de frecuencia con la opción elegida.

Al terminar de enviar el comando de configuración de frecuencia, el microcontrolador envía un mensaje a la PC, indicando que ya se estableció la frecuencia deseada, presentando, además, la opción para configurar de nuevo este parámetro, ver figura 3.42.

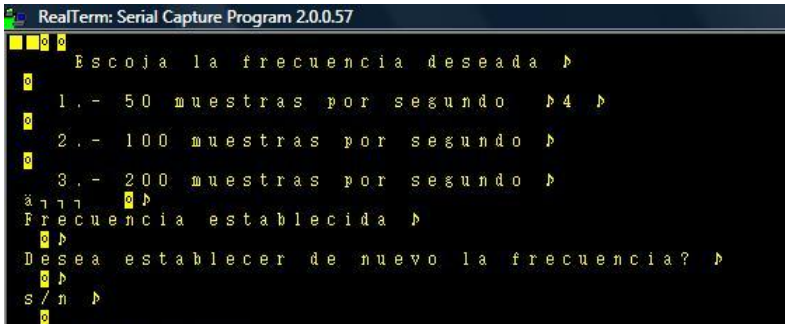


Figura 3.42. Mensaje posterior al ajuste de la frecuencia de muestreo de la unidad sísmica SR04.

Configuración de la frecuencia de muestreo

Conforme a lo dicho en la sección 2.4.3 del presente trabajo, en cuanto a los comandos aplicables a la tarjeta SADC20 de la unidad SR04, se desarrolla la función que genera el comando de configuración de la frecuencia de muestreo.

Recordamos la sintaxis del comando en cuestión:

0x84 sps1 sps2 sps3 0x00 0x00

En el comando anterior SP1, SP2 y SP3 se calculan con la expresión 3.6:

$$spsX = 200/(frecuencia\ requerida) \qquad \textbf{Ec. 3. 6.}$$

Las opciones de frecuencias de muestreo que se podrán configurar son: 50, 100 y 200 m.p.s. por lo cual los valores que tomará spsX serán: 4, 2 y 1, respectivamente.

En la figura 3.43 se muestra el diagrama de flujo para el envío del comando de configuración de la frecuencia de muestreo.

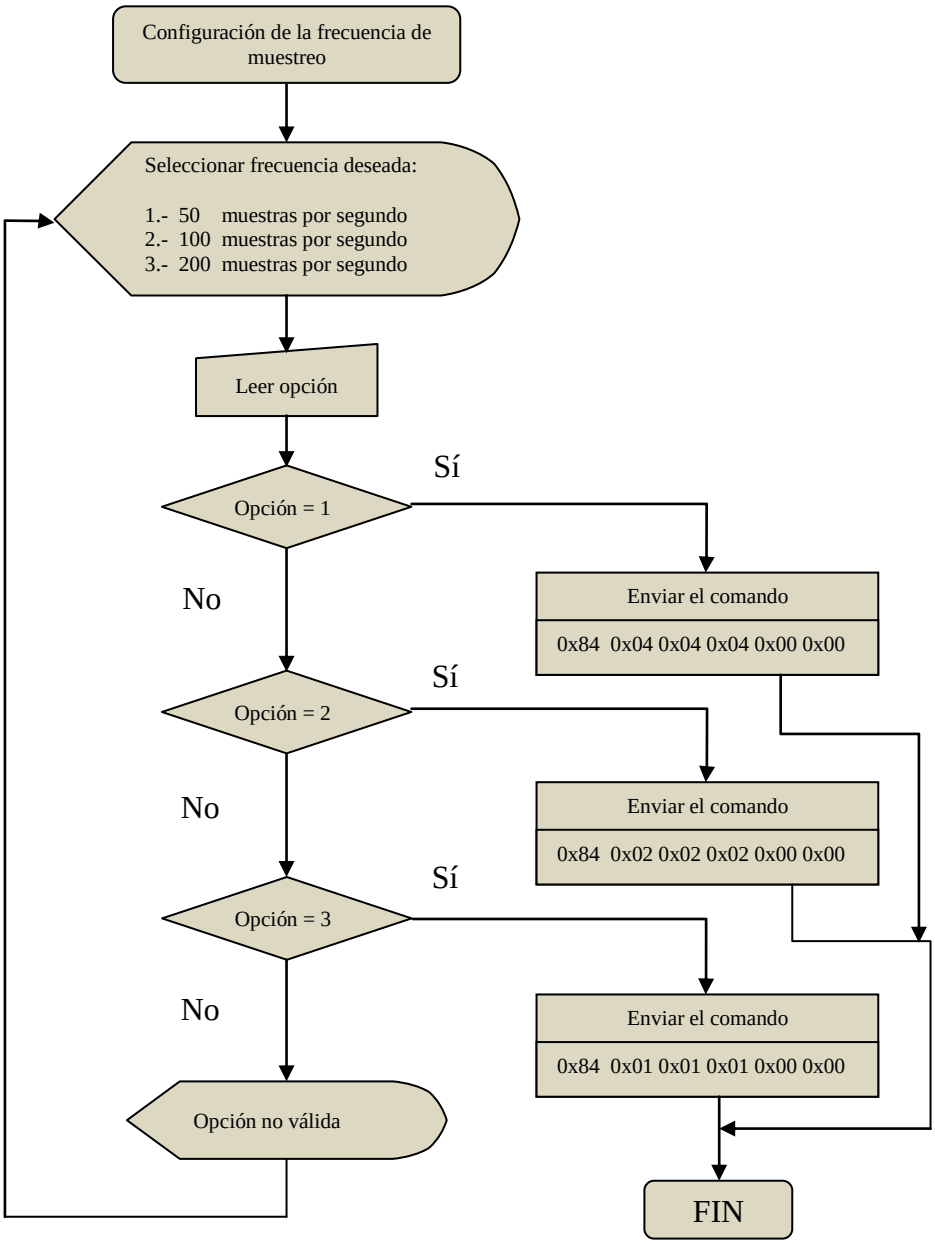


Figura 3.43. Diagrama de flujo para la función de configuración de la frecuencia de muestreo de la unidad sísmica SR04.

3.3.7. Manejo del teclado

El teclado estará compuesto por cinco teclas asignadas a igual número de terminales del puerto B del microcontrolador. La función de cada tecla se menciona a continuación:

- Terminal RB7: Tecla Entrar
- Terminal RB6: Tecla Arriba o Incrementar
- Terminal RB5: Tecla reservada para futuras actualizaciones
- Terminal RB4: Tecla Regresar
- Terminal RB3: Tecla Abajo o Disminuir

La detección de las teclas estará a cargo de una función que monitoreara las terminales del puerto B en donde se encuentran las teclas.

El efecto del rebote mecánico que se presenta en los interruptores que se usarán como teclado ha sido objeto de varias pruebas para determinar los tiempos de rebote. Por ejemplo, en un reporte hecho por el grupo *Ganssle*[2] , se puede observar que han obtenido una serie de gráficas y tablas en las que se reportan los tiempos de rebote para una gran cantidad de interruptores. Los tiempos en el que la salida de estos interruptores aún no se encuentra estable varían desde unos cuantos microsegundos hasta algunas decenas de milisegundos, teniendo como promedio 1557 µS y un máximo de 6200 µS.

Tomando en cuenta la información revisada de distintos estudios y pruebas realizadas a interruptores, en cuanto a los tiempos de rebote mecánico, para evitar este efecto en nuestro sistema, se pondrán algunos retardos a través de una subrutina descrita en la librería *delays.h*, propia de C18. El periodo de tiempo que será utilizado para descartar dicho rebote será de 25 ms, esto a partir de pruebas que se hicieron con los *switches* utilizados en la interfaz y considerando que el equipo está pensado para su uso en campo, donde los distintos factores ambientales podrían afectar el desempeño de estos dispositivos mecánicos

La función con la que se crearán los retardos es $Delay1KTCYx(N)$, en donde N es una cantidad en miles de ciclos de instrucción que corresponderán al retardo. N se calcula de acuerdo a las siguientes expresiones:

$$\boldsymbol{Ciclos = \frac{Retardo\ en\ mseg * Fosc}{4}} \qquad \boldsymbol{Ec. 3.7.}$$

$$\boldsymbol{N = \frac{Ciclos}{1000}} \qquad \textbf{Ec. 3.8.}$$

Como ya se comentó, utilizaremos un retardo de 25 ms para descartar los errores provocados por el efecto de rebote, por lo que se utilizará N = 46 en la función $Delay1KTCYx(N)$.

[2] http://www.ganssle.com/debouncing.htm

Las distintas opciones de selección se presentarán en el display, las cuales serán elegidas mediante el teclado. Dicha selección se hará moviendo una flecha o cursor que se posicionará en la opción seleccionada. Cada opción de cada menú será tendrá un valor único que será asignado a la variable ***opc***, que servirá de apuntador, y se tendrán otras dos variables cuyo valor estará determinado por la cantidad de opciones presentadas en pantalla y al valor de ***opc***. Por ejemplo, si al entrar a la función ***Teclado (opc, lo, hi)*** con $\boldsymbol{opc = 1}$ (en el caso del menú principal) y si se tienen tres opciones a elegir, tendremos $\boldsymbol{lo = 1}$ y $\boldsymbol{hi = 3}$ como argumento. Con la función mensaje se muestra una pantalla en la que el cursor señala a la opción con valor igual a ***opc.*** Al pulsar la tecla *arriba* se disminuirá en 1 el valor de ***opc,*** y si se pulsa la tecla *abajo* se incrementara en 1 el valor de la misma variable. Estas operaciones tendrán como límite superior e inferior a las variables ***hi*** y ***lo*** respectivamente, de manera que la selección siempre estará dentro de las opciones admitidas para el menú actual. En la figura 3.44 se presenta el diagrama de flujo para el manejo del teclado.

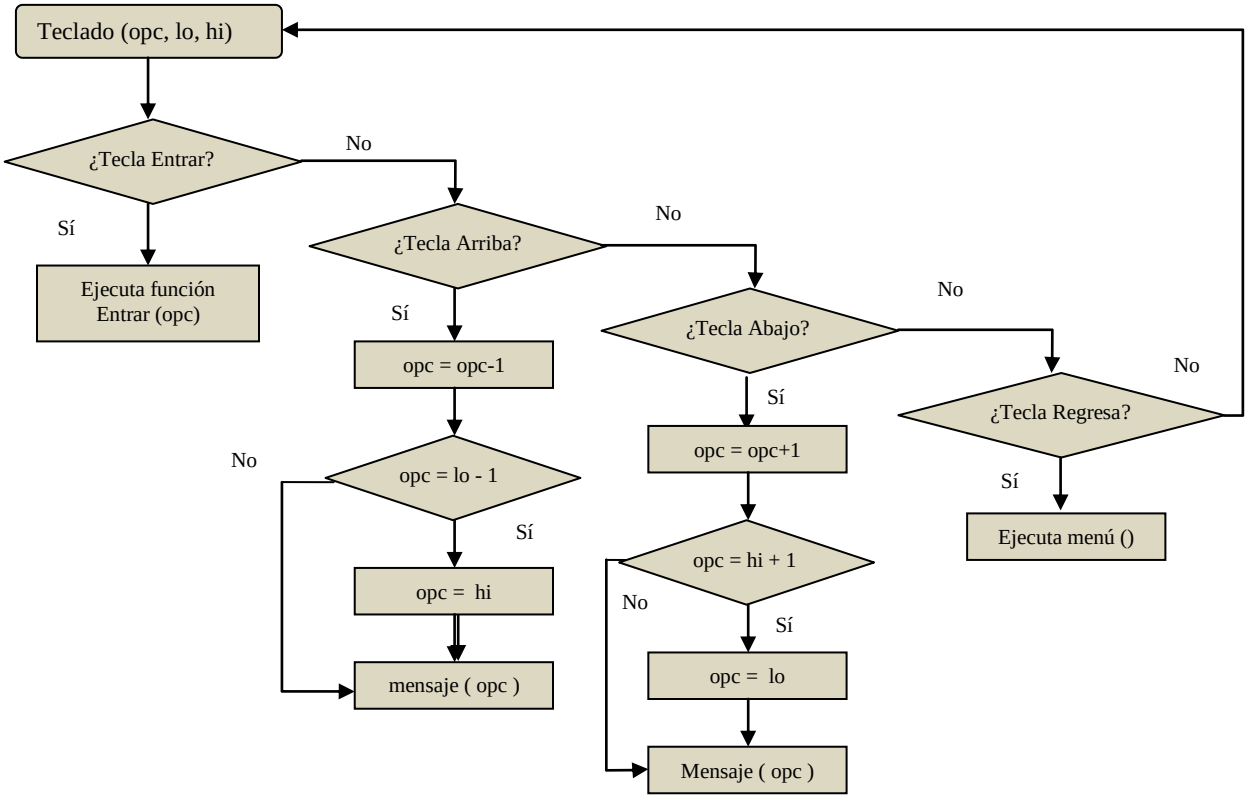


Figura 3.44. Diagrama de flujo del manejo del teclado.

3.3.8. Manejo del display gráfico

Para controlar al display gráfico se hará uso de una librería de funciones para *displays* gráficos basados en el controlador KS0108. La librería cuenta con funciones que están escritas en C y que son compatibles con el compilador MPLAB C18. Las funciones programadas más destacadas en dicha librería se describen a continuación:

- Inicialización del display: la función ***GDispInit***( ) inicializa al display gráfico por lo que deberá ejecutarse antes de cualquier otra. Consta de los siguientes pasos:

  A. Configurar el puerto asignado al display
  B. Enviar la señal Restablecer (*RESET*)
  C. Deshabilitar el display
  D. Poner la Línea de Inicio (*Start Line*) en la parte superior de la pantalla
  E. Limpiar el display escribiendo ceros en toda la pantalla
  F. Habilitar de nuevo el display

- Habilitación del display: la función ***GDispSwitch***( ) enciende o apaga el display enviando la señal de habilitación.

- Escribir un byte: Por medio de la función ***GDispWrByte***( ) se puede enviar un byte a la columna y página deseada. Las columnas se cuentan de izquierda a derecha del display, de 0 a 127. Las páginas se cuentan de arriba hacia abajo de 0 a 7, cada página consta de 8 puntos.

- Limpiar el display: con la función ***GDispClr***( ) se envía el byte cero (0x00) a todas las páginas de todas las columnas, que tiene como efecto el de *apagar* todos los pixeles. Esta función se puede modificar para *encender* todos los puntos de la pantalla enviando el byte 0xFF como argumento.

- Escribir cadena en el display: a través de la función ***GDispPixStrAt***( ) se convierte una cadena de caracteres a un patrón de pixeles que son *encendidos* en la pantalla. Para usar esta función se requiere usar un archivo donde estén definidos los caracteres a partir de su código ASCII. Adicionalmente se puede indicar si se los puntos habrán de encenderse o si se apagarán para crear el contraste deseado.

- Escribir carácter: Para escribir un único carácter en la pantalla del display se ejecuta la función ***GDispCharAt***( )**,** que tiene como argumento el código ASCII de dicho carácter.

- Configurar el puerto del display: con el fin de asignar y configurar las terminales del microcontrolador que servirán para manejar el display, se ejecuta la función ***GDispInitPort***( )**.**

- Seleccionar parte izquierda/derecha: el display *JHD12864G/J* cuenta con dos secciones de pantalla, mismas que son habilitadas con la función ***GDispChipSel***( )**.**

A partir de estas funciones básicas se pueden crear otras que facilitarán la creación de figuras, gráficas y menús que permitirán al usuario visualizar procesos e interactuar con el microcontrolador y con otros dispositivos periféricos. En la figura 3.45 se tiene un ejemplo de la utilización de la librería de este display para mostrar un dibujo.

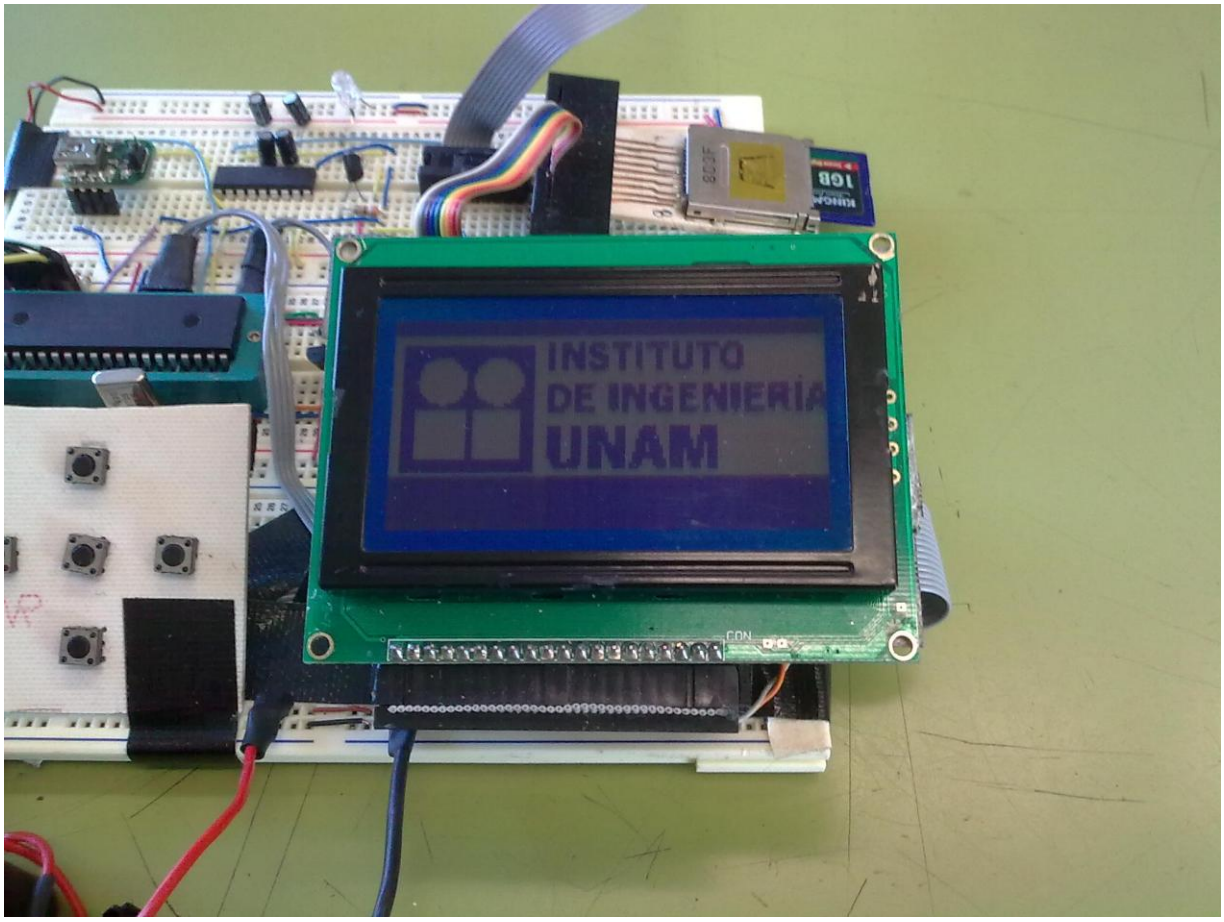


Figura 3.45. Visualización de gráficos en el display.

Creación de los menús

Un menú se construye con una serie de líneas de texto, mismas que serán creadas con la función ***GDispPixStrAt*( ).** El argumento de esta función será la cadena a mostrar y la posición del display donde se desea escribir dicha línea. En la figura 3.46 se presenta el código de una pantalla de menú con tres opciones, a manera de ejemplo.

```
int menu(void) //================================================================
{
    int opc = 1, hi=3, lo=1;
    EnablePullups();
    TRISB = 0xF8;       // Inicializa puerto B: teclado, LED indicador y Terminal CS del display
    GDispClr(0x00);     // Limpia la pantalla
    osport( );          // Configura USART
    GDispPixStrAt(0,0,"      MENU    ",3,1);
    GDispPixStrAt(0,10,"Especifique la opcion:",0,1);
    GDispPixStrAt(0,25,">>Configuracion   ",0,1);
    GDispPixStrAt(0,35,"  Modo de prueba   ",0,1);
    GDispPixStrAt(0,45,"  Registro   ",0,1);
    teclado(opc,hi,lo); // Llama a la subrutina que lee el teclado
}
```

Figura 3.46. Ejemplo del código para mostrar una pantalla de menú.

Como se mencionó anteriormente, la pulsación de la tecla adecuada nos servirá para mover un cursor que indicará la opción seleccionada. Para lograr este efecto se escribió una función que dibujará el cursor (>>) en una posición asociada a la variable ***opc***, cuyo valor cambiará al pulsar las teclas *arriba* o *abajo* y borrará el cursor de su posición anterior. La función mencionada es ***mensaje***(***opc***), que consiste en un estructura de control *switch – case*, en la que se definirán las posiciones del cursor. Siguiendo al ejemplo anterior, en la figura 3.47 se muestra parte del código de la función ***mensaje***(***opc***) para el menú mostrado en la figura 3.46.

```
int mensaje(int opc)
{
    switch(opc)
            {
            case 1:
                            //  0 corresponde a la posicion en Y
                           //// 25 corresponde a la posicion en x
                          ////        //-----> Byte de color: 1 - Encendido; 0 - apagado
            GDispPixStrAt(0,25,">>",0,1);  // Flecha encendida
            GDispPixStrAt(0,35,">>",0,0);  // Flecha apagada
            GDispPixStrAt(0,45,">>",0,0);  // Flecha apagada
            opc=1;
            break;

            case 2:
            GDispPixStrAt(0,25,">>",0,0); // Flecha apagada
            GDispPixStrAt(0,35,">>",0,1); // Flecha encendida
            GDispPixStrAt(0,45,">>",0,0); // Flecha apagada
            opc=2;
            break;

            case 3:
            GDispPixStrAt(0,25,">>",0,0); // Flecha apagada
            GDispPixStrAt(0,35,">>",0,0); // Flecha apagada
            GDispPixStrAt(0,45,">>",0,1); // Flecha encendida
            opc=3;
            break;
```

Figura 3.47. Extracto del código de la función mensaje ( ).

El resto de los menús están creados siguiendo la misma lógica. Los menús creados y sus opciones se presentan en el diagrama de flujo de la figura 4.48, que se encuentra en la siguiente página.

Selección de opciones a través de menús

De acuerdo a las necesidades planteadas al principio de este capítulo, se diseñará una serie de pantallas que conformarán un menú. A través del uso de la función ***teclado,*** descrita anteriormente, se podrá seleccionar entre las distintas opciones mostradas.

Al ejecutar el programa, el sistema presenta un menú principal con las siguientes opciones:

- **Configuración**
- **Modo de prueba**
- **Registro**

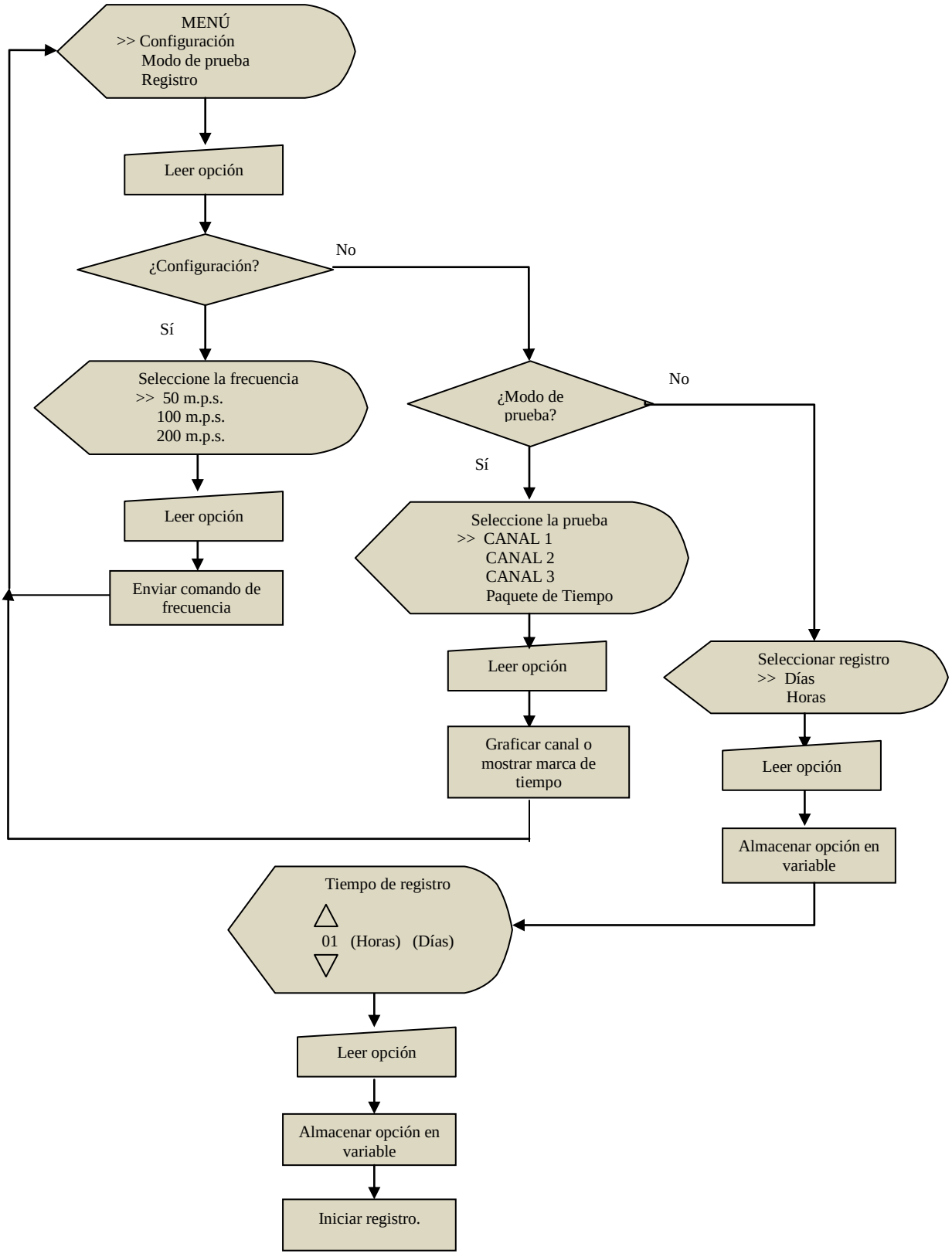


Figura 3.48. Diagrama de flujo de los menús mostrados en el display gráfico.

Configuración

En la opción de **Configuración** se presentará otro menú en el cual se mostrarán las frecuencias más utilizadas en las pruebas que se realizan con la unidad SR04. Estas frecuencias son 50, 100 y 200 muestras por segundo (mps). En la figura 3.49 se presenta una captura de pantalla con un extracto del código que se escribió para desplegar el menú de configuración y, en la figura 3.50 se muestra la pantalla que se genera en la Interfaz de registro.

```
//==== Menu 3 ==== AJUSTE FRECUENCIA ====================
char menu3(int opc)
{
    int  hi=12, lo=10;
    GDispClr(0x00);
    GDispPixStrAt(0,0,"Seleccione la frecuencia:",0,1);
    GDispPixStrAt(0,25,"  50 m.p.s.   ",1,1);
    GDispPixStrAt(0,35,">>100 m.p.s.   ",1,1);
    GDispPixStrAt(0,45,"  200 m.p.s.   ",1,1);
    mensaje(opc);
    teclado(opc,hi,lo);
    return ;
}
//=========================================================
```

Figura 3.49. Extracto del código del menú de ajuste de frecuencia de muestreo.

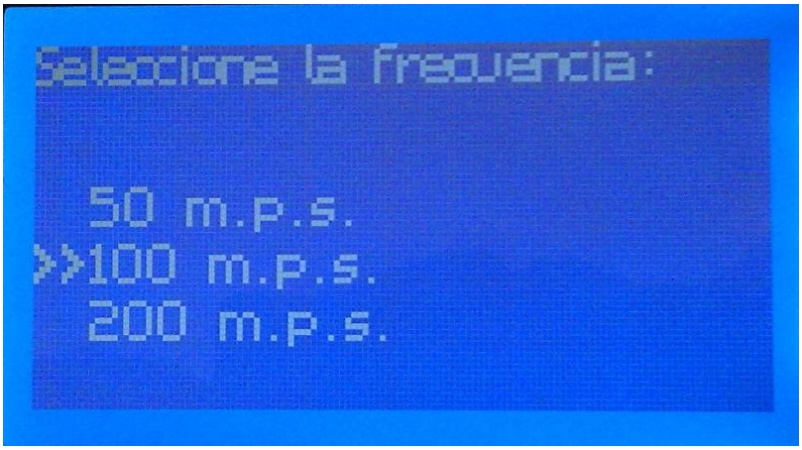


Figura 3.50. Pantalla del menú de ajuste de frecuencia de muestreo.

Una vez seleccionada la opción deseada se deberá pulsar la tecla ENTRAR, con lo cual el microcontrolador enviará el comando de configuración de frecuencia estableciendo este parámetro con el valor indicado, ver figura 3.43. Habiéndose realizado lo anterior, en el display se mostrará durante 2 segundos el mensaje “Frecuencia establecida” y al terminar este periodo regresará al menú de configuración de frecuencia, pero ahora el cursor indicará la frecuencia establecida.

Modo de Prueba

En la opción **Modo de Prueba** el usuario podrá seleccionar entre los canales 1, 2 y 3, para ver su comportamiento, uno a la vez, o ver la marca de tiempo compuesta por la fecha y la hora GMT.

En la figura 3.51 se observa en la interfaz de registro el mensaje mostrado al seleccionar la opción Modo de Prueba. El usuario podrá escoger entre graficar el comportamiento de los canales 1, 2 y 3, en los que están conectados los geófonos, o mostrar la marca de tiempo del GPS, de este último incluye la fecha y la hora GMT.

Figura 3.51. Menú de opciones en el modo de prueba.

Al elegir cualquiera de los canales, como ya se comentó, se mostrará en la pantalla la gráfica del comportamiento del mismo.

Para mostrar las gráficas mencionadas se realizó un programa consistente en los siguientes pasos:

- Esperar marca del canal deseado
- Almacenar los bytes correspondientes a la muestra
- Decodificar la muestra obtenida
- Realizar corrimiento
- Mostrar el punto obtenido

Cabe señalar que, al tener el display 128 puntos de ancho, se graficarán igual número de muestras y cada una de ellas pasará por el proceso anterior. Los primeros tres pasos de dicho proceso se realizan conforme al diagrama presentado en las figuras 3.33 y 3.34, en el apartado que se refiere a la decodificación de datos de la unidad sísmica SR04.

Para graficar la amplitud de la señal proveniente de cualquiera de los canales, el display utilizado cuenta con 64 puntos a lo alto, en donde 32 puntos corresponden a la parte positiva y el resto para la parte negativa de la señal. Dado que el dato de la muestra de la señal que genera la unidad sísmica es de 24 bits, se debe realizar un corrimiento de 18 bits a cada dato obtenido, de manera que se pueden trazar gráficas claras con la parte más significativa de dicha señal dentro de los 64 puntos del display.

La señales graficadas tienen como propósito verificar el buen funcionamiento de la unidad sísmica, es decir, si la conexión de los sensores es correcta y si se está recibiendo la señal del GPS.

En la figura 3.52 se muestra, a manera de ejemplo, una gráfica del comportamiento del canal 1.

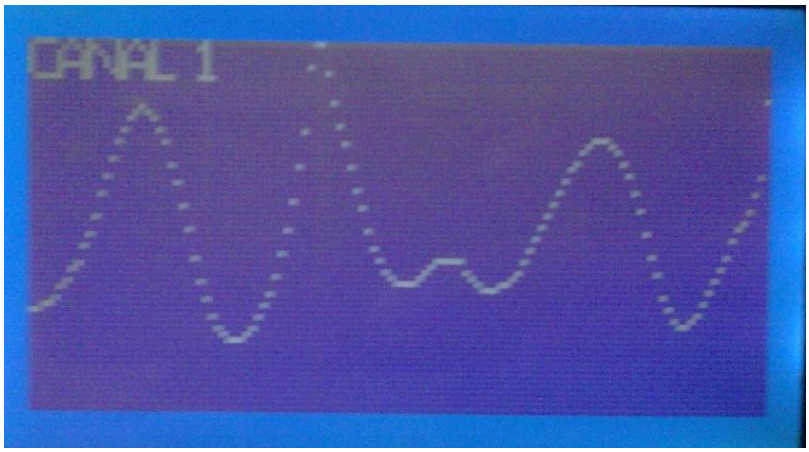


Figura 3.52. Gráfica del comportamiento del Canal 1 de la unidad sísmica SR04.

En cuanto a la señal del GPS, hay que recordar que el paquete de tiempo que envía la unidad SR04 es de 9 bytes, en donde el primero de estos corresponde al identificador (0x81 hexadecimal) y los restantes a la fecha, hora, extra y fin, de la siguiente manera:

0x81 año mes día seg min hora extra 0xFF (Paquete de tiempo)

En el caso del byte de año, para obtener el valor correcto se le debe sumar 2000 a la cantidad que indica éste. Por ejemplo, si el byte de año tiene un valor de 10, al sumarle 2000 obtendremos 2010, lo que indica que el año actual es 2010. El byte de mes tendrá un valor entre 1 y 12, en donde 1 es Enero y 12 es Diciembre. El byte de segundos (seg) tendrá un valor entre 0 y 59. El byte de minutos (min), al igual que el de segundos, tendrá un valor entre 0 y 59 y, por último, el byte de hora tendrá un valor entre 0 y 23. El byte *extra* no tiene utilidad por lo que será ignorado.

Al seleccionar Paquete de tiempo en el Modo de Prueba de la interfaz de registro, el microcontrolador esperará recibir el byte identificador hexadecimal 0x81, una vez que se recibe se almacenarán únicamente los siguientes 6 bytes, dado que no se utilizará, para esta aplicación, el byte extra.

La información del paquete de tiempo obtenido se mostrará en el display grafico, la fecha en un formato de texto, es decir, se imprimirá en la pantalla en un formato de *Mes/ Día/ Año*. Adicionalmente, se mostrará el mensaje 'Hora GMT' seguido de la hora en un formato de *Horas: Minutos: Segundos*. En la figura 3.53 se muestra un ejemplo de una pantalla con la información del paquete de tiempo.

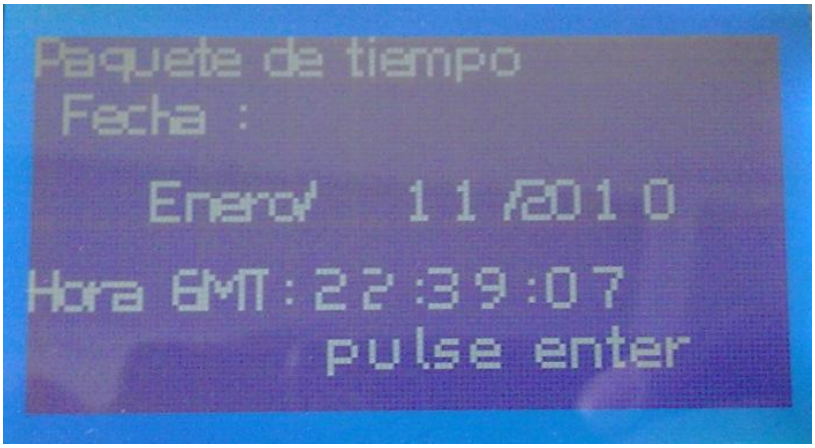


Figura 3.53. Paquete de tiempo del GPS de la unidad SR04.

Modo de Registro

Al seleccionar la opción **Registro** en el menú principal, la pantalla de la interfaz mostrará un mensaje indicando que se debe insertar la tarjeta de memoria SD y a continuación pulsar la tecla entrar. Realizadas estas acciones, en el programa del microcontrolador se llama a la subrutina que se encarga de inicializar la tarjeta de memoria SD en el modo bus SPI. Si la tarjeta es inicializada correctamente llamará a la siguiente subrutina, la cual tiene como tarea determinar el sistema de archivos con el que cuenta dicha tarjeta. El sistema de archivos detectado es mostrado en la pantalla, como se puede observar en la figura 3.54.

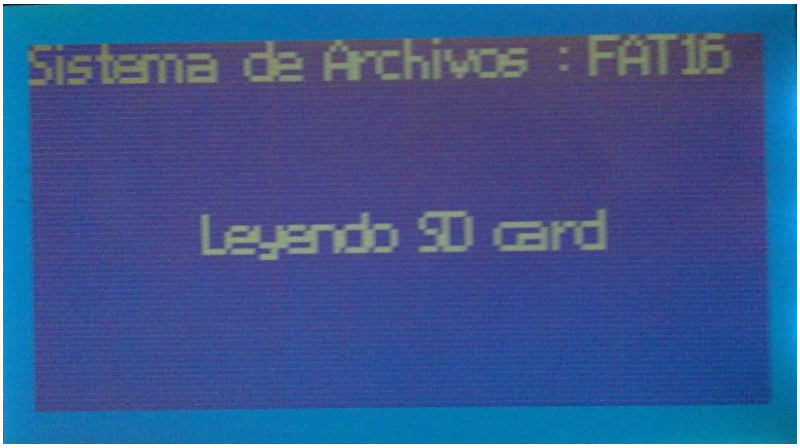


Figura 3.54. Sistema de archivos FAT16 detectado en el modo de registro.

Si en el proceso de inicialización se detecta que la tarjeta de memoria no es compatible, debido a que no soporta las especificaciones técnicas en su versión 2.0, se detendrá el proceso en cuestión y se indicará, a través del display, que la tarjeta insertada no es una tarjeta válida y el usuario deberá pulsar entrar para regresar al menú principal.

Cuando la tarjeta de memoria es válida y el sistema de archivos es FAT16 ó FAT32 se continuará con la presentación de un nuevo menú, en el cual se podrá seleccionar entre un registro en periodos de días y un registro en periodos de horas, como se muestra en la figura 3.55.

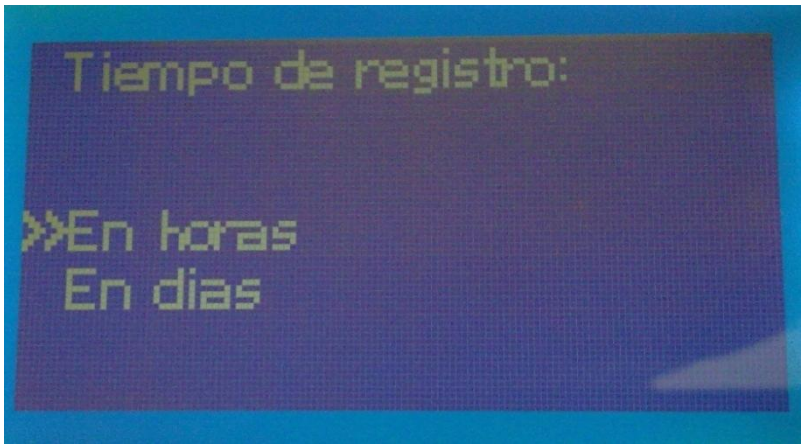


Figura 3.55. Opciones para el periodo de registro.

Al elegir un registro por horas el parámetro ***segs*** de la expresión 3.3, presentada y explicada en un apartado anterior, toma el valor de 3,600. Por otro lado, si se opta por un registro en días dicho parámetro tomará el valor de 86,400. Una vez seleccionado el periodo requerido, la Interfaz de registro presentará una pantalla en la que el usuario indicará la cantidad de horas o días que desea registrar los datos provenientes de la unidad SR04. Se podrá incrementar el número mostrado en pantalla (hasta 255 horas ó hasta 30 días), con las teclas 'Incrementar' o 'Disminuir', hasta llegar al tiempo de registro necesario, donde deberá pulsar la tecla *entrar* para hacer efectiva la selección e iniciar el registro de los datos.

Después de especificar la cantidad de tiempo a registrar, al pulsar la tecla *entrar* se le indicará al microcontrolador de la Interfaz que almacene en la memoria RAM el valor correspondiente, además, se mostrará el menú de Selección de Frecuencia de muestreo. Con el valor de este último parámetro se puede aplicar la mencionada expresión 3.3 para determinar la cantidad de bytes a registrar.

3.3.9. Registro de los datos provenientes de la unidad sísmica en las memorias SD

El proceso de creación de archivos y registro de datos, como se ha comentado con anterioridad, requiere de una serie de pasos, que en resumen incluyen: el registro de las entradas en el directorio raíz para cada archivo, asignación del espacio requerido para cada archivo a través de las entradas en la tabla FAT y por último, registro de la información en el espacio asignado en la región de datos.

El procedimiento inicia al leer la cantidad de bytes a registrar, que dependerá de la configuración hecha por el usuario. Considerando que la información almacenada en las tarjeras SD será procesada en un equipo de cómputo y que dicho procesamiento depende de los recursos de memoria con los que cuente el equipo mencionado y las pruebas que se realizaron con la interfaz, se establecerá como tamaño máximo de los archivos la cantidad de 260,755,200 Bytes (aproximadamente 248 Mega Bytes).

A partir del tamaño máximo de los archivos y de la cantidad de bytes a registrar, se calculará la cantidad de archivos que se crearán. La operación es simple, como se muestra en el diagrama de la figura 3.56.

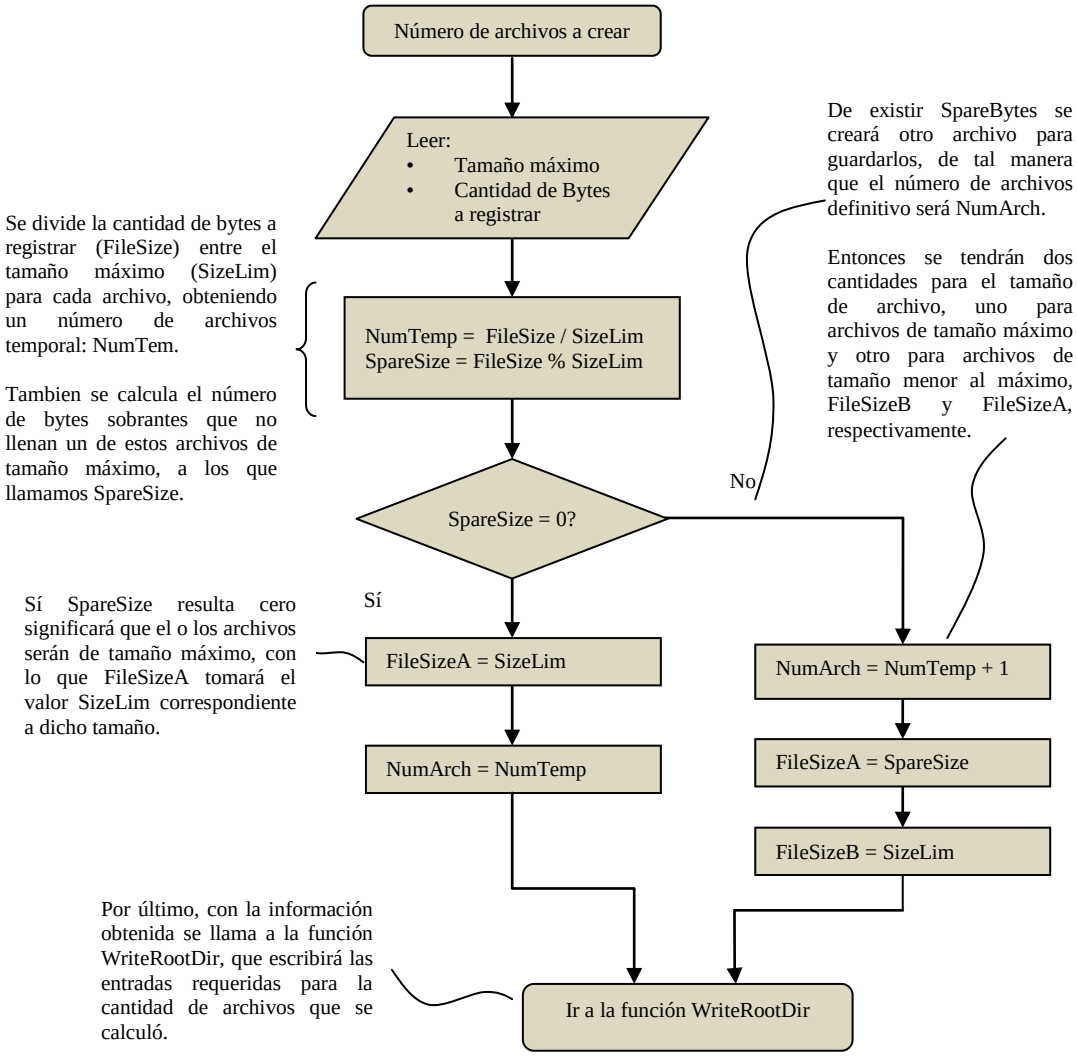


Figura 3.56. Determinación de la cantidad de archivos que se crearán.

Conociendo la cantidad de archivos y el tamaño de éstos, el programa del microcontrolador llama a la subrutina *WriteRootDir* que escribirá las entradas correspondientes en el directorio raíz de la tarjeta de memoria. Las figuras 3.57 a 3.59 muestran el diagrama para la función mencionada.

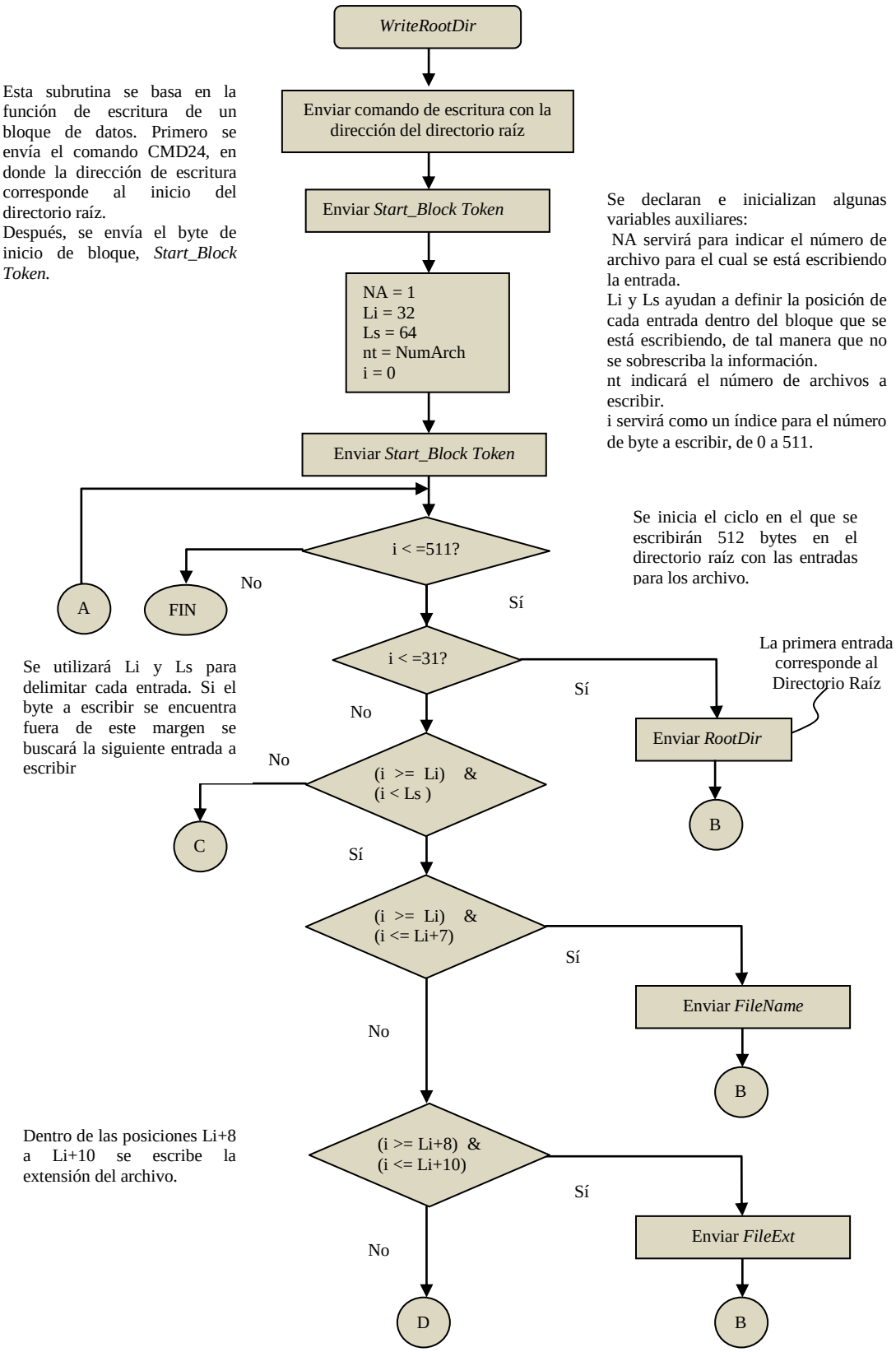


Figura 3.57. Diagrama de la función WriteRootDir (1 de 3).

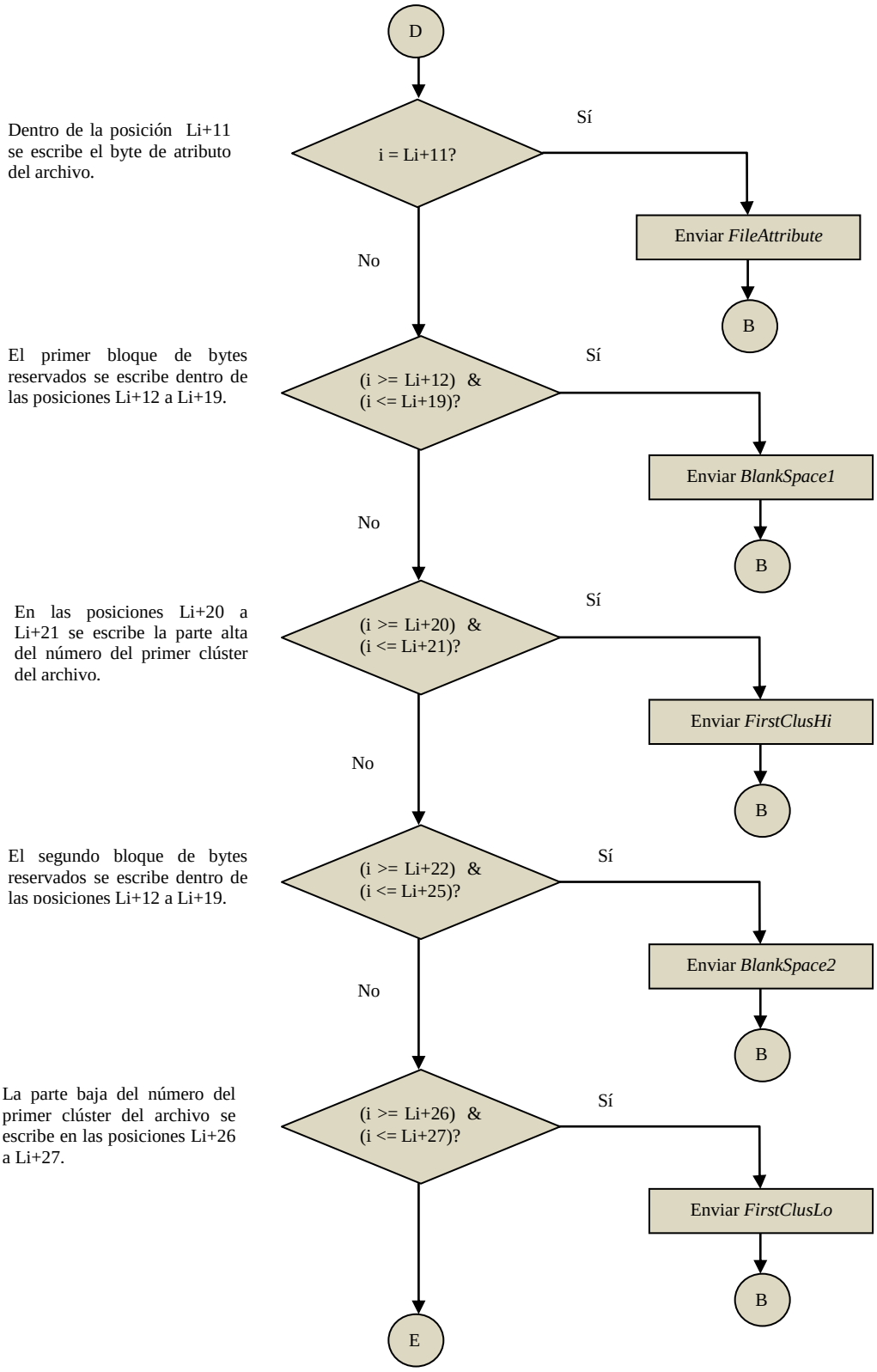


Figura 3.58. Diagrama de la función WriteRootDir (2 de 3).

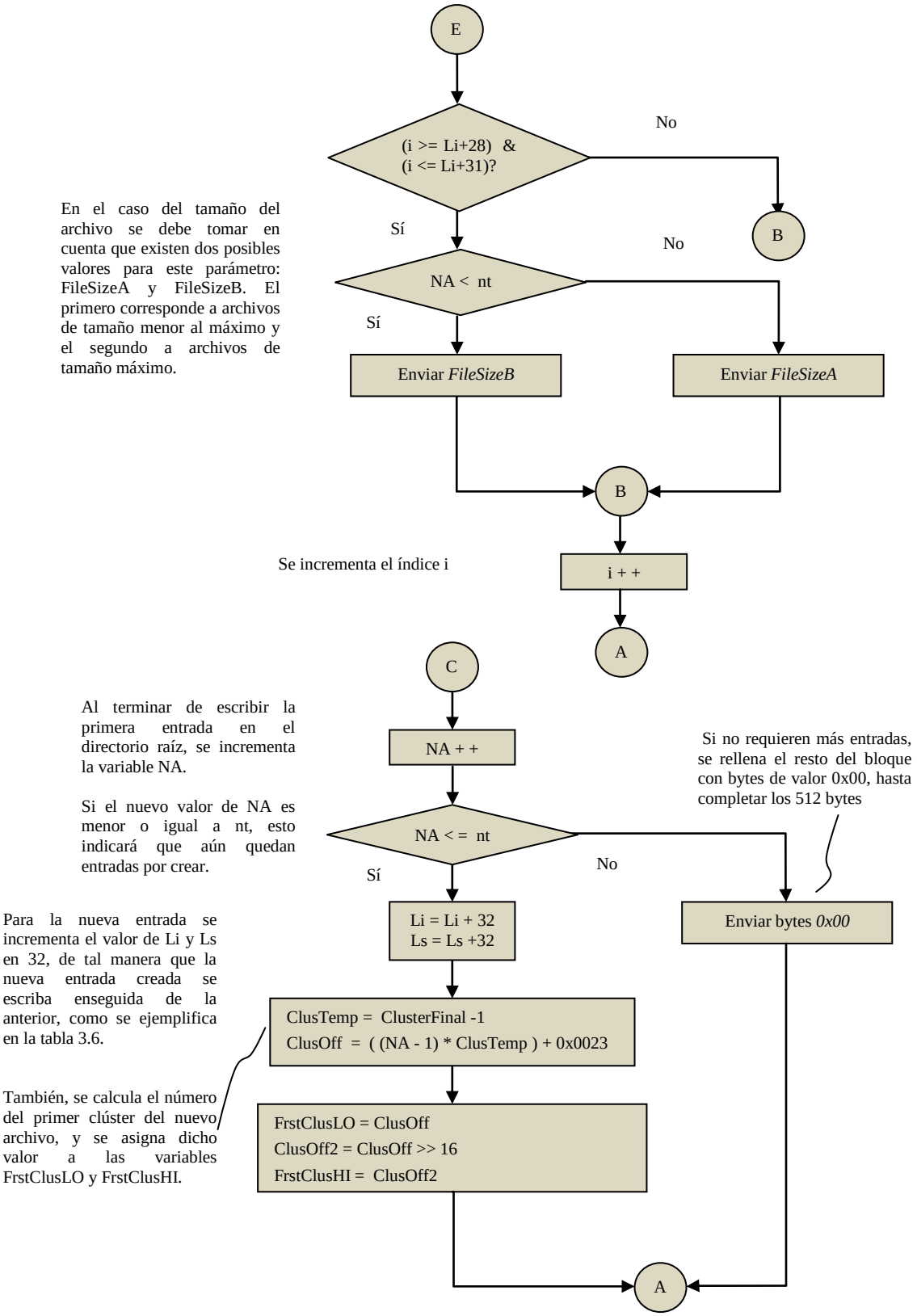


Figura 3.59. Diagrama de la función WriteRootDir (3 de 3).

Al entrar en la función anterior, el programa tiene en memoria la cantidad de archivos a escribir y el tamaño correspondiente. Se utilizan las variables Li y Ls para escribir cada entrada en donde le corresponde de acuerdo a la tabla 3.6.

Nombre	Inicia en el byte (Li)	Termina en el byte (Ls)
Registro 0	0	31
Registro 1	32	63
Registro 2	64	95
Registro 3	96	127
Registro 4	128	159
Registro 5	160	191
Registro 6	192	223
Registro 7	224	255
Registro 8	256	287
Registro 9	288	319
Registro 10	320	351
Registro 11	352	383
Registro 12	384	415
Registro 13	416	447
Registro 14	448	479
Registro 15	480	511

Tabla 3.9. Bytes de inicio y fin para las distintas entradas en el directorio raíz.

Si la cantidad de bytes a registrar requiere crear más de un archivo, se deberá calcular el número del primer clúster para dichos archivos, recordando que se estableció como número del primer clúster del primer archivo al clúster 35.

Considerando que el registro de los datos obtenidos de la unidad sísmica SR04 debe realizarse sin pérdidas, los archivos se escribirán de forma adyacente, es decir, el primer clúster del archivo 2 se encontrará un clúster adelante del último clúster del archivo 1, y de manera similar para los siguientes archivos.

Para calcular el número del primer clúster de los archivos posteriores al primero, se utiliza el siguiente algoritmo:

$$ClusTemp = CntClusLim$$
$$ClusOff = ((NA - 1) * ClusTemp) + 0x0023$$
$$FrstClusLO = ClusOff$$
$$Clusoff = ClusOff \gg 16$$
$$FrstClusHI = ClusOff$$

En las expresiones anteriores, se asigna a la variable *ClusTemp* el valor de la cantidad de clusters que componen a un archivo de tamaño máximo. En la segunda expresión se obtiene el valor del primer clúster del archivo con número *NA*, multiplicando *NA – 1* por la cantidad

ClusTemp y sumándole 0x0023 en hexadecimal (35 en decimal), que corresponde al número del primer clúster del primer archivo.

Dado que las variables *FrstClusLO* y *FrstClusHI* son de 16 bits y que el número de clúster es de 32 bits, al igualar *FrstClusLO* a *ClusOff*, el primero tomará sólo los primeros 16 bits (que corresponden a la parte baja). Para *FrstClusHI*, se hace un corrimiento de 16 bits a la variable *ClusOff*, y se iguala a *FrstClusHI*. Con esto último, la siguiente entrada se escribirá con los parámetros recién calculados del número del primer clúster del archivo.

Al terminar la función *WriteRootDir*, se llamará a la función *Allocation*, que se encargará de asignar el espacio en la región de datos para cada archivo. Dicha función leerá las entradas recién creadas en el directorio raíz, obteniendo el número del primer clúster y el tamaño. A partir de lo anterior, creará las entradas necesarias en la tabla FAT. En las figuras 3.60 a 3.61 se detalla la función *Allocation*.

La función *Allocation* inicia leyendo la segunda entrada del directorio raíz (la primera corresponde al propio Directorio Raíz) y obtiene los parámetros *FrstClusHI* y *FrstClusLO*, con los cuales conforma el número del primer clúster del archivo, *FrstClusNum*.

Se utiliza la función *ClusNumToFAT*, presentada anteriormente, para obtener la posición en la tabla FAT de un clúster determinado, en este caso el primero del archivo. Esta subrutina calcula el número de sector (*ThisFATSecNum*) y el número de byte (*ThisFATEntOff*) en el que se encuentra la entrada de un clúster dado.

Considerando que en la mayoría de los casos, la primera entrada no se encuentra en la primera posición del sector y que el resto de la información pude corresponder a otro archivo o al Directorio Raíz, se almacenarán las entradas que se encuentren antes de dicha entrada, es decir, se leerá y almacenará el sector *ThisFATSecNum*. Además, se calcula la cantidad de entradas disponibles en este sector.

Con la función *CntOfClusters* se obtendrá la cantidad de clusters que componen al archivo y que equivaldrán al número de entradas en la tabla FAT.

Al iniciarse la escritura de las entradas, en el primer bloque se enviarán las entradas anteriores al byte *ThisFATEnOff*, que fueron almacenadas.

El valor de cada entrada indica el número del siguiente clúster que compone al archivo. En FAT16 cada entrada es de 16 bits (2 bytes) y en FAT32 las entradas son de 32 bits (4 bytes). Como los archivos se registrarán de manera continua y al igual que los clusters que los componen, el valor de las entradas FAT tendrá un orden consecutivo. Para indicar el fin de un archivo la última entrada debe tener el valor 0xFFFF en FAT16 y 0xFFFF FFFF en FAT32.

Si el Directorio Raíz tiene registrado más de un archivo, se leerá la siguiente entrada del mismo y se llamará de nuevo a la función *Allocation*, y de la misma manera para los siguientes archivos.

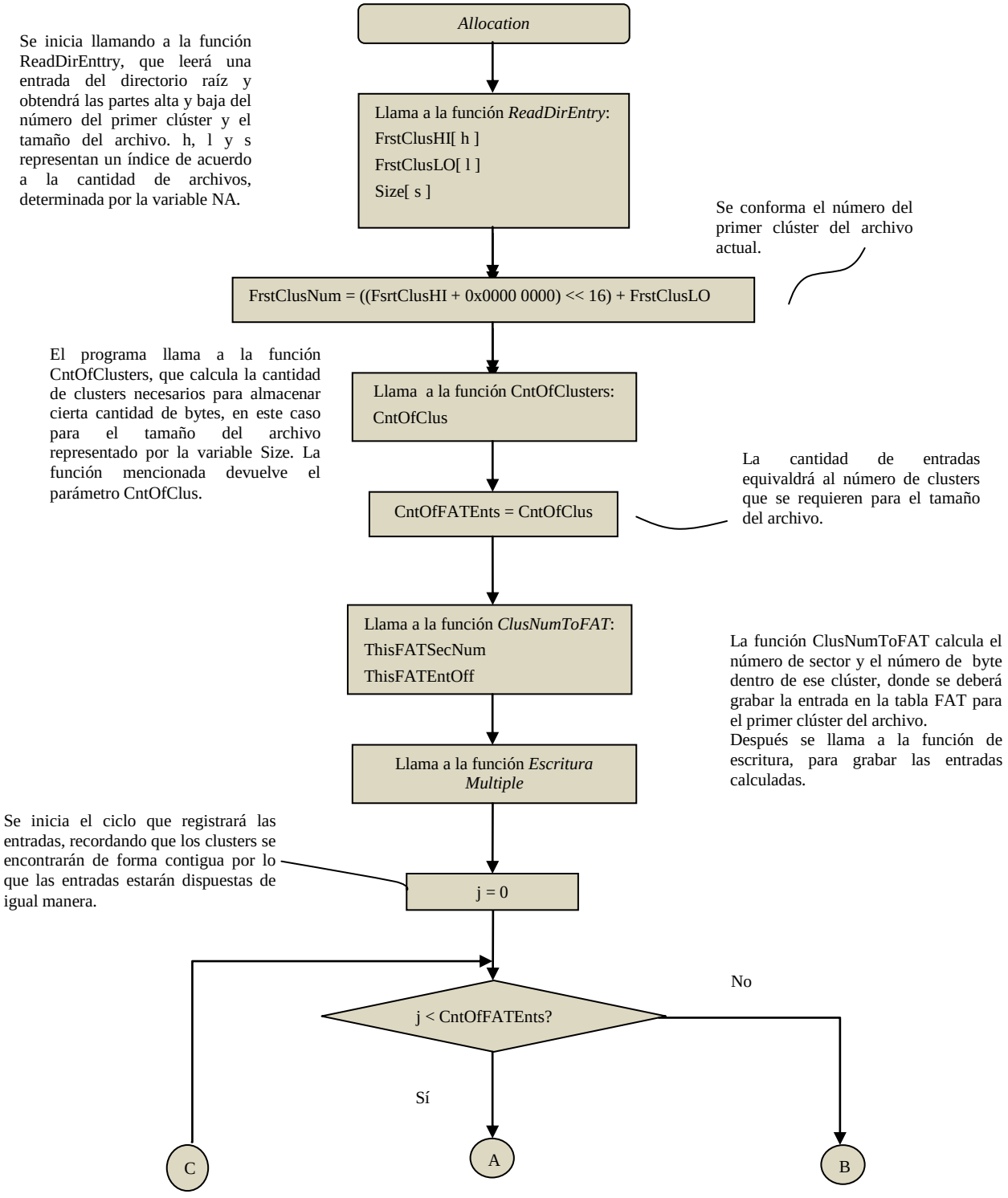


Figura 3.60. Diagrama de la función Allocation (1 de 2).

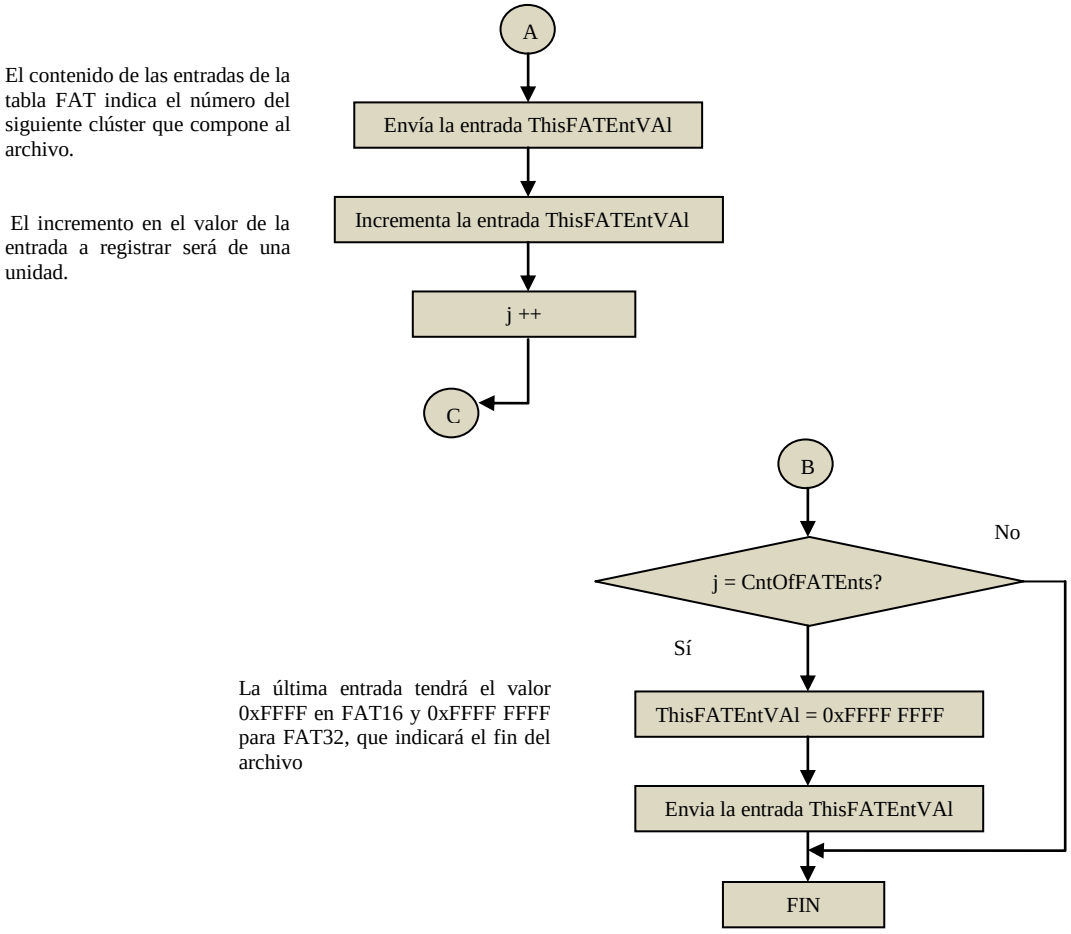


Figura 3.61. Diagrama de la función Allocation (2 de 2).

Una vez que se ha aplicado la función *Allocation* para todos los archivos registrados en el Directorio Raíz, se puede iniciar el almacenamiento de los datos. En este punto, el sistema operativo de una PC que leyera la tarjeta SD, mostraría que tiene la cantidad de archivos creados y los atributos programados, pero, al abrir estos archivos en un editor de textos, se observaría que están vacíos.

El registro de los datos provenientes de la unidad sísmica SR04 se realizará considerando la cantidad total de bytes a registrar, es decir, no se registrará cada archivo individualmente. De esta manera, se iniciará una sola rutina de escritura, que iniciará en el primer clúster del primer archivo y terminará en el último clúster del último archivo, evitando la pérdida de datos al momento de salir de la rutina principal para obtener información del siguiente archivo, cuando la cantidad de información a registrar así lo requiera.

La rutina de almacenamiento de los datos adquiridos utiliza la función de escritura múltiple de bloques de datos. Cada bloque estará compuesto por 512 bytes. En la figura 3.62 se puede observar el diagrama para esta rutina.

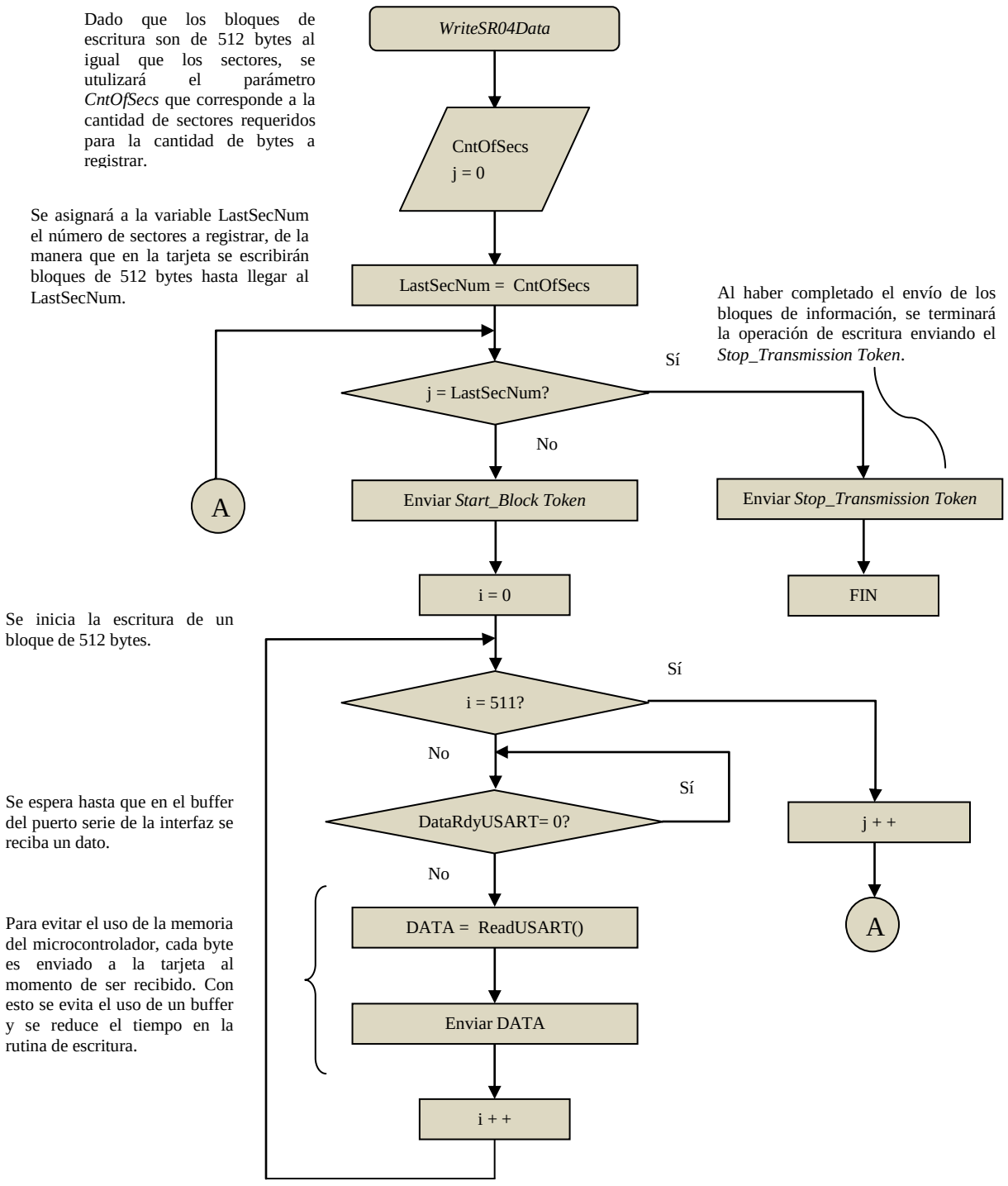


Figura 3.62. Diagrama de la función de registro de los datos de la unidad sísmica SR04 en las tarjetas de memorias SD.

Al terminar la rutina de registro se mostrará un mensaje en el display de la interfaz que indicará el fin del registro.

Las funciones hasta aquí descritas constituyen el software del microcontrolador o *firmware*.

En este capítulo se ha mostrado el desarrollo de la Interfaz de Registro para la Unidad Sísmica SR04. Se describió tanto el desarrollo de los componentes del *Hardware* como el desarrollo de las rutinas programadas para el microcontrolador (*software*), que en conjunto pretende dar solución a la problemática planteada en la introducción del presente trabajo. En el siguiente capítulo se presentarán las pruebas realizadas a la Interfaz desarrollada.

4. PRUEBAS REALIZADAS A LA INTERFAZ DE REGISTRO

En este capítulo se comentarán las pruebas realizadas a la Interfaz de registro desarrollada. Comenzaremos por mostrar las pruebas realizadas a las partes más significativas del *Hardware* y del *Software*. Posteriormente se presentarán las pruebas realizadas a la Interfaz en la etapa final del desarrollo.

4.1 Integración del Hardware

Una vez analizado el funcionamiento básico de los distintos elementos que conforman a la interfaz de registro, éstos se montaron en tarjetas *Protoboard*, lo que permite realizar pruebas, cambios e integrar componentes adicionales conforme se avanza en el desarrollo del prototipo, ver figura 4.1.

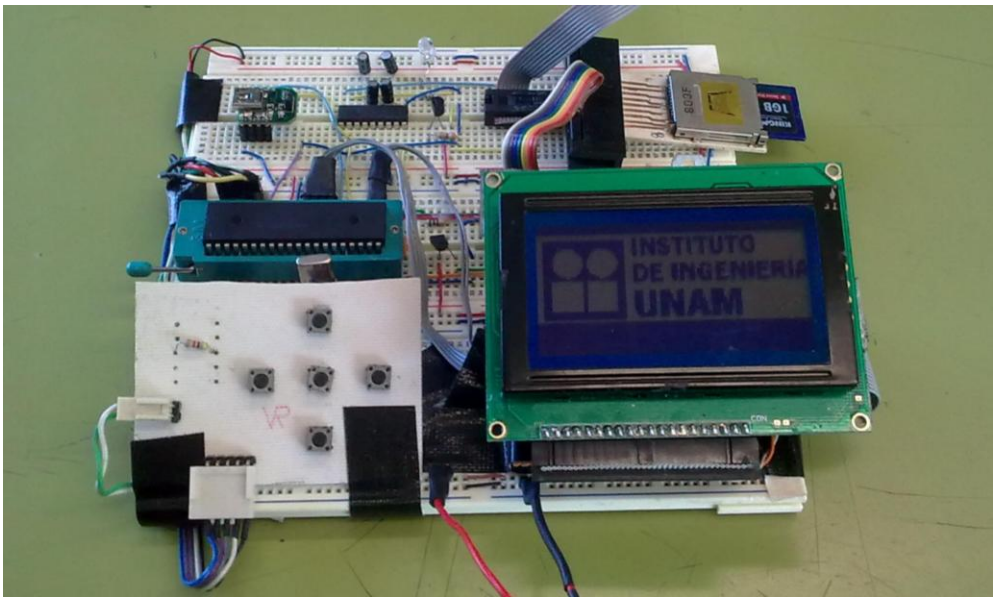


Figura 4.1. Hardware de la interfaz montado en tarjetas protoboard.

En la figura 4.2 se muestra el diagrama eléctrico de la interfaz, donde se observa la interconexión de los distintos elementos de *hardware* mencionados anteriormente.

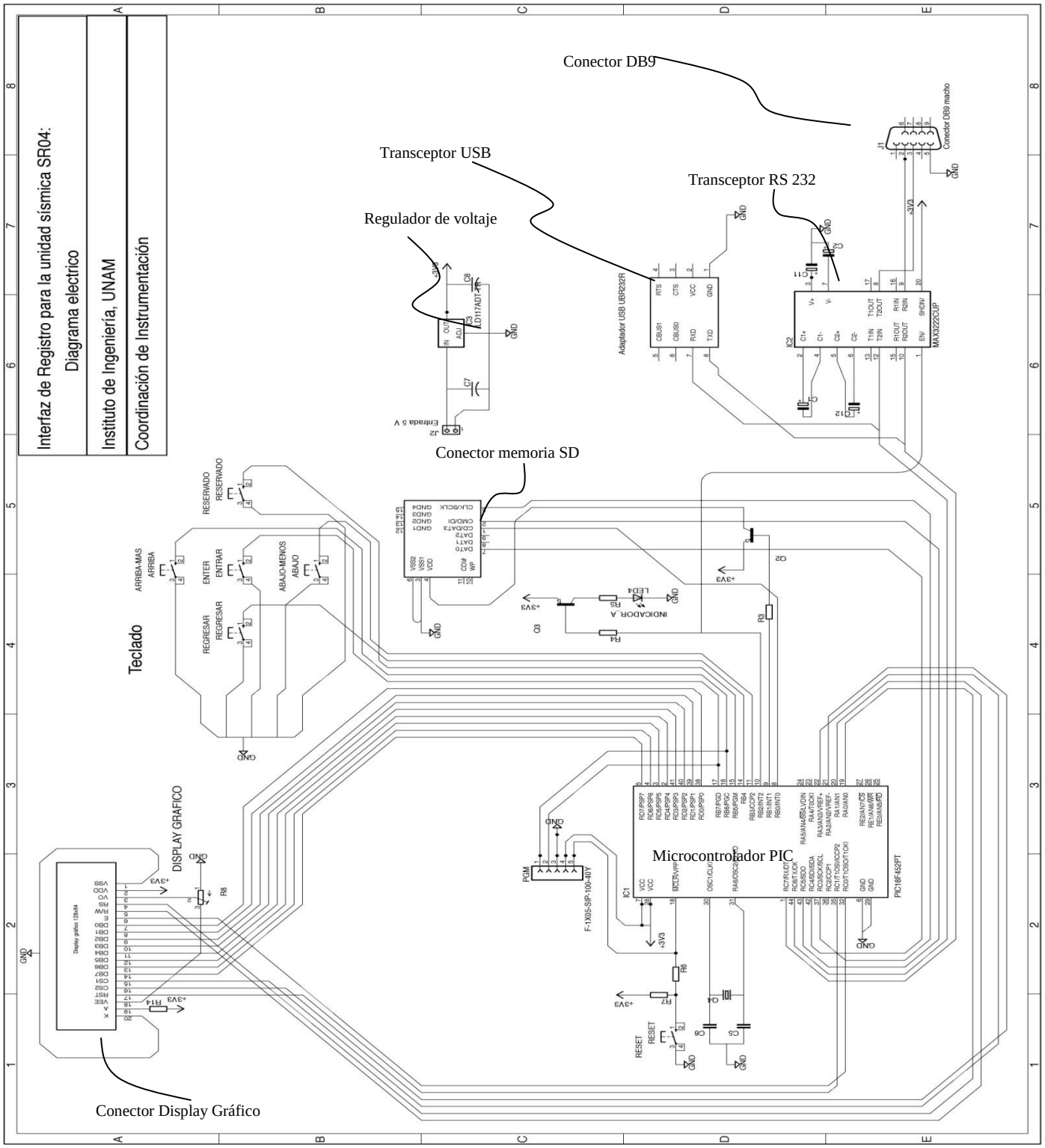


Figura 4.2. Diagrama eléctrico de la Interfaz de registro.

Al ser la Interfaz de registro un sistema basado en microcontrolador, la parte más significativa del *hardware* es el propio microcontrolador, los componentes restantes están sujetos a las rutinas que se programen para ellos, por lo que las pruebas que se realizan al *software* incluyen las pruebas al hardware.

4.2. Prueba a los menús de opciones

El propósito de esta prueba es comprobar que al pulsar las teclas Arriba o Abajo se cambia la flecha o cursor en la pantalla y que al pulsar la tecla entrar, ésta lleve a otro submenú o que se llame a la función a ejecutar.

Al ser energizada la Interfaz, ésta muestra el menú principal con las opciones generales de Configuración, Modo de Prueba y Registro, que siempre inicia con la opción de Configuración seleccionada, ver figura 4.3. Al pulsar la tecla abajo se cambia el cursor a la opción Modo de prueba, si se pulsa de nuevo el cursor señala a la opción de Registro y si se pulsa una vez más el cursor regresará a la opción de Configuración.

Figura 4.3. Menú de principal

Si se pulsa la tecla Arriba al iniciar el sistema, el cursor cambia de la opción Configuración a la opción Registro. De esta manera, siempre que se pulsa alguna de estas teclas se cambiará la opción seleccionada. Lo mismo ocurre en cualquiera de los submenús que se han desarrollado.

Al pulsar la tecla Entrar en la opción de **Configuración** se muestra el submenú de configuración, con las frecuencias más utilizadas en las pruebas realizadas con la unidad sísmica SR04, como se muestra en la figura 4.4.

Al pulsar Entrar sobre la opción de **Modo de prueba** se muestran las opciones de prueba. Dichas opciones incluyen a los tres canales y al paquete de tiempo, ver figura 4.5. Más adelante se mostrarán las pruebas realizadas para obtener y visualizar las gráficas del comportamiento de los canales y la información de fecha y hora GMT.

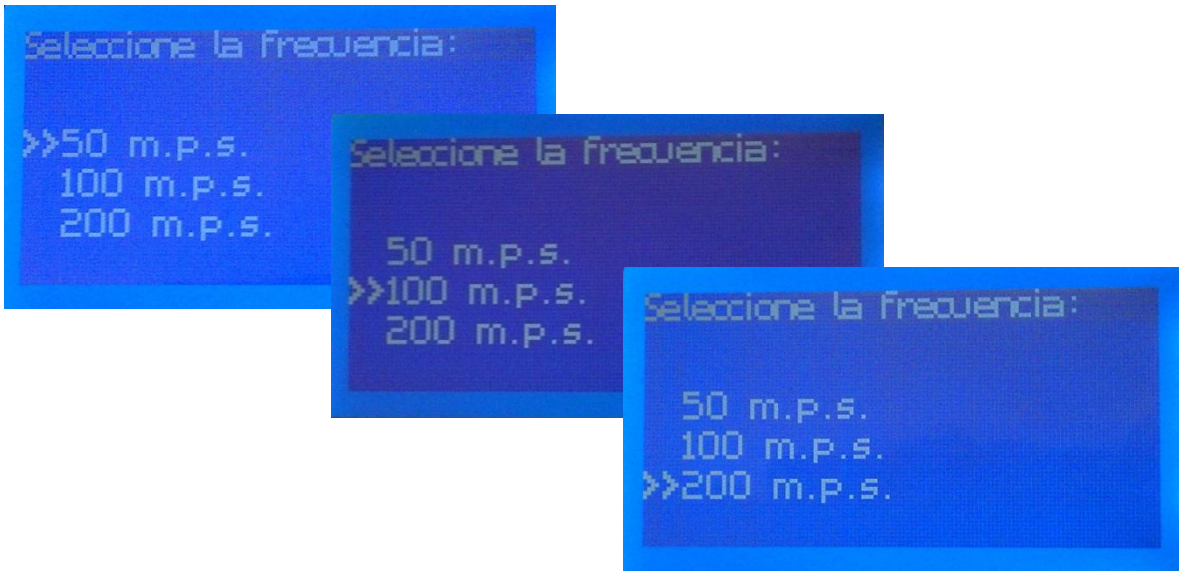


Figura 4.4. Menú de selección de frecuencias.

Figura 4.5. Menú del Modo de Prueba.

En el caso de pulsar Entrar al estar seleccionada la opción de **Registro** (ver figura 4.6), se comenzará con el proceso de configuración del registro de datos, mostrando un mensaje que invita al usuario a insertar la tarjeta de memoria SD y después pulsar Entrar.

Al pulsar Entrar se muestra un mensaje que indica que se está leyendo la tarjeta SD y, en la parte superior de la pantalla, se muestra otro mensaje que indica el sistema de archivos con el que cuenta la tarjeta SD, tal como se observa en la figura 4.7.

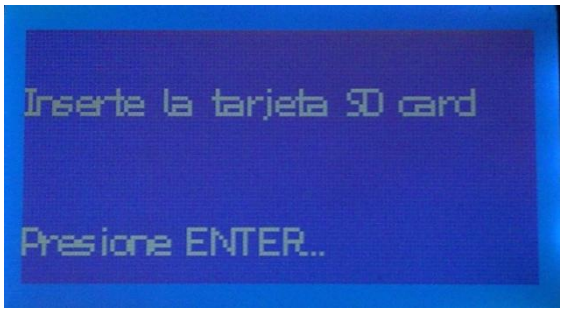


Figura 4.6. Mensaje inicial en el modo de Registro.

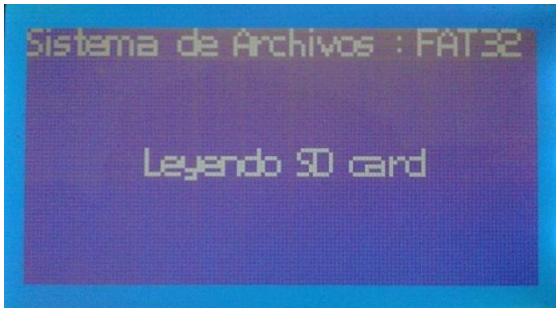


Figura 4.7. Mensaje que indica el sistema de archivos detectado.

Unos segundos después de mostrar la pantalla anterior, se genera otro menú para la selección del periodo de registro, que puede ser en horas o en días, ver figura 4.8.

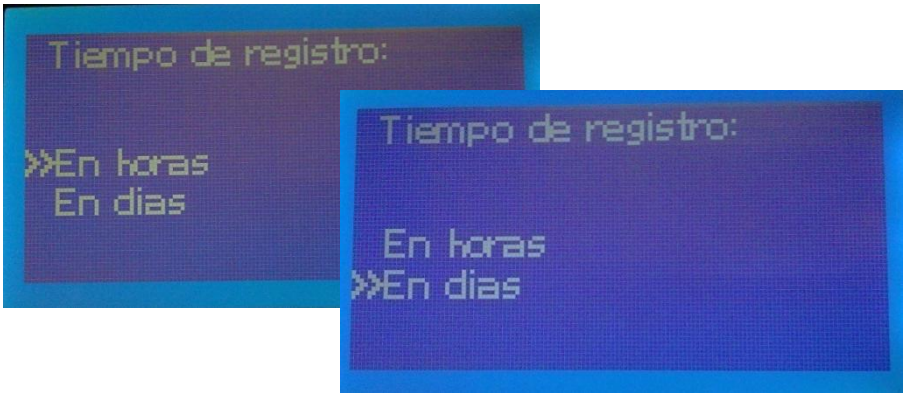


Figura 4.8. Selección del tipo de periodo de registro.

En el caso de seleccionar la opción de horas se muestra la pantalla de la figura 4.9. En esta opción, cada vez que se pulsa la tecla arriba, el número mostrado se incrementa en una unidad. El número de horas a registrar llega hasta 250 y luego se regresa a 1. Si se pulsa la

tecla Abajo, se disminuye en una unidad el número mostrado. Si el número de horas mostrado es 1 y se pulsa la tecla Abajo, el número mostrado cambia a 250.

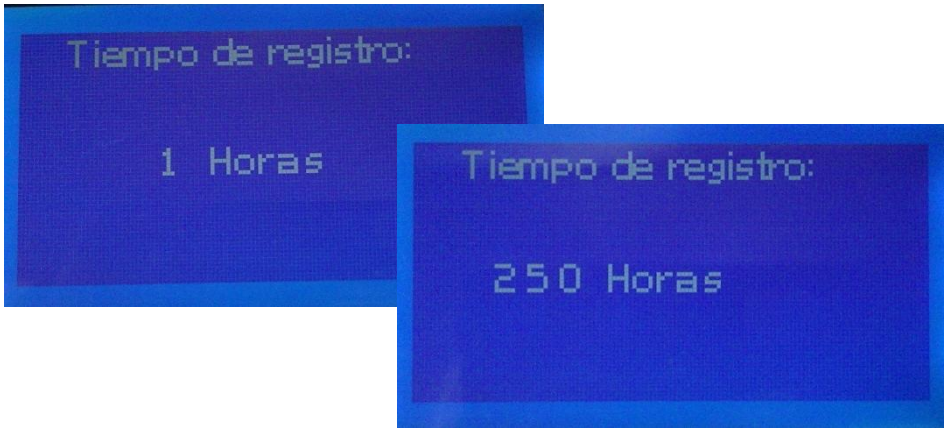


Figura 4.9. Selección de la cantidad de horas de registro.

Cuando se selecciona el registro por días se muestra la pantalla de la figura 4.10. Al igual que en el registro por horas, al pulsar la tecla Arriba se incrementa en una unidad el número mostrado, y si se pulsa la tecla Abajo se disminuirá en una unidad. El número mostrado se incrementa hasta 30, es decir, el máximo número de días a registrar es 30.

Después de seleccionar la cantidad de días u horas a registrar y de pulsar la tecla entrar, se mostrará de nuevo el menú de configuración de frecuencia y por último, se pedirá al usuario que pulse la tecla Entrar para iniciar el registro de datos.

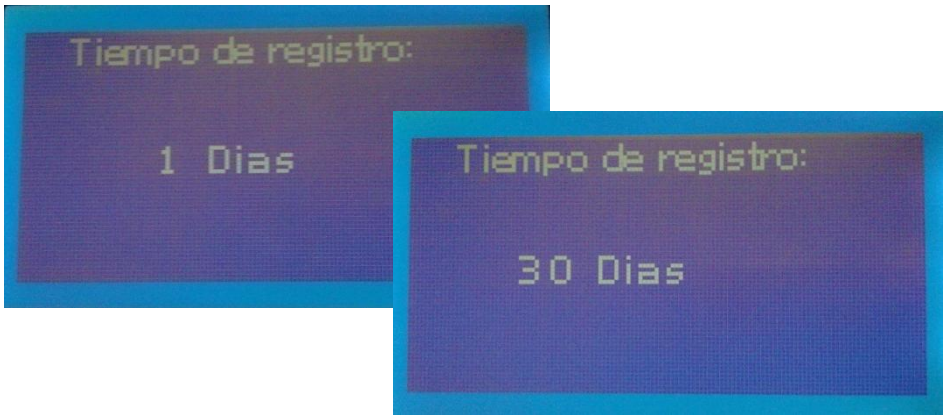


Figura 4.10. Selección de la cantidad de días de registro.

4.3. Modo de Prueba

La función Modo de prueba de la Interfaz se encarga de comprobar el funcionamiento de la unidad sísmica SR04. Este modo permite graficar el comportamiento de los sensores conectados a los 3 canales con los que cuenta la tarjeta SADC20, además, muestra la información de fecha y hora GMT que proporciona la tarjeta GPSDCF de dicha unidad.

Al seleccionar cualquiera de los 3 canales, la Interfaz comenzará a adquirir los datos que genera la unidad sísmica, esperando recibir el byte de identificación del canal seleccionado. Al detectar dicho byte, tomará los siguientes cuatro bytes, correspondientes a la muestra, y aplicará el algoritmo de decodificación, comentado en el capítulo anterior. Este proceso se realiza hasta obtener 127 muestras, graficando cada muestra obtenida. La Interfaz está programada para llenar la pantalla con 3 gráficas, tras las cuales se muestra de nuevo el menú del Modo de Prueba.

Para evaluar el funcionamiento del algoritmo de decodificación y trazado de gráficas con el que cuenta la Interfaz de registro, se introdujo una señal senoidal proporcionada por un generador de funciones.

Se ajustó el generador de funciones para obtener una señal senoidal con amplitud pico de medio volt y frecuencia de 1 Hz, esta señal se conecta a los tres canales de la tarjeta SADC20 en sustitución de los geófonos.

Se utilizó, además, el modo de Configuración para establecer las distintas opciones de frecuencia, con el fin de obtener gráficas a distintas tasas de muestreo de la señal introducida.

En la figura 4.11 se muestran las gráficas obtenidas en una prueba para cada canal a 100 mps.

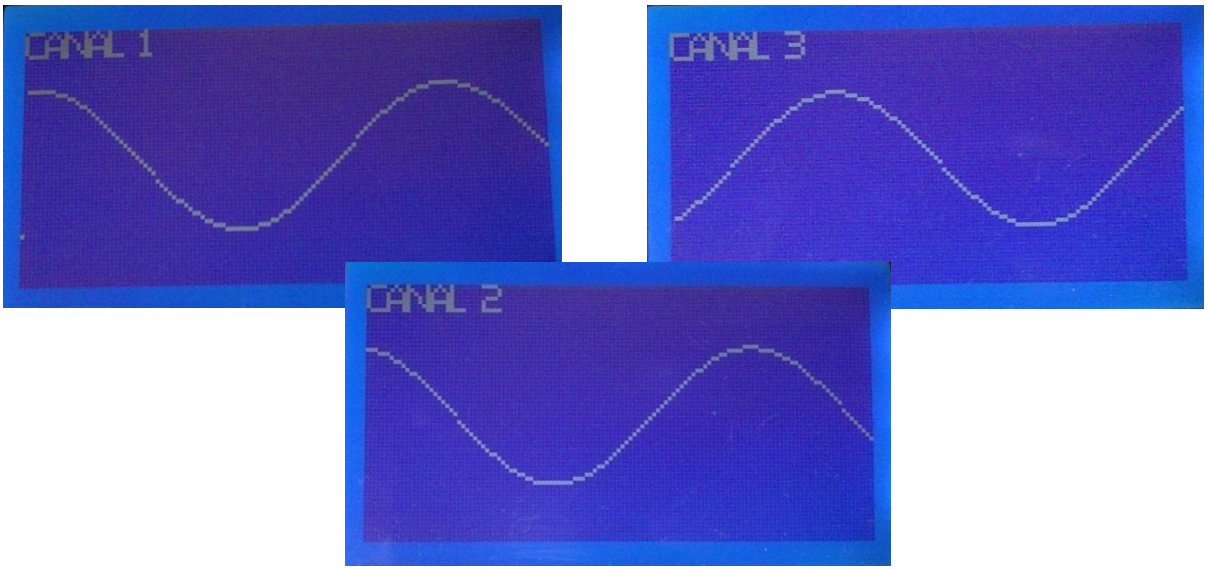


Figura 4.11. Gráficas obtenidas de los 3 canales a 100 mps.

En la figura 4.12 se muestran las graficas obtenidas en una prueba para el canal 1 a 50 mps (izquierda), 100 mps (centro) y 200 mps (derecha). Cabe comentar que la rutina que traza las gráficas siempre dibuja 128 puntos, independientemente de la frecuencia de muestreo, para este caso se puede notar que para 50 mps se obtienen poco más de 2 periodos de la señal, para 100 mps se obtiene un periodo y para 200 mps se observa el trazo de al menos medio periodo.

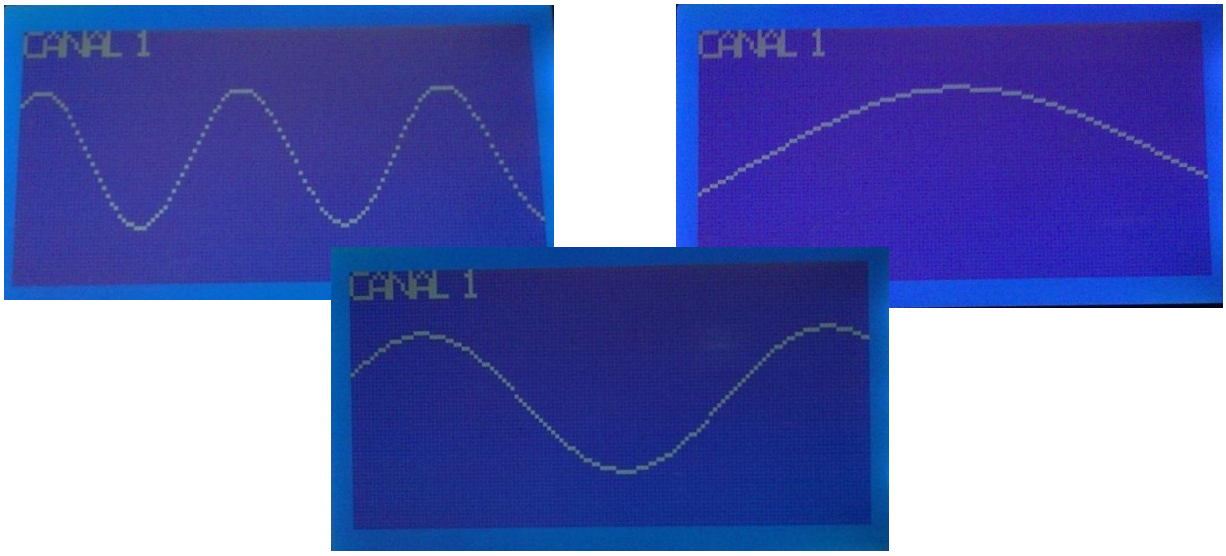


Figura 4.12. Gráficas obtenidas del canal 1 a 50, 100 y 200 mps.

El Modo de prueba permite, además, verificar si el GPS de la tarjeta GPSDCF está sincronizado. Al seleccionar Paquete de tiempo la interfaz se conecta con la unidad sísmica y comienza a adquirir datos, esperando recibir el byte de identificación del paquete de tiempo. El programa del microcontrolador de la interfaz decodifica la información del tiempo y la muestra como se observa en la figura 4.13.

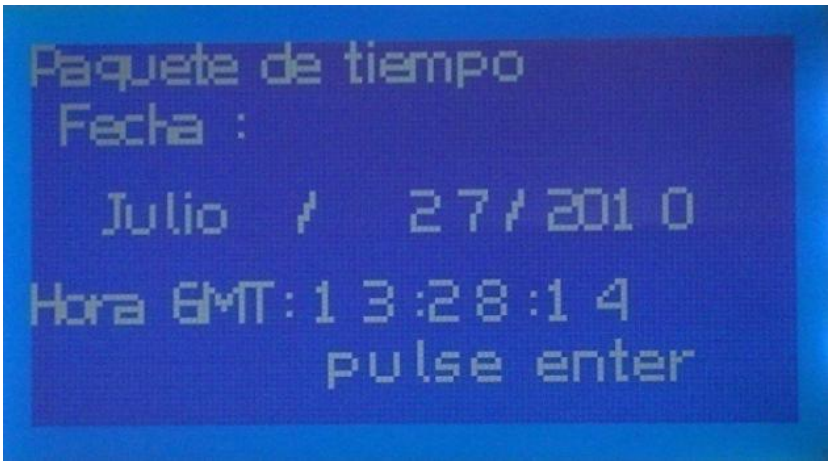


Figura 4.13. Paquete de tiempo decodificado.

Para comprobar que la información del paquete de tiempo obtenida es correcta, se compararon los datos obtenidos por la Interfaz con la fecha y hora proporcionada por la página de internet http://wwp.greenwichmeantime.com, que proporciona la hora GMT exacta.

En la figura 4.14 se muestra la captura de pantalla de la página con la hora GMT, en el mismo instante de tomar la foto de la figura 4.13.

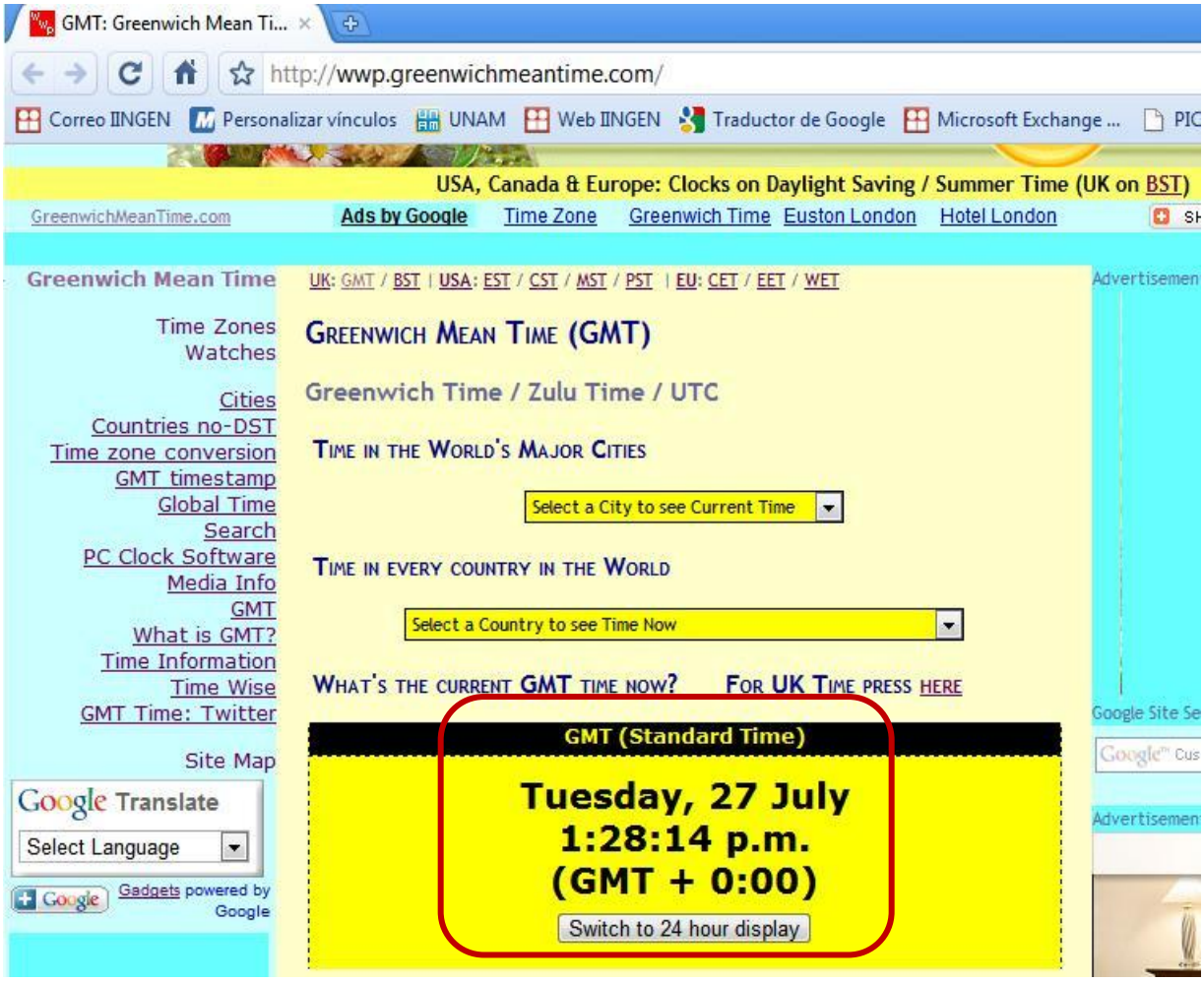


Figura 4.14. Comprobación del paquete de tiempo de la Interfaz de registro.

4.4. Prueba de Configuración de la frecuencia de muestreo

Con esta prueba se pretende comprobar el cambio en la frecuencia de muestreo de la unidad sísmica SR04, al utilizar la opción de **Configuración** de la Interfaz de registro desarrollada.

Esta prueba consiste en adquirir 10 segundos de información de la unidad SR04 con la terminal *Realterm* en una PC. Se utilizó la opción de captura de este software para guardar en un archivo la información que envía la unidad sísmica durante el periodo de tiempo mencionado. La cantidad de bytes obtenidos depende de la frecuencia de muestreo a la que esté trabajando dicha unidad.

La unidad sísmica se conectó a un puerto serie de la PC donde se ejecutó el software *Realterm*. El puerto fue configurado a 38 400 baudios, sin señal de *handshake*, 8 bits de datos, 1 bit de paro y sin paridad, como se señala en la figura 4.15.

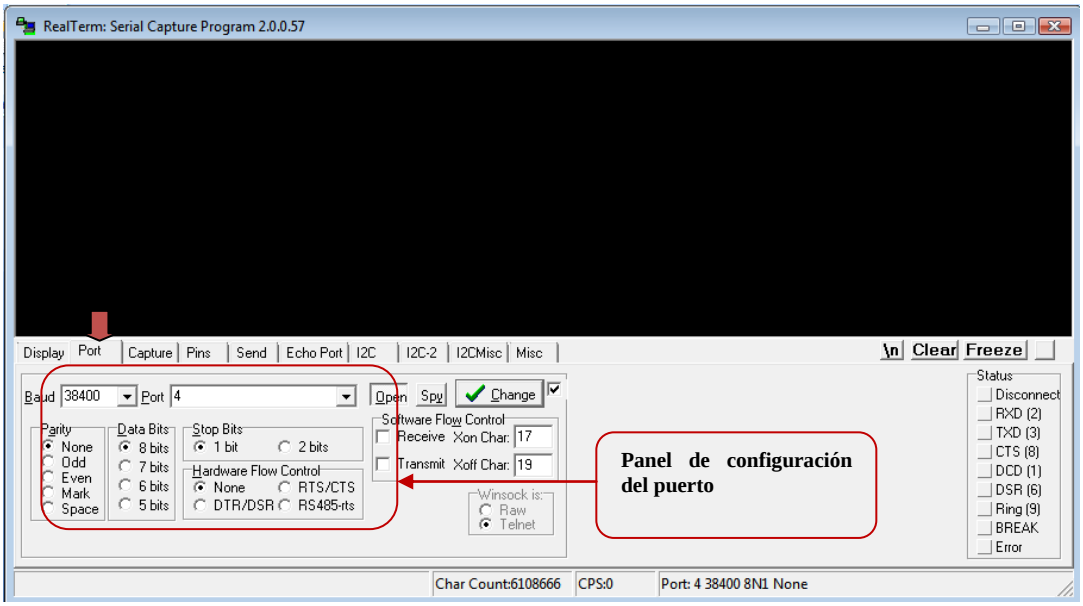


Figura 4.15. Configuración del puerto mediante el programa RealTerm.

En la *pestaña Capture* se seleccionó el archivo donde se almacenó la información colectada. Además, se le indicó al programa que capturara 10 segundos, como se resalta en la figura 4.16.

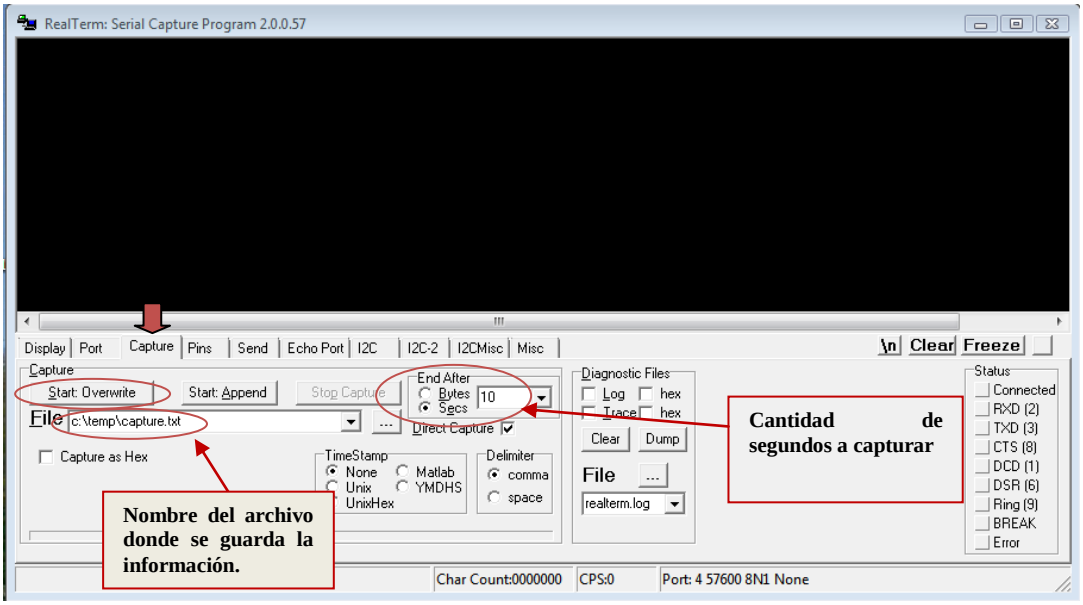


Figura 4.16. Opciones de captura en RealTerm.

Una vez realizada la conexión entre la unidad SR04 y la PC y después de energizar a la primera, se mostraron en la ventana de la terminal los datos que va arrojando la unidad SR04, como se observa en la figura 4.17. Así mismo, al dar clic sobre el botón *Start Overwrite* el programa comenzó a capturar y almacenar la información en el archivo designado.

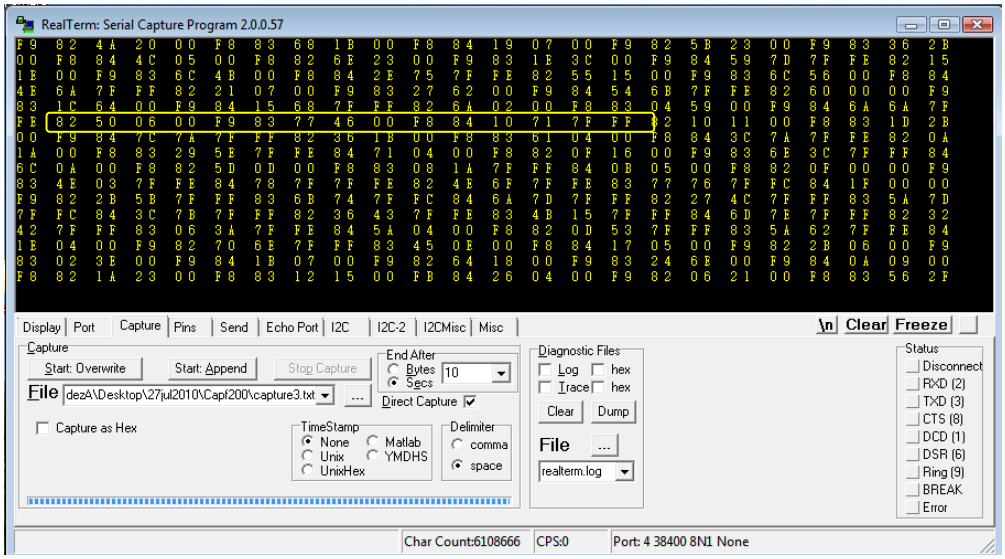


Figura 4.17. Captura de los datos de la unidad SR04 con Realterm.

Como se comentó al principio de este apartado, primero se configuró la frecuencia de muestreo de la unidad sísmica utilizando la función de Configuración de la Interfaz y después se realizó la captura de los 10 segundos de información con la PC.

En primer lugar se estableció una frecuencia de 50 mps. De acuerdo a la expresión 3.3, la cantidad de bytes de información que se debe generar con esta frecuencia de muestreo durante 10 segundos es 7590 bytes. En la figura 4.18 se observa el resultado obtenido.

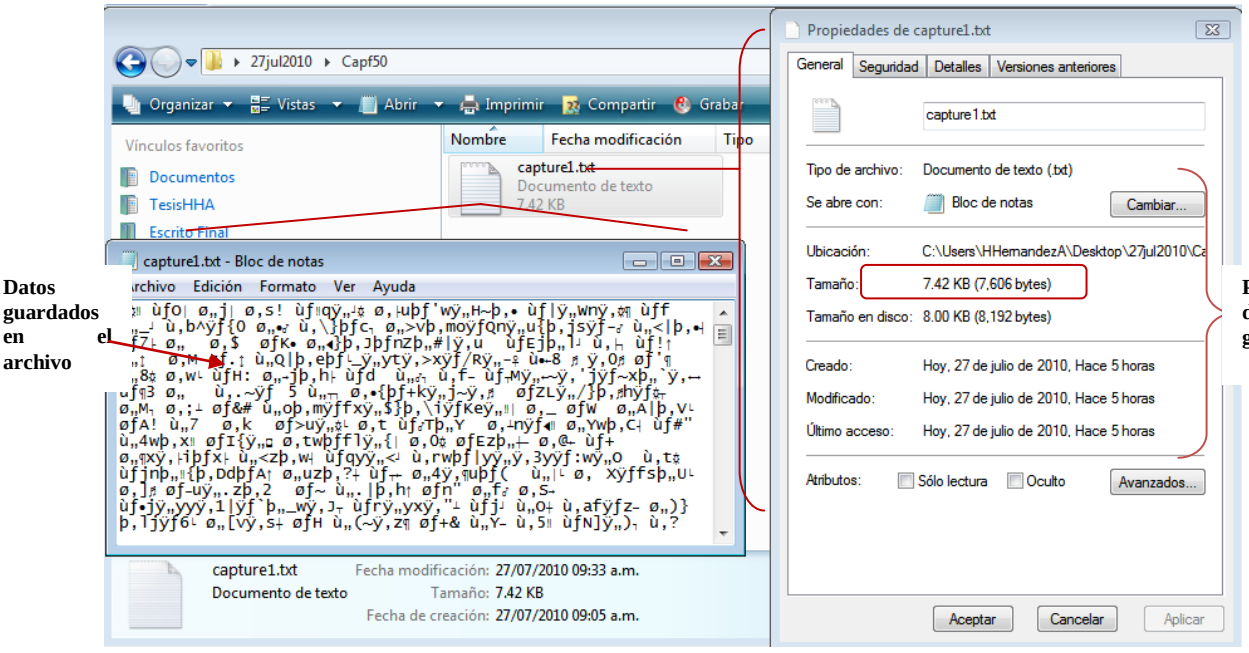


Figura 4.18. Propiedades del archivo generado para la frecuencia de 50 mps.

Las propiedades del archivo generado indican que contiene 7,606 bytes, cantidad aproximada al cálculo teórico, en donde la diferencia entre la cantidad calculada y la obtenida es debida al manejo del archivo por parte del software *Realterm*.

Después, se utilizó de nuevo la Interfaz en el modo de configuración para establecer la frecuencia de muestreo de la unidad SR04 en 100 mps, tras lo cual se conectó a la PC para realizar la captura de los 10 segundos de información. En la figura 4.19 se observa el resultado de este proceso.

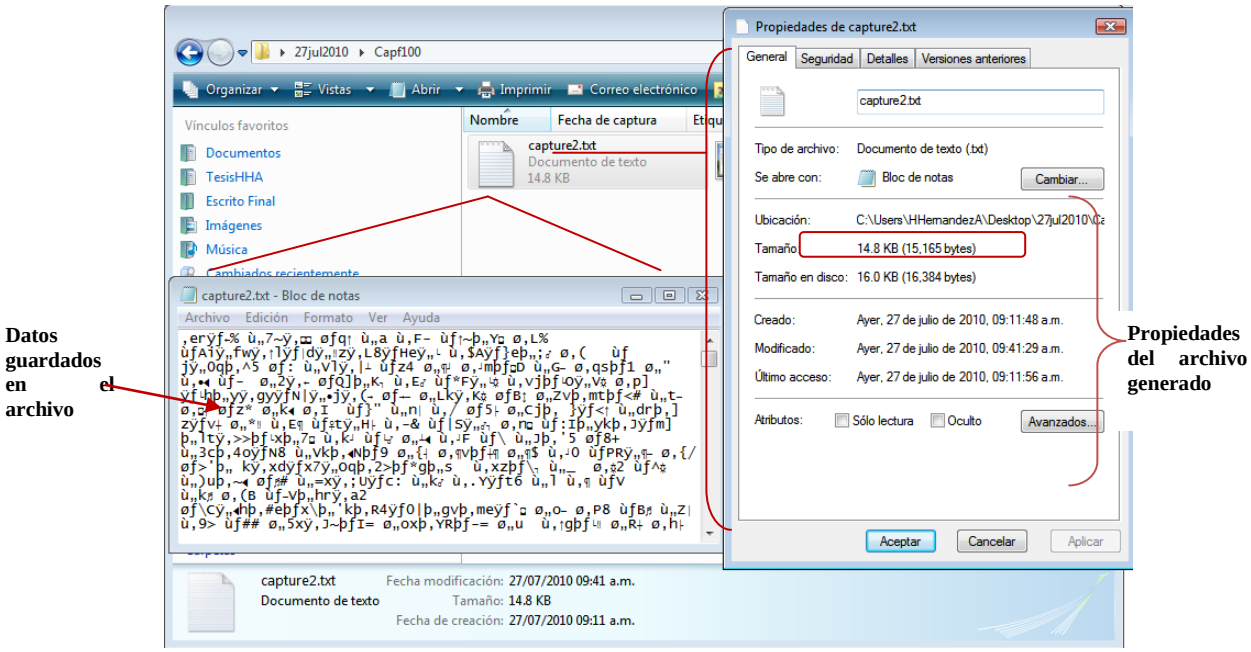


Figura 4.19. Propiedades del archivo generado para la frecuencia de 100 mps.

Al igual que en la prueba anterior, se obtienen las propiedades del archivo generado, en donde se puede observar que el archivo contiene 15,165 bytes. Para esta prueba, la cantidad de bytes calculada es de 15,090.

Por último, se realizó el proceso correspondiente para establecer la frecuencia de muestreo en 200 mps. El archivo capturado deberá contener 30,090 bytes, esto de acuerdo a la expresión 3.3. En esta prueba, las propiedades del archivo obtenido mostraron que la cantidad de bytes recolectada durante los 10 segundo fue de 30 089, cantidad aproximada al valor teórico (ver figura 4.20). Cabe comentar que se realizaron varias veces cada prueba y la diferencia entre el valor teórico y el obtenido fue distinta en cada caso, tratándose de un error atribuible al programa.

Como un método alternativo, se puede realizar una inspección cualitativa del cambio de frecuencia al aplicar la función de Configuración de la interfaz, esto es, primero se entra en el Configuración y se establece la frecuencia de muestreo más baja, es decir 50 mps, después se entra en el Modo de prueba y se selecciona alguno de los canales. De nuevo, se entra en el modo de Configuración para establecer la frecuencia en 100 mps, tras lo cual regresamos al

Modo de prueba a graficar alguno de los canales. Con esta alternativa se pudo observar la diferencia en la velocidad del trazado de las graficas en el display de la Interfaz, que para la frecuencia de 50 mps se nota más lenta que para 100 mps.

Como conclusión podemos decir que, mediante la realización de estas pruebas se verificó el funcionamiento de las rutinas del programa de la Interfaz, que se encargan de conformar y enviar el comando de Ajuste de frecuencia de muestreo.

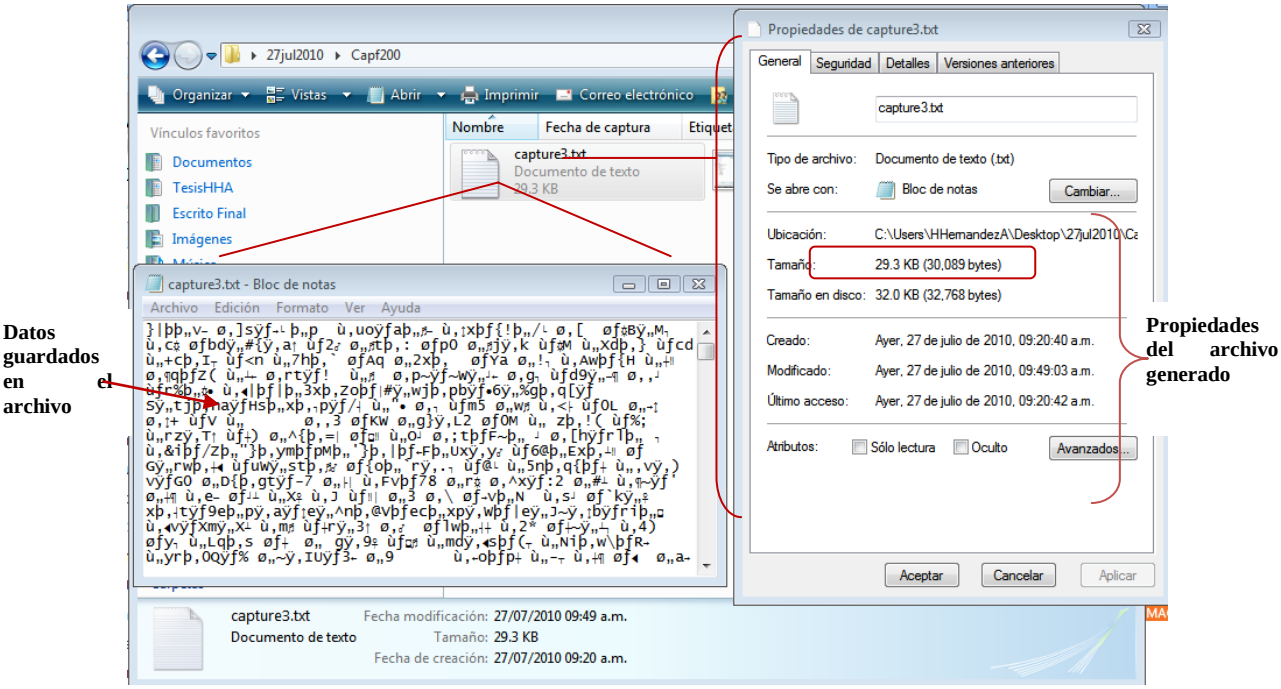


Figura 4.20. Propiedades del archivo generado para la frecuencia de 200 mps.

4.5. Pruebas de comunicación entre la Interfaz y la tarjeta de memoria SD

El objetivo de estas pruebas es verificar el funcionamiento de las rutinas de lectura y de escritura que se han programado en el microcontrolador de la Interfaz.

En las pruebas que se aplicaron en este apartado, además de utilizar la terminal *RealTerm*, se usó el software *WinHex*, en su versión de prueba, que es una herramienta que permite analizar datos a bajo nivel dentro de las unidades de almacenamiento, como lo son las tarjetas de memoria SD. Este programa permite visualizar todas las regiones que componen al sistema de archivos de algún medio de almacenamiento.

Prueba de lectura del Sector de Arranque

Esta prueba consistió en leer con la interfaz el contenido del sector de arranque de una tarjeta SD y enviarlo a la terminal de la PC, a través del puerto serie. Posteriormente, la misma tarjeta fue leída directamente con la PC, utilizando el programa *WinHex*. La información obtenida por ambos medios fue comparada.

En el microcontrolador se escribió una rutina que inicializa la tarjeta y posteriormente llama a la función de lectura de un bloque, especificando como dirección de lectura el primer sector de la memoria, es decir, el sector de arranque.

En la figura 4.21 se observa la pantalla con la información que envía el microcontrolador y que fue capturada con *Realterm*.

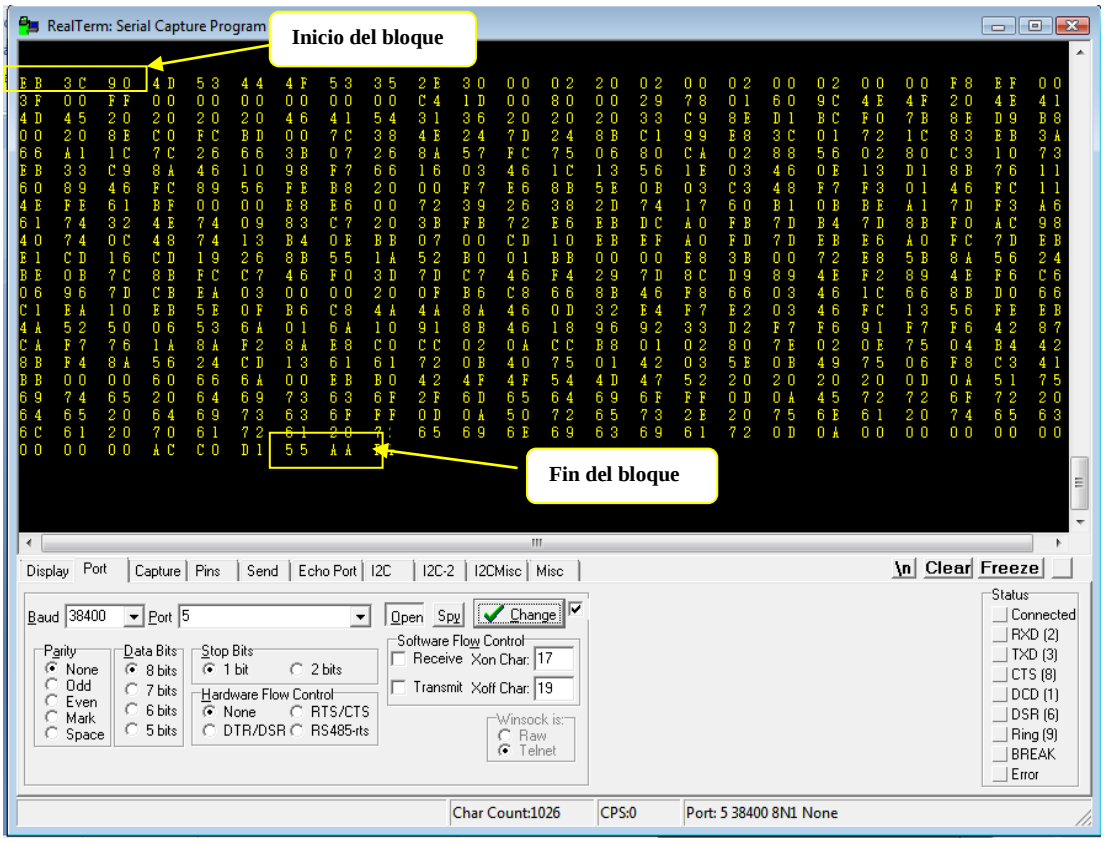


Figura 4.21. Bloque de datos leído de la tarjeta y enviado a la terminal.

De acuerdo con las especificaciones del sistema de archivos FAT de *Microsoft*, los primeros tres bytes del sector de arranque corresponden al código de la instrucción de inicio para procesadores Intel x86. Los dos últimos bytes corresponden a la denominada *firma del registro de inicio*, que siempre se encuentra en el offset 510 en decimal. En la figura 4.21 se han resaltado los primeros 3 bytes y los dos últimos del bloque capturado, donde se observa que su valor corresponde con las especificaciones mencionadas.

La misma tarjeta fue llevada a la PC para ser analizada con el software WinHex. Al leer dicha tarjeta y posicionar el visor de datos en el offset cero se pudo observar la información del sector de arranque. En la figura 4.22 se tiene la pantalla con la información obtenida en esta prueba, también se resaltan los primeros 3 bytes y los últimos 2, al igual que en la figura anterior.

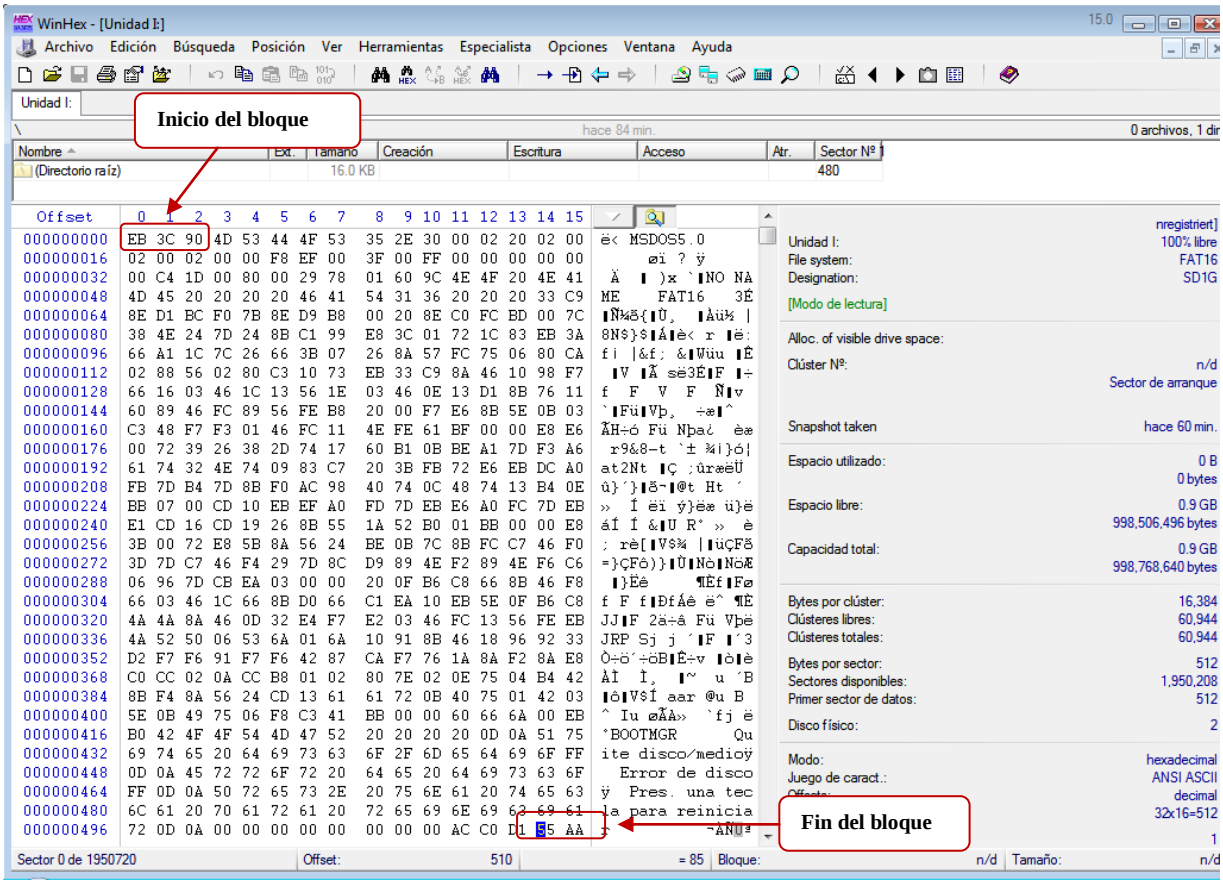


Figura 4.22. Sector de arranque leído en una PC con WinHex.

Al comparar los datos obtenidos y que se muestran en las figuras anteriores (Fig. 4.21 y 4.22), se pude observar que los dos bloques son idénticos, con lo que se verifica que la función de lectura implementada en la interfaz funciona correctamente.

Prueba de lectura del sistema de archivos

Para que la información escrita por la Interfaz de registro en las tarjetas SD pueda ser leída correctamente por un equipo de cómputo que reconozca los sistemas de archivos FAT, dicha información debe ser escrita respetando el sistema de archivos de la tarjeta en la que se realice la operación de escritura. Por lo anterior, antes de escribir la información necesaria se debe conocer el sistema de archivos de la tarjeta de memoria insertada en la Interfaz de registro.

En esta prueba se pretende verificar que la Interfaz de registro aplica correctamente los algoritmos implementados para la determinación del sistema de archivos de las tarjetas SD con las que se trabaja.

Para esta prueba se utilizaron 4 tarjetas de memoria SD con distintas capacidades y de distinta marca, además de tener formato de archivos FAT16 ó FAT32, como se muestra en la tabla 4.1.

Marca	Capacidad	Sistema de archivos
KINGMAX	1 GB	FAT16
A DATA	2 GB	FAT16
KINGSTON	2 GB	FAT16
KINGMAX	8 GB	FAT32

Tabla 4.1. Tarjetas SD utilizadas en la prueba de lectura del sistema de archivos.

Después de inicializar la tarjeta de memoria SD, se llama a la subrutina que se encarga de leer el sector de arranque y operar los campos necesarios para determinar el sistema de archivos con el que fue formateada dicha tarjeta. Al ser detectado el sistema FAT correspondiente se muestra un mensaje en el display indicando el sistema de archivos detectado.

En las pruebas realizadas, al insertar la tarjeta *KINGMAX* de 1 GB la Interfaz indicó que tiene sistema de archivos FAT16, ver figura 4.23. Lo mismo ocurrió con la tarjeta *ADATA* de 2 GB.

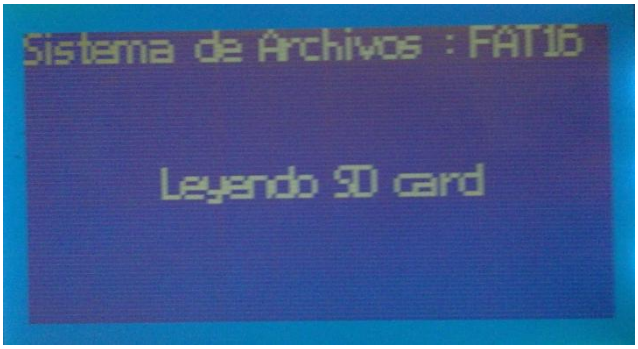


Figura 4.23. Mensaje indicando que se detectó el sistema de archivos FAT16.

Para la tarjeta *KINGMAX* de 8 GB, la interfaz indica que se cuenta con el sistema de archivos FAT32, ver figura 4.24.

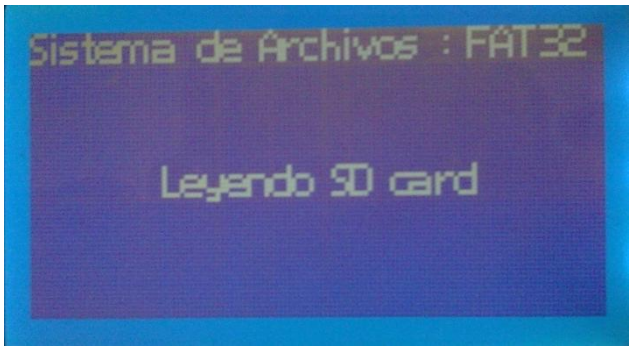


Figura 4.24. Mensaje que indica que se detectó el sistema de archivos FAT32.

En el caso particular de la tarjeta KINGSTON de 2GB, durante el proceso de inicialización la tarjeta responde indicando que no es compatible con las especificaciones implementadas. Debido a lo anterior, se detiene el proceso y se regresa al menú principal, dando oportunidad al usuario de insertar una tarjeta compatible. Esto implica que esta memoria SD no puede ser utilizada en la Interfaz de registro. Ver figura 4.25.

Figura. 4.25. Mensaje de error mostrado con tarjetas no compatibles.

Una tarjeta detectada por la Interfaz como no compatible implica que no soporta las especificaciones técnicas en su versión 2.0, debido a que el voltaje utilizado es distinto al soportado o a que se trata de un tipo de tarjeta diferente, por ejemplo una tarjeta *MMC*.

Tras realizar estas pruebas comprobamos que las funciones que determinan el sistema de archivos funcionan correctamente.

Prueba de creación de archivos de caracteres en las tarjetas SD

Una de las partes más importantes del registro en archivos de los datos provenientes de la unidad sísmica SR04 es la propia creación de los archivos, recordando que éstos deben cumplir con las especificaciones de los sistemas de archivos FAT. Con esta prueba se busca comprobar que las rutinas que fueron programadas para este fin funcionan correctamente, al crear archivos que puedan leerse en equipos de cómputo que soporten sistemas de archivos FAT.

La prueba aplicada consistió en utilizar las rutinas de creación de archivos para generar un sólo archivo, cuyo contenido fuese una serie de caracteres, en este caso "**ii UNAM**".

Después de realizar la prueba descrita, al leer la memoria SD con la PC pudimos observar que el archivo fue generado correctamente, con el contenido programado y el tamaño designado, tal como se muestra en la figura 4.26.

La misma tarjeta de memoria SD utilizada en esta prueba pudo ser leída en un equipo de cómputo *Apple* con sistema operativo *MAC OS X*, que tiene compatibilidad con los sistemas de archivos FAT, como se puede observar en la figura 4.27.

Con lo anterior se comprobó que la creación de archivos implementada cumple con las especificaciones de los sistemas de archivos FAT, permitiendo que los archivos generados puedan ser leídos por casi cualquier equipo de cómputo.

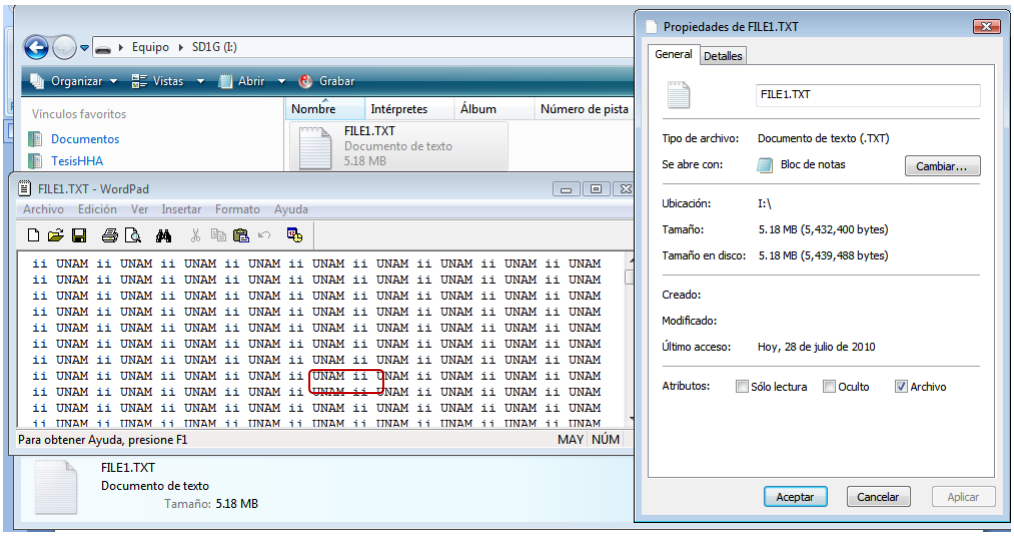


Figura 4.26. Archivo con caracteres generados en el microcontrolador visto en una PC.

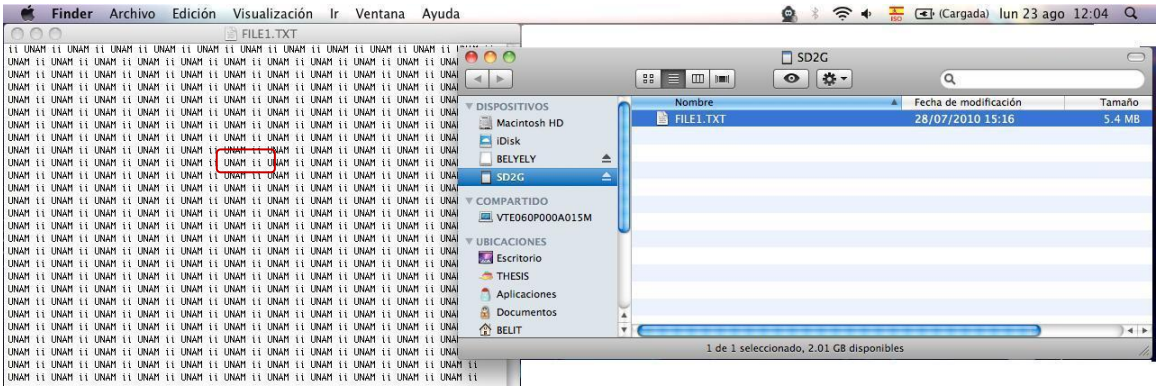


Figura 4.27. Archivo de caracteres visto en un equipo Apple.

4.6. Prueba del modo de Registro

Esta prueba consiste en utilizar la opción de Registro de la Interfaz para programar distintos periodos de registro y verificar la correcta creación de archivos en la tarjeta SD.

Primero de programó a la Interfaz para que realizara el registro de 1 hora a 100 mps. Una vez transcurrido el tiempo requerido se llevó la tarjeta a una PC y al abrirla pudimos observar que se creó un archivo con las características mostradas en la figura 4.28.

De acuerdo a la expresion 3.3, que permite clacular la cantidad de bytes a registrar, en funcion de la frecuencia de muestreo y del tiempo que se requiere registrar, en una hora se generan 5,432,400 bytes, que corresponde al contenido del archivo registrado en la tarjeta SD.

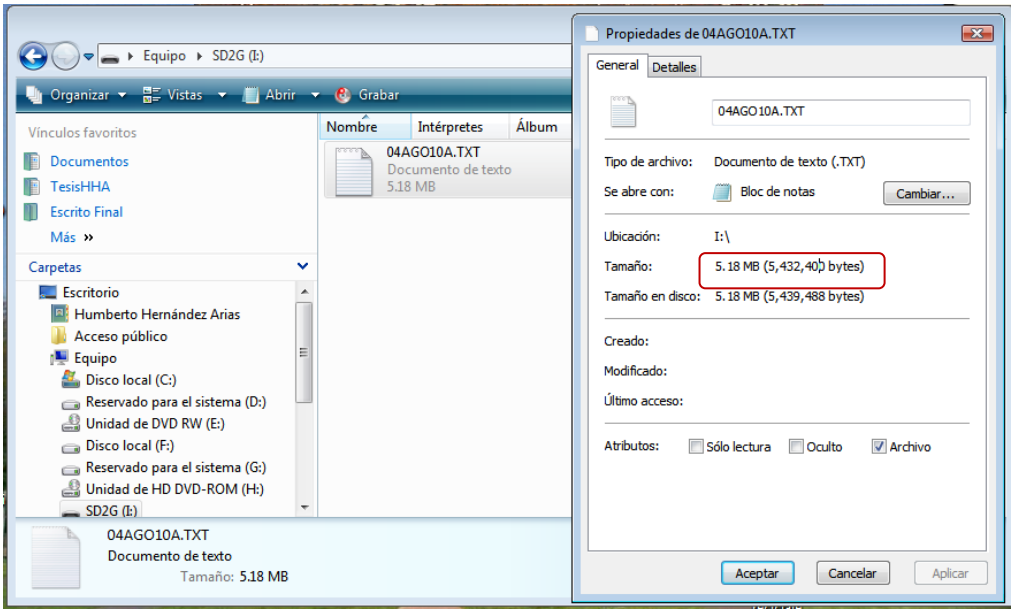


Figura 4.28. Propiedades del archivo generado para una hora de registro.

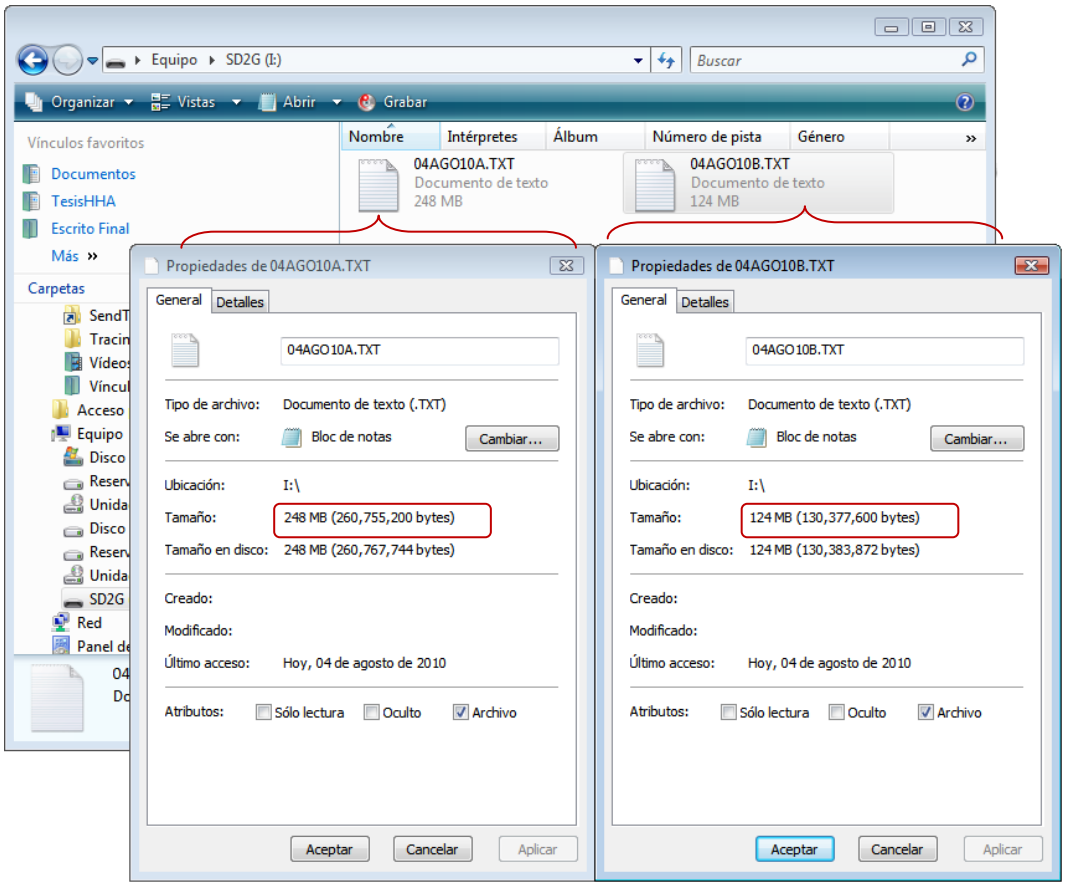


Figura 4.29. Propiedades de los archivos generados para un registro de 3 días.

También se programó el registro de 3 días a 100 mps, con el fin de verificar si se crean los 2 archivos requeridos. Como se muestra en la figura 4.29, la interfaz creó los 2 archivos con distinto nombre, donde el primero es de tamaño máximo (aproximadamente 248 MB) y el segundo con un tamaño de 124 MB.

Por último, se realizó la programación de la Interfaz para el registro de 30 días de información a 100 mps, que es la máxima cantidad de tiempo que se puede registrar con este sistema. Después de que en la pantalla de la Interfaz se muestra el mensaje ***Registrando,*** se retira la tarjeta SD para obtener las propiedades de los archivos creados. Como se observa en la figura 4.30, se confirmó la creación de 15 archivos de tamaño máximo requeridos, que albergarán la información generada por la unidad sísmica SR04 durante los 30 días programados.

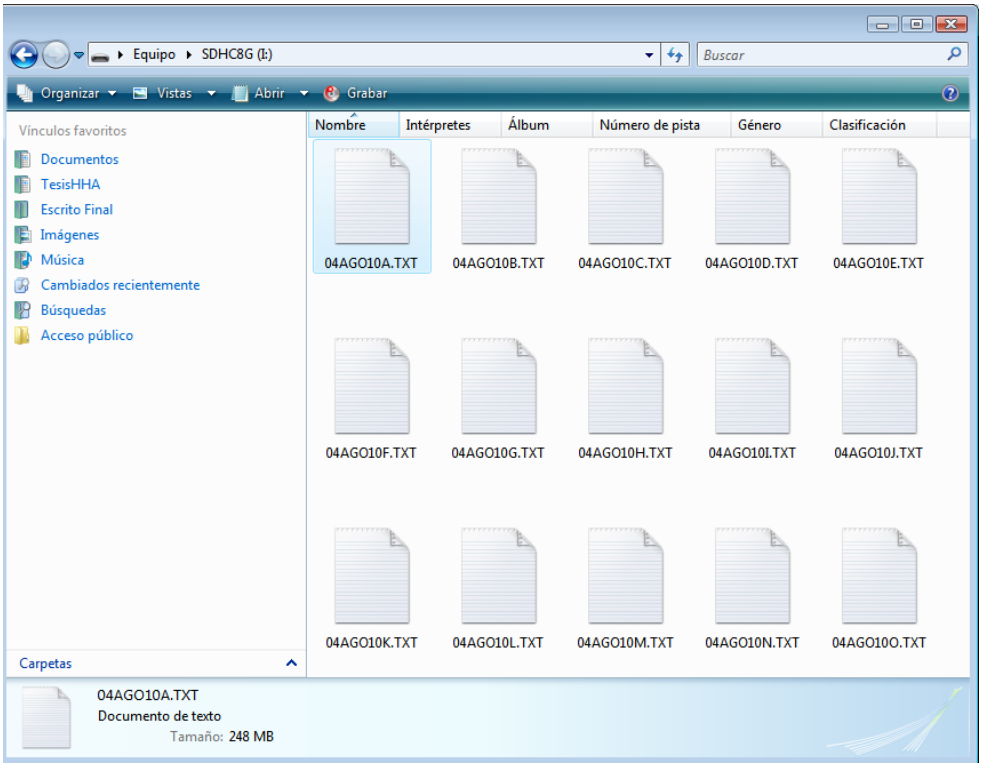


Figura 4.30. Aspecto de los 15 archivos creados para el registro de 30 días.

Cabe recordar que a los archivos se les asigna un nombre correspondiente a la fecha actual, al momento de programar el tiempo de registro, utilizando dos caracteres para el día, tres para el mes, dos para el año y un carácter para diferenciar cada archivo.

4.6.1. Prueba de adquisición y registro de datos provenientes de la unidad sísmica SR04

Una vez que se comprobó que los archivos que se registran por la Interfaz en las tarjetas de memoria SD pueden ser leídos en computadoras compatibles con los sistemas de archivos FAT, se crearon archivos cuyo contenido fue información generada por la unidad SR04.

Para realizar esta prueba se introdujo una señal senoidal con amplitud pico de medio volt y frecuencia de 1 Hz a los tres canales de la tarjeta SADC20, con el fin de registrar una señal de parámetros conocidos y evaluar la información registrada en los archivos creados en las tarjetas de memoria SD.

Se realizaron pruebas de registro para las tres distintas opciones de frecuencia de muestreo. En primer lugar, se llevó a cabo una prueba a 50 mps, en la que se programó a la Interfaz para registrar un archivo de una hora.

Cabe mencionar que se utilizó una aplicación desarrollada en *LabView* para realizar un procesamiento básico a la información registrada en las tarjetas SD. Esta aplicación utiliza algunos módulos de otro software desarrollado en el Instituto de Ingeniería, el denominado ***APSIS***, que permite la adquisición y el procesamiento de los datos recolectados por la unidad sísmica SR04.

Con la aplicación mencionada se lee el archivo que se creó en la tarjeta SD y que contiene la información de la unidad sísmica. La información obtenida es separada por canales, cada canal es decodificado y, finalmente, graficado, ver figura 4.31.

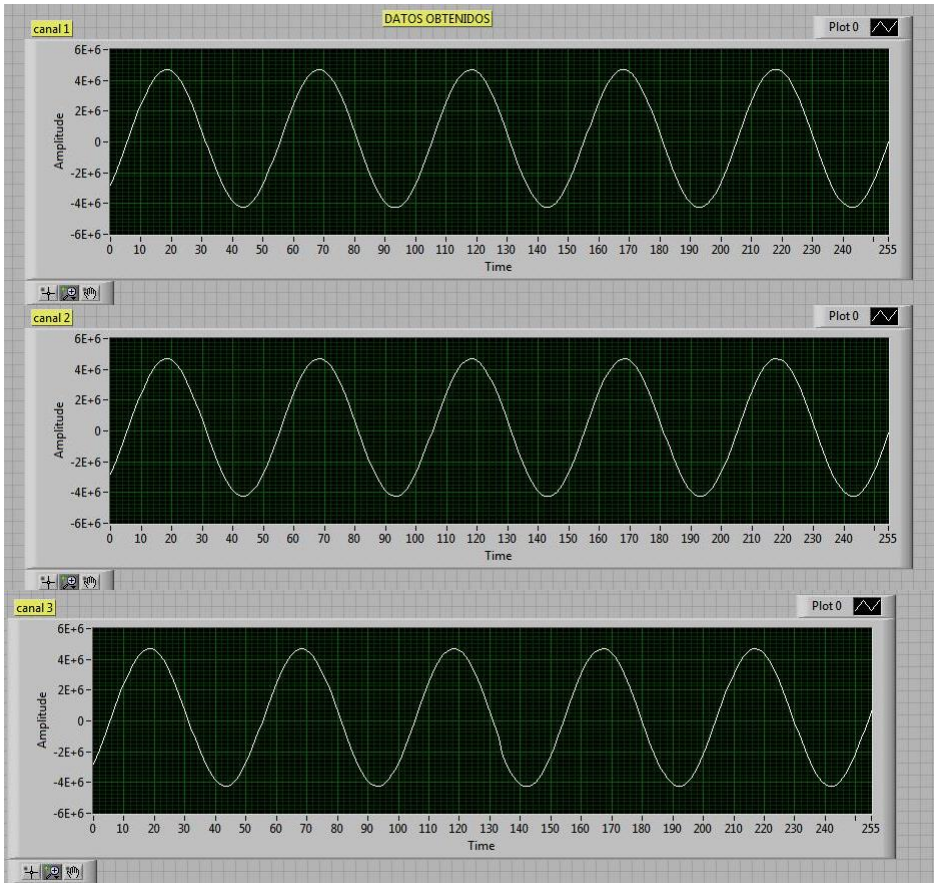


Figura 4.31. Gráficas obtenidas a partir de la información almacenada en las tarjetas SD.

En la imagen anterior se puede observar que las gráficas generadas para los tres canales son prácticamente idénticas, además, se verifica que para esta prueba se obtiene un periodo de la señal en 50 muestras, dado que la frecuencia de dicha señal es de 1 Hz.

En lo que se refiere a la amplitud, al hacer un acercamiento a alguna de las gráficas se puede observar que la señal registrada tiene 4.69 x 10^6 cuentas, aproximadamente. Recordando que cada cuenta equivale a 119 nV, la amplitud de la señal registrada tiene un valor de 0.55 V, que es congruente con el valor de la señal proporcionada por el generador de funciones.

A partir del mismo archivo registrado en la memoria SD se obtuvieron los datos del paquete de tiempo, generando una tabla con dichos datos (ver figura 4.32).

Figura 4.32. Encabezados de tiempo registrados en la tarjeta de memoria SD.

Se comprobó que los datos obtenidos coinciden con la fecha y hora en la cual fue realizada la prueba, además, al dar seguimiento a la columna de segundos se verifica que los valores obtenidos son continuos, por lo que se presume no hay perdidas de información.

Se realizaron pruebas adicionales de registro a 100 y 200 mps, para observar la diferencia en las gráficas que se registran a distinta tasa de muestreo. En las figuras 4.33 y 4.34 se muestran las gráficas para 50 mps, 100 mps y 200 mps.

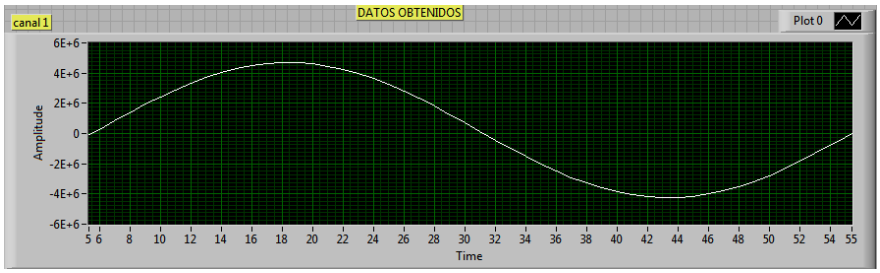
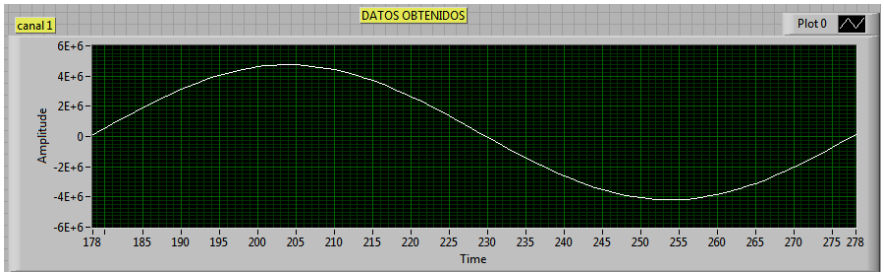
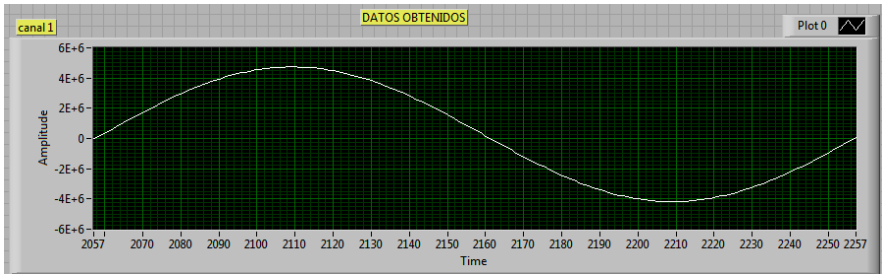


Figura 4.33. Señal registrada a 50 muestras por segundo.



a)



b)

Figura 4.34. Señal registrada a: a) 100 mps. y b) 200 mps.

Después de haber realizado estas pruebas y analizado los resultados obtenidos se pudo comprobar que los datos registrados en las tarjetas de memoria SD son confiables y que no se presentan pérdidas o distorsión en la información recolectada.

4.7. Pruebas integrales a la Interfaz de registro de datos

Hasta este punto se han realizado las pruebas necesarias para evaluar el funcionamiento de los distintos módulos que integran el *hardware* y el *software* de la Interfaz de Registro de datos para la Unidad Sísmica SR04. Por lo anterior, podemos decir que mediante el sistema implementado se puede configurar la frecuencia de muestreo de dicha unidad, además, permite visualizar los datos que obtiene y hace posible registrar estos datos en tarjetas de memoria SD, otorgando autonomía a la unidad SR04.

Mediante la prueba de registro de datos se pretende evaluar el sistema en su etapa final, esperando registrar información real generada por los sensores conectados a los canales de la tarjeta digitalizadora SADC20, durante dos distintos periodos de tiempo, uno de 3 días a 200 mps y otro de 15 días a 200 mps. Primeramente se programó la Interfaz de registro para guardar la información generada durante 3 días. Para esta prueba se utilizó la tarjeta SD marca KINGMAX de 1 GB, con sistema de archivos FAT16.

Después de programar la Interfaz se dejó registrando durante el periodo establecido, tras lo cual se retiro la memoria SD para leer la información registrada en una PC. De acuerdo a lo programado, se crearon 3 archivos de 248 MB. Para leer la información registrada se utilizó la aplicación de LabView mencionada en el apartado 4.6.1. Se obtuvieron gráficas de los tres canales y una tabla con las estampas de tiempo que proporciona el GPS de la unidad sísmica SR04. En la figura 4.35 se pueden muestran las propiedades de los tres archivos generados; en la figura 4.36 se observan las gráficas obtenidas a partir de los datos registrados

en uno de los archivos, en este caso se trata del archivo 13AGO10A.TXT, y en la figura 4.37 se tienen algunos de los datos de tiempo que se decodifican con la aplicación mencionada, dichos datos corresponden al archivo 13AGO10A.TXT y 13AGO10C.TXT.

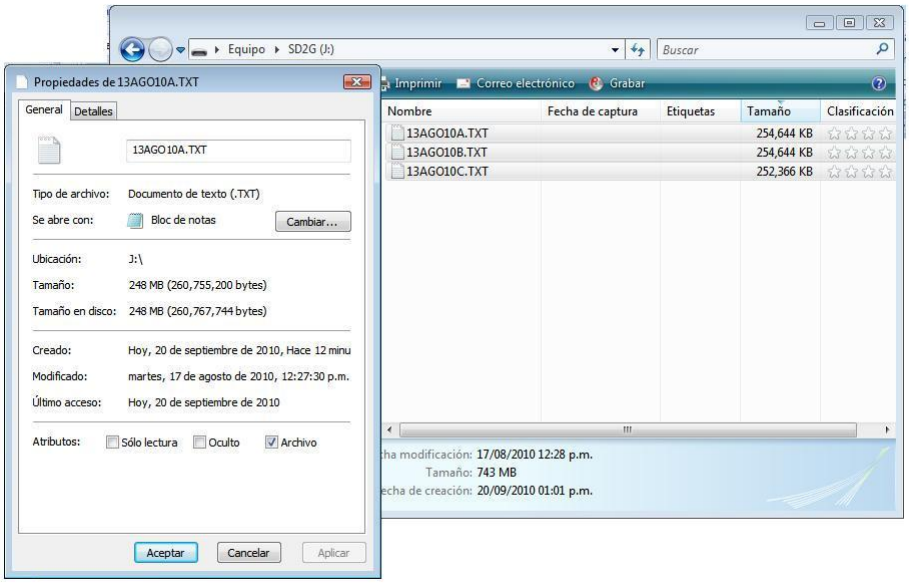


Figura 4.35. Archivos creados para 3 días de registro a 200 mps.

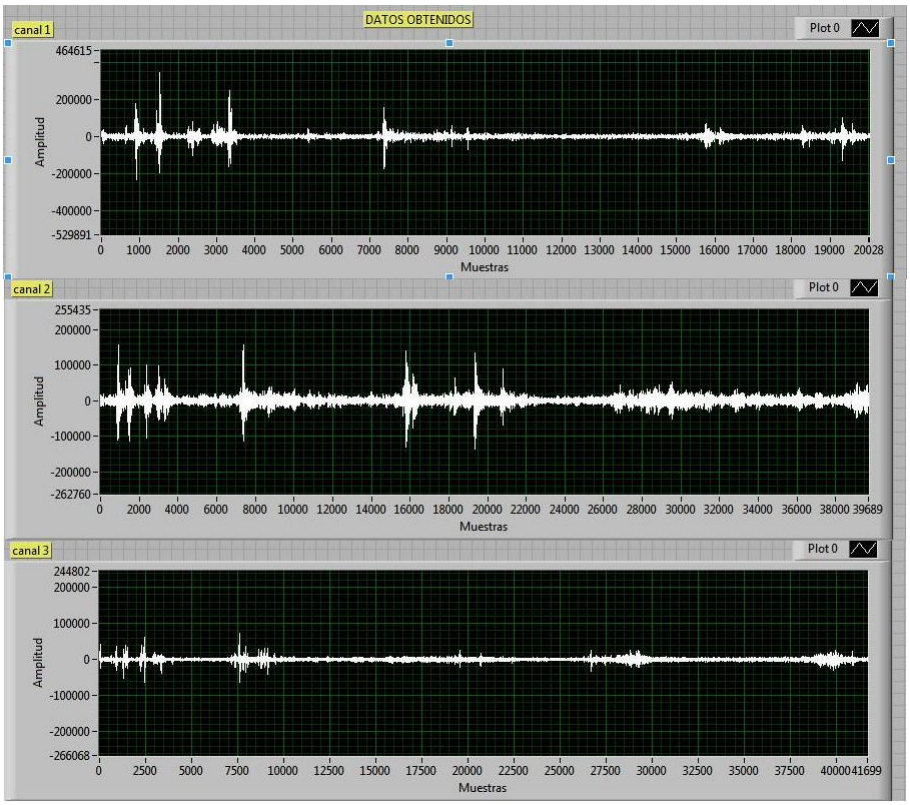


Figura 4.36. Gráficas obtenidas a partir del archivo 13AGO10A.

Encabezados de Tiempo: _

//año	/mes	/dia	/Segundo	/minuto	/Hora
129	10	8	13	32	38	14
129	10	8	13	33	38	14
129	10	8	13	34	38	14
129	10	8	13	35	38	14
129	10	8	13	36	38	14
129	10	8	13	37	38	14
129	10	8	13	38	38	14
129	10	8	13	39	38	14
129	10	8	13	40	38	14
129	10	8	13	41	38	14

a)

Encabezados de Tiempo: _

//año	/mes	/dia	/Segundo	/minuto	/Hora
129	10	8	16	10	38	14
129	10	8	16	11	38	14
129	10	8	16	12	38	14
129	10	8	16	13	38	14
129	10	8	16	14	38	14
129	10	8	16	15	38	14
129	10	8	16	16	38	14
129	10	8	16	17	38	14
129	10	8	16	18	38	14
129	10	8	16	19	38	14
129	10	8	16	20	38	14

b)

Figura 4.37. Estampas de tiempo registradas en los archivos: a) 13AGO10A.TXT b) 13AGO10C.TXT.

Terminada la prueba de los tres días, se realizó la prueba de registro para el periodo de 15 días a 200 mps. Tras lo cual se le realizó el mismo procesamiento que en la prueba anterior. En la figura 4.38 se observan las propiedades de los 15 archivos generados; en la figura 4.39 se presentan los datos de tiempo obtenidos para los archivos 17AGO10A.TXT y

17AGO10O.TXT y, por último, en la figura 4.40 se muestran las gráficas que se obtienen con la información almacenada en el archivo 17AGO10E.TXT.

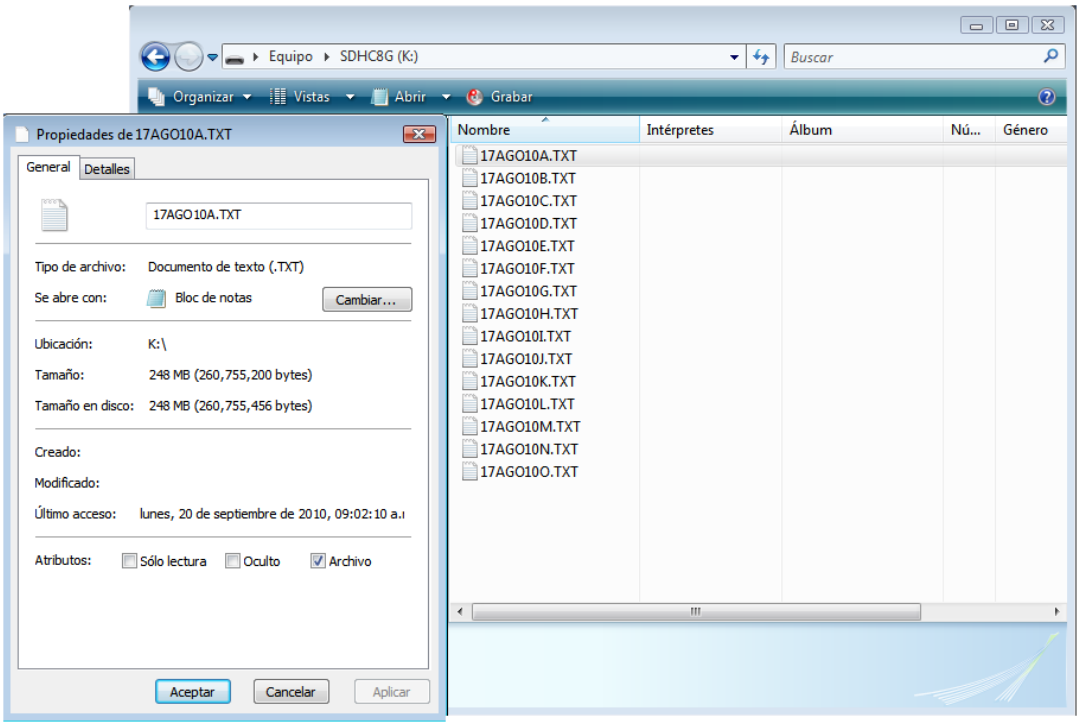


Figura 4.38. Archivos generados para 15 días de registro a 200 mps.

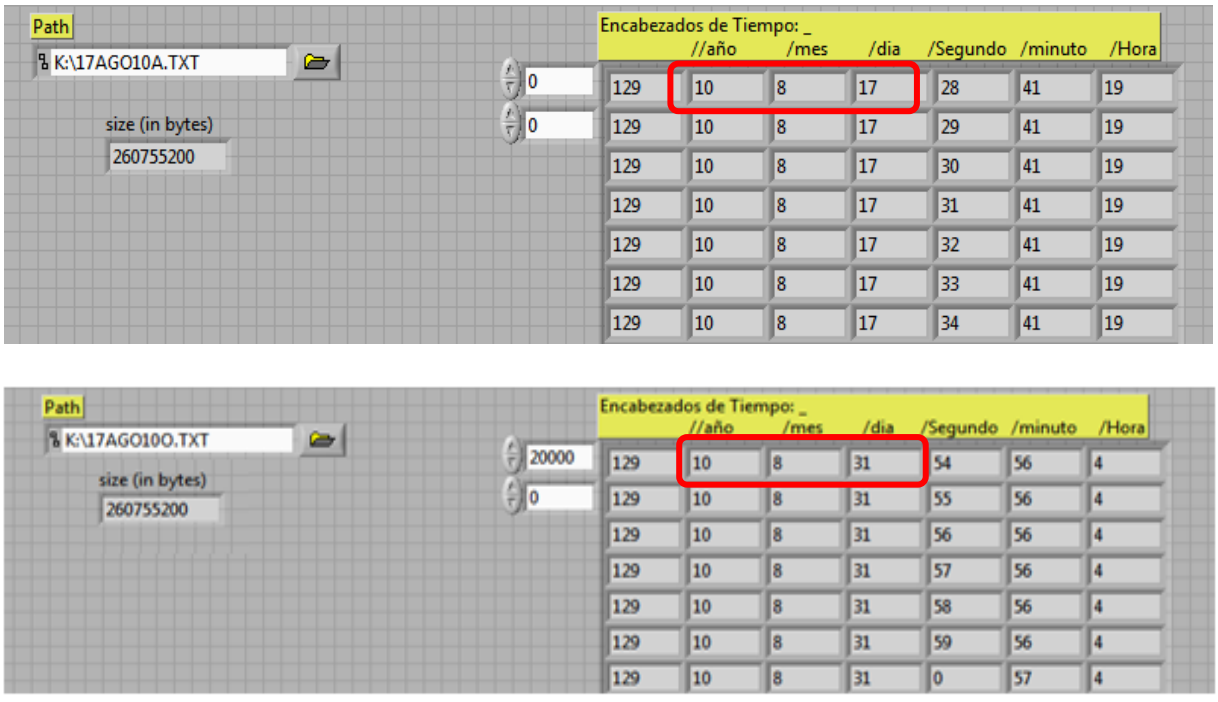


Figura 4.39. Marcas de tiempo extraídas de los archivos 17AGO10A Y 17AGO10O.

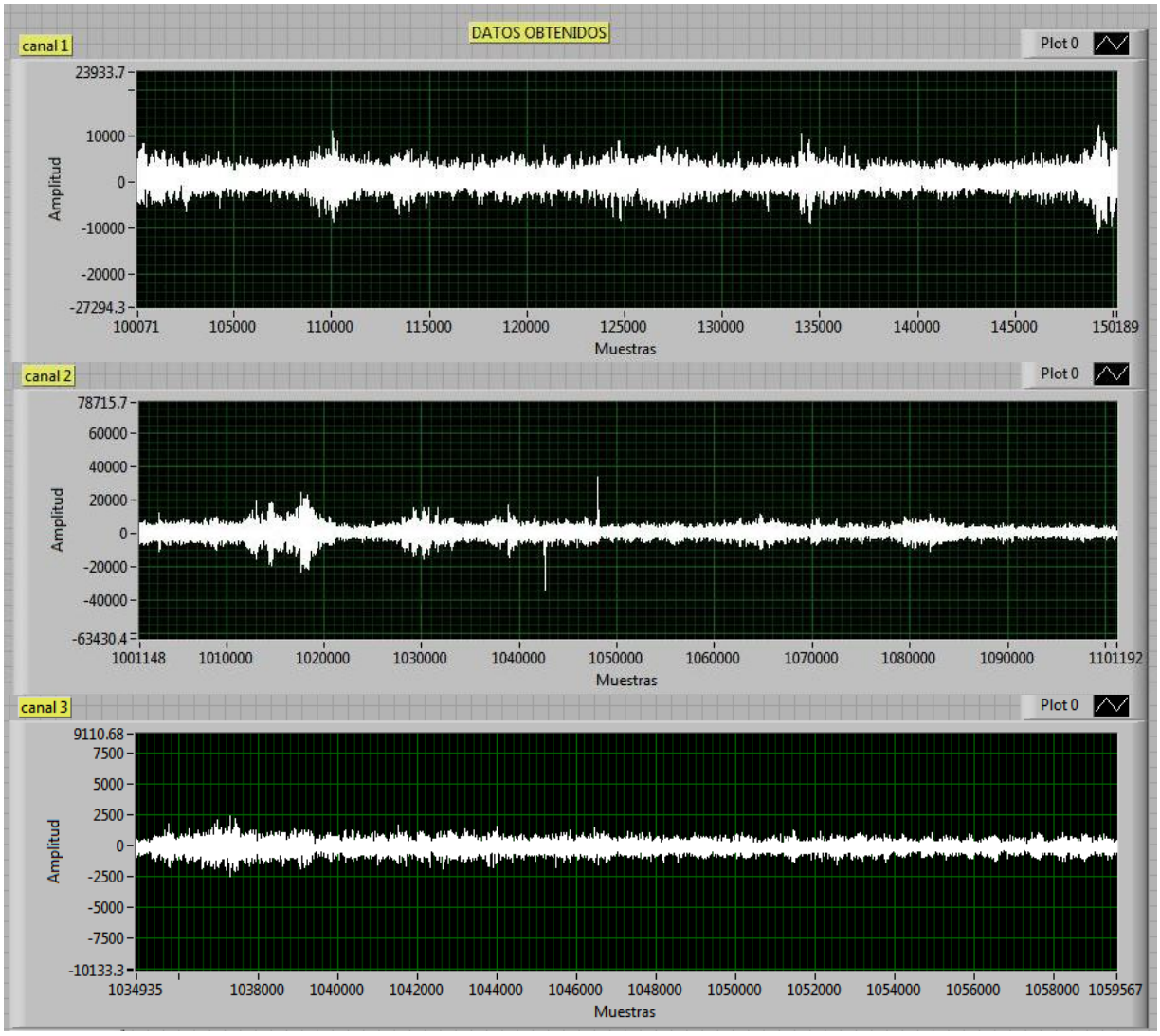


Figura 4.40. Gráficas obtenidas del archivo 17AGO10E.TXT.

En las figuras 4.41 y 4.42 se muestra la última estampa de tiempo del archivo 17AGO10A y la primera del archivo 17AGO10B. Al observar los datos que nos proporcionan dichas figuras podemos verificar la continuidad de los datos registrados.

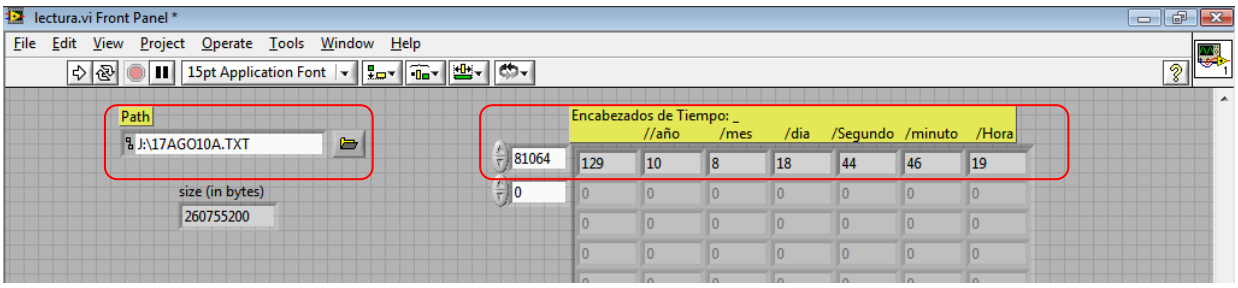


Figura 4.41. Última estampa de tiempo del archivo 17AGO10A.TXT.

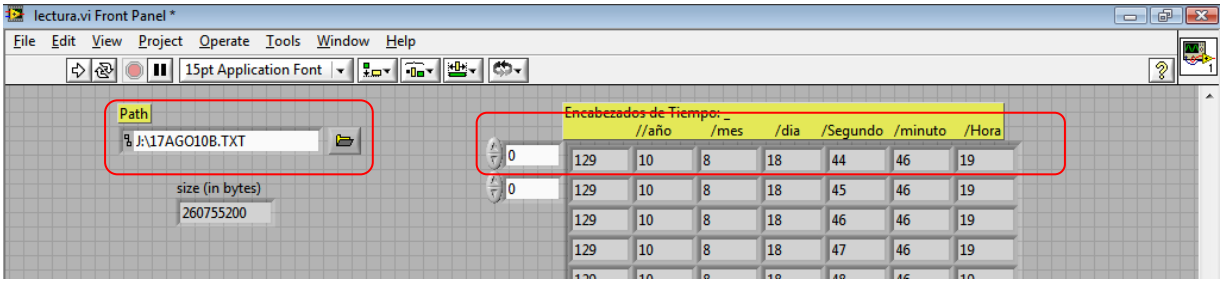


Figura 4.42. Primer estampa de tiempo del archivo 17AGO10B.TXT.

De acuerdo a los resultados obtenidos al realizar estas pruebas de registro, comprobamos que la Interfaz de registro realiza las tareas programadas de manera correcta, creando la cantidad de archivos requerida y registrando la información deseada durante el tiempo designado. En las tarjetas utilizadas se crearon archivos cumpliendo con los sistemas de archivos FAT16 y FAT32. El registro de los datos se realiza escribiendo todos los datos de manera contigua dentro de la región de datos de la tarjeta de memoria SD, por lo que no se presentan pérdidas de datos.

4.8. Tiempo de registro, cantidad de bytes a registrar y tarjetas soportadas

Como parte de las pruebas que se realizaron a la Interfaz de registro, y como información útil al usuario de la misma, se calcularon las distintas cantidades de bytes a registrar, de acuerdo al tiempo de registro y a las tres opciones de frecuencia de muestreo disponible. En los datos presentados en las tablas 4.2 a 4.5 se especifican las tarjetas que soportarían la cantidad de bytes a registrar.

Para realizar las tablas mencionadas, se toma en cuenta la expresión 3.3, con la cual se calcula la cantidad de bytes generados en cierta cantidad de segundos a una tasa de muestreo determina:

$$\boldsymbol{FileSize = \big((15 * mps) + 9\big) * Segs * Num} \qquad \boldsymbol{Ec.3.3}$$

En la expresión anterior:

- **15** = (3 canales) * (5 Bytes por muestra)
- **9 =** son los bytes por segundo de la marca de tiempo
- **mps** = Frecuencia de muestreo.
- **segs** = Cantidad de segundos: 3600 segundos por hora, 86000 segundos por día.
- **Num** = Cantidad de días u horas a registrar (parámetro introducido por el usuario).

A partir de la expresión anterior, se construyen las tablas en las que tiene el tamaño del archivo generado para distintos tiempos de registro en cada una de las opciones de frecuencia posibles.

Las tarjetas de memoria flash SD que se utilizaron en el desarrollo de la Interfaz de registro y que pueden encontrarse de manera común son de las siguientes capacidades:

- 1 GB
- 2 GB
- 4 GB
- 8 GB

Como ya se comentó, se señala en las tablas qué tarjetas de memoria que se pueden utilizar para realizar el registro del tiempo requerido. Cabe comentar que se considera un Megabyte como 2^{20} = 1 048 576 bytes y a un Gigabyte como 2^{30} = 1 073 741 824 bytes.

En la tabla 4.2 se presentan los datos para un registro por horas, en donde se reportan los cálculos hechos para los periodos más representativos en horas.

Horas	Frecuencia de muestreo mps	Tamaño del archivo M B	Tarjetas soportadas			
			1 GB	2 GB	4 GB	8 GB
1	50	2.6	S	S	S	S
6	50	15.6	S	S	S	S
12	50	31.3	S	S	S	S
24	50	62.5	S	S	S	S
48	50	125.1	S	S	S	S
72	50	187.6	S	S	S	S
96	50	250.2	S	S	S	S
120	50	312.7	S	S	S	S
1	100	5.2	S	S	S	S
6	100	31.1	S	S	S	S
12	100	62.2	S	S	S	S
24	100	124.3	S	S	S	S
48	100	248.7	S	S	S	S
72	100	373.0	S	S	S	S
96	100	497.4	S	S	S	S
120	100	621.7	S	S	S	S
1	200	10.3	S	S	S	S
6	200	62.0	S	S	S	S
12	200	124.0	S	S	S	S
24	200	247.9	S	S	S	S
48	200	495.9	S	S	S	S
72	200	743.8	S	S	S	S
96	200	991.7	X	S	S	S
120	200	1239.7	X	S	S	S

Tabla 4.2. Registro en horas, para las frecuencias 50, 100 y 200 mps.

En la tabla 4.3 se reportan los cálculos correspondientes al registro en días, de 1 a 30 días, para 50 mps.

Días	Frecuencia de muestreo mps	Tamaño MB	Tarjetas soportadas			
			1 GB	2 GB	4 GB	8 GB
1	50	62.5	S	S	S	S
2	50	125.1	S	S	S	S
3	50	187.6	S	S	S	S
4	50	250.2	S	S	S	S
5	50	312.7	S	S	S	S
6	50	375.2	S	S	S	S
7	50	437.8	S	S	S	S
8	50	500.3	S	S	S	S
9	50	562.9	S	S	S	S
10	50	625.4	S	S	S	S
11	50	687.9	S	S	S	S
12	50	750.5	S	S	S	S
13	50	813.0	S	S	S	S
14	50	875.6	S	S	S	S
15	50	938.1	X	S	S	S
16	50	1000.6	X	S	S	S
17	50	1063.2	X	S	S	S
18	50	1125.7	X	S	S	S
19	50	1188.3	X	S	S	S
20	50	1250.8	X	S	S	S
21	50	1313.3	X	S	S	S
22	50	1375.9	X	S	S	S
23	50	1438.4	X	S	S	S
24	50	1501.0	X	S	S	S
25	50	1563.5	X	S	S	S
26	50	1626.0	X	S	S	S
27	50	1688.6	X	S	S	S
28	50	1751.1	X	S	S	S
29	50	1813.7	X	S	S	S
30	50	1876.2	X	S	S	S

Tabla 4.3. Registro en días con frecuencia de 50 mps.

En la tabla 4.4 se reportan los cálculos correspondientes al registro en días para la 100 mps, al igual que en la anterior, se calculan los tamaños de archivo de 1 a 30 días.

Días	Frecuencia de muestreo mps	Tamaño	Tarjetas soportadas			
			1 GB	2 GB	4 GB	8 GB
1	100	124.3 MB	S	S	S	S
2	100	248.7MB	S	S	S	S
3	100	373.0 MB	S	S	S	S
4	100	497.4 MB	S	S	S	S
5	100	621.7 MB	S	S	S	S
6	100	746.0 MB	S	S	S	S
7	100	870.4 MB	S	S	S	S
8	100	994.7 MB	X	S	S	S
9	100	1.1 GB	X	S	S	S
10	100	1.2 GB	X	S	S	S
11	100	1.3 GB	X	S	S	S
12	100	1.5 GB	X	S	S	S
13	100	1.6 GB	X	S	S	S
14	100	1.7 GB	X	S	S	S
15	100	1.8 GB	X	X	S	S
16	100	1.9 GB	X	X	S	S
17	100	2.1 GB	X	X	S	S
18	100	2.2 GB	X	X	S	S
19	100	2.3 GB	X	X	S	S
20	100	2.4 GB	X	X	S	S
21	100	2.5 GB	X	X	S	S
22	100	2.7 GB	X	X	S	S
23	100	2.8 GB	X	X	S	S
24	100	2.9 GB	X	X	S	S
25	100	3.0 GB	X	X	S	S
26	100	3.2 GB	X	X	S	S
27	100	3.3 GB	X	X	S	S
28	100	3.4 GB	X	X	S	S
29	100	3.5 GB	X	X	S	S
30	100	3.6 GB	X	X	S	S

Tabla 4.4. Registro en días con frecuencia de 100 mps.

Por último, se presentan los cálculos para el registro de 1 a 30 días a 200 mps en la tabla 4.5, siendo el registro de 30 días a 200 mps el más grande que se puede realizar con la Interfaz desarrollada.

Días	Frecuencia de muestreo mps	Tamaño GB	Tarjetas soportadas			
			1 GB	2 GB	4 GB	8 GB
1	200	0.2	S	S	S	S
2	200	0.5	S	S	S	S
3	200	0.7	S	S	S	S
4	200	1.0	X	S	S	S
5	200	1.2	X	S	S	S
6	200	1.5	X	S	S	S
7	200	1.7	X	S	S	S
8	200	1.9	X	X	S	S
9	200	2.2	X	X	S	S
10	200	2.4	X	X	S	S
11	200	2.7	X	X	S	S
12	200	2.9	X	X	S	S
13	200	3.1	X	X	S	S
14	200	3.4	X	X	S	S
15	200	3.6	X	X	S	S
16	200	3.9	X	X	X	X
17	200	4.1	X	X	X	X
18	200	4.4	X	X	X	X
19	200	4.6	X	X	X	X
20	200	4.8	X	X	X	X
21	200	5.1	X	X	X	X
22	200	5.3	X	X	X	X
23	200	5.6	X	X	X	X
24	200	5.8	X	X	X	X
25	200	6.1	X	X	X	X
26	200	6.3	X	X	X	X
27	200	6.5	X	X	X	X
28	200	6.8	X	X	X	X
29	200	7.0	X	X	X	X
30	200	7.3	X	X	X	X

Tabla 4.5. Registro en días con frecuencia de 200 mps.

4.9. Integración del Hardware en un circuito impreso

Al llegar a la etapa final del desarrollo de la Interfaz, y considerando que no se requería de ningún otro componente de hardware, éste se integró en un circuito impreso (*PCB: Printed Circuit Board*), como primer prototipo funcional, para lo cual se utilizó el software de diseño asistido por computadora *Eagle PCB*. En las figuras 4.43 y 4.44 se muestran los diseños realizados del PCB de la interfaz.

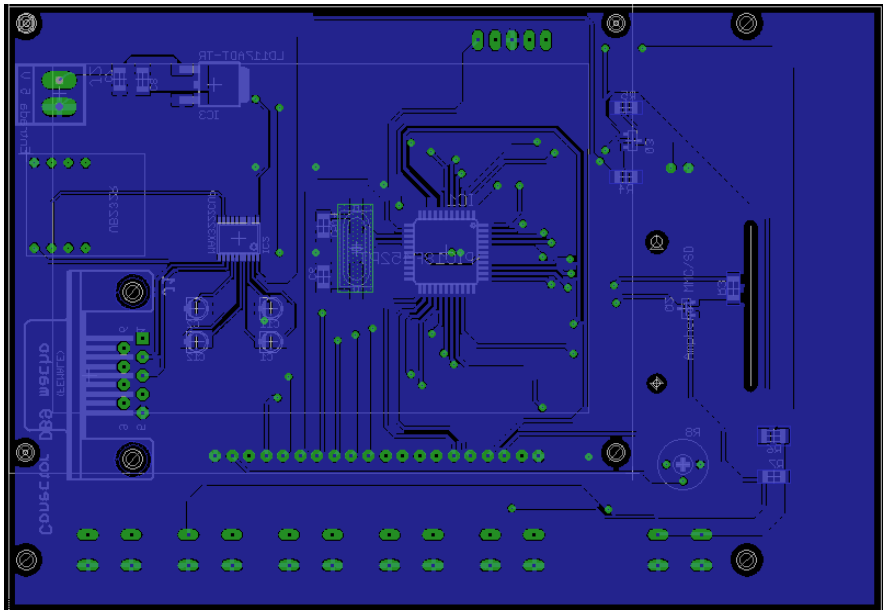


Figura 4.43. Cara inferior del PCB de la interfaz de registro.

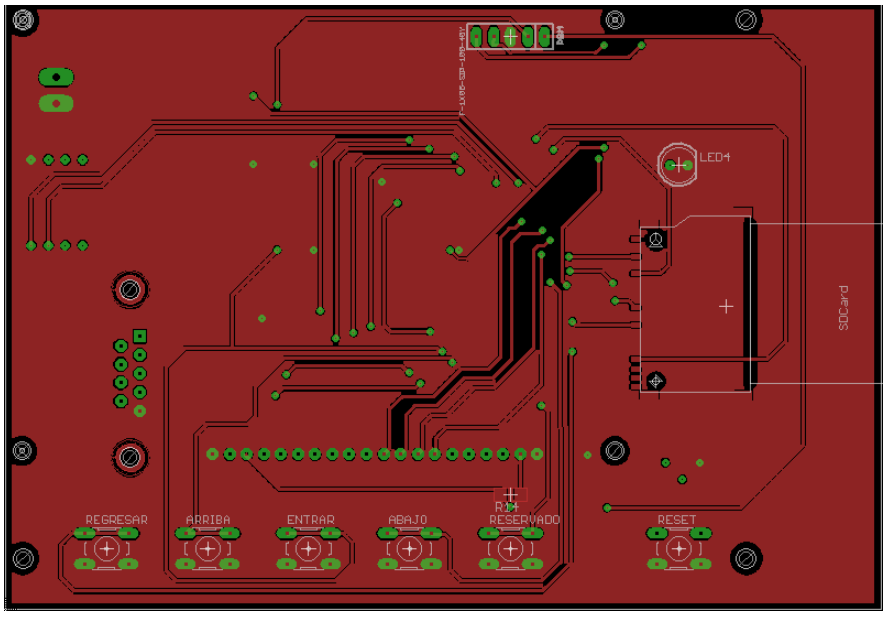


Figura 4.44. Cara superior del PCB de la interfaz de registro.

En el diseño del circuito impreso se tomaron las siguientes consideraciones:

- La intensión de integrar la Interfaz de registro y la unidad sísmica SR04 en el mismo gabinete, como una más de sus tarjetas, por lo que se tomó en cuenta las dimensiones del mencionado gabinete y el espacio disponible.

- La localización de los componentes se realizó pensando en las conexiones que se requieren entre la tarjeta SADC20 y la tarjeta de la unidad SR04.

- En cuanto a los dispositivos con los que interactúa el usuario de la interfaz, como son display gráfico, el conector de las tarjetas SD y el teclado, fueron dispuestos de tal manera que facilitaran dicha interacción y la inserción y extracción de la tarjeta de memoria SD.

- Dada la posibilidad de realizar el PCB en doble cara, se optó por colocar la mayor parte de los componentes en la capa inferior y, en la capa superior colocar únicamente los elementos con los que interactúa el usuario, es decir, el display, las teclas y el conector para la tarjeta SD.

En las figuras siguientes se muestran fotografías de los prototipos fabricados listos para ser utilizados.

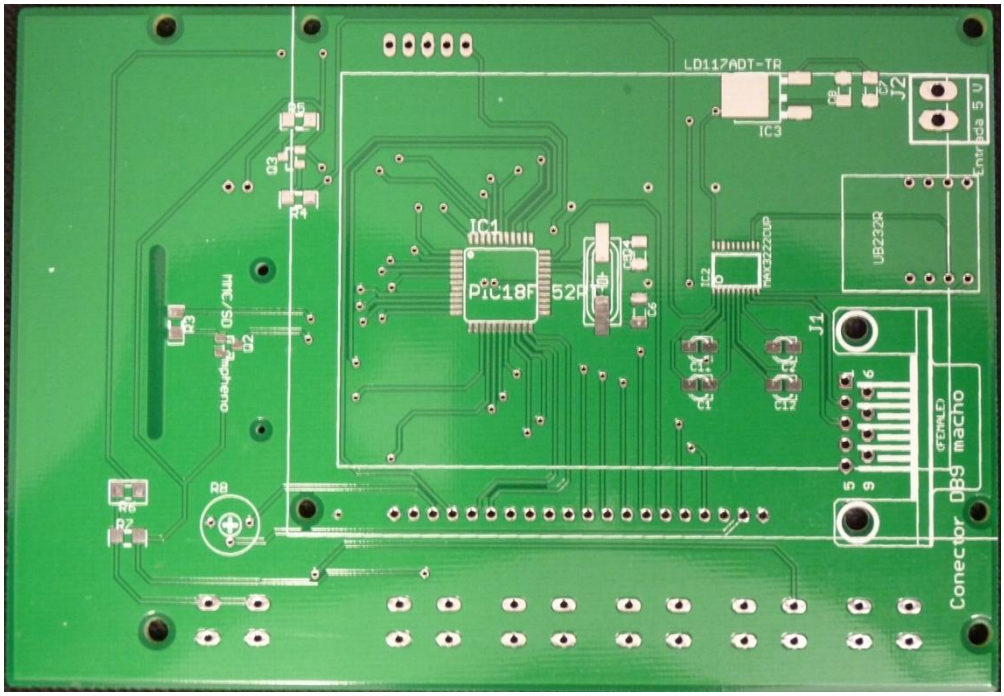


Figura 4.45. Fotografía de la parte inferior de la placa fabricada.

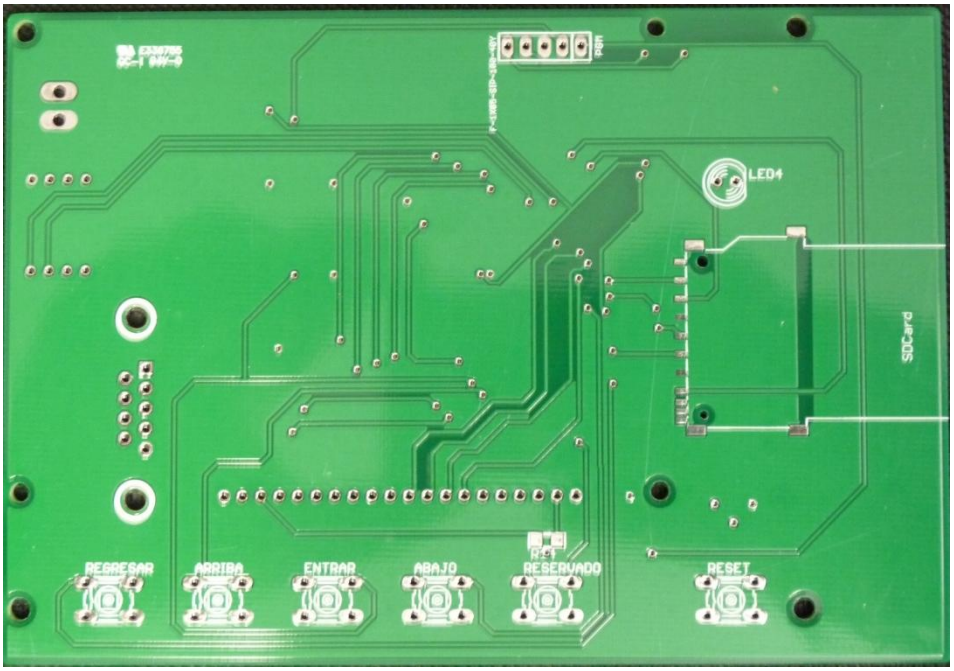


Figura 4.46. Fotografía de la parte superior de la placa fabricada.

Una vez adquirido el material necesario, se soldaron todos los componentes que integran el PCB de la Interfaz de registro, como se puede observar en las figuras 4.47 y 4.48, correspondientes a la parte inferior y a la parte superior del mencionado PCB, respectivamente. Se enmarcan y numeran algunas de las partes que integran a la tarjeta que posteriormente se describen.

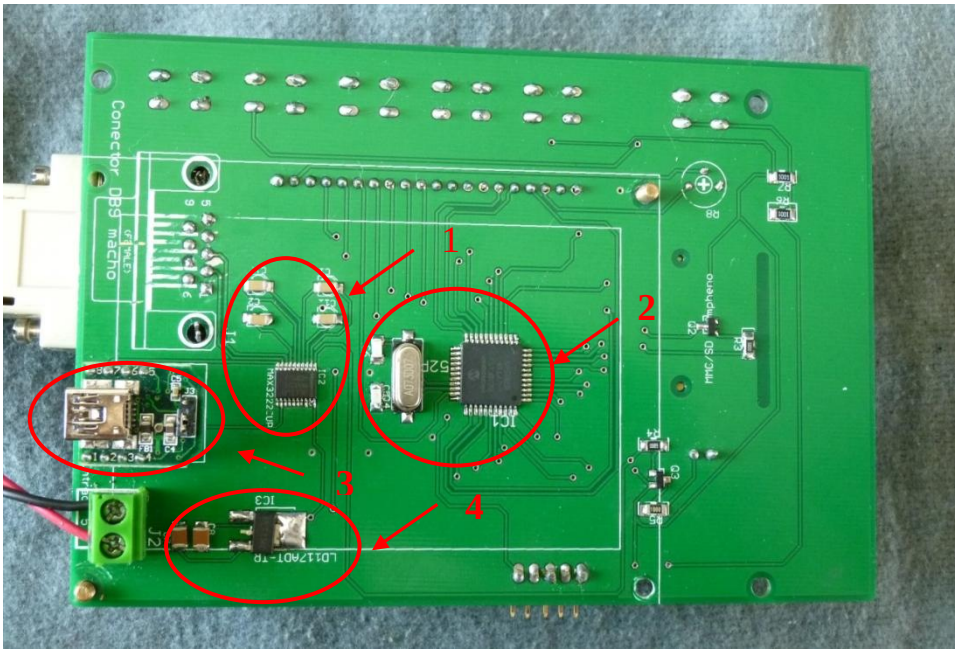


Figura 4.47. Lado inferior de la tarjeta final de la interfaz de registro.

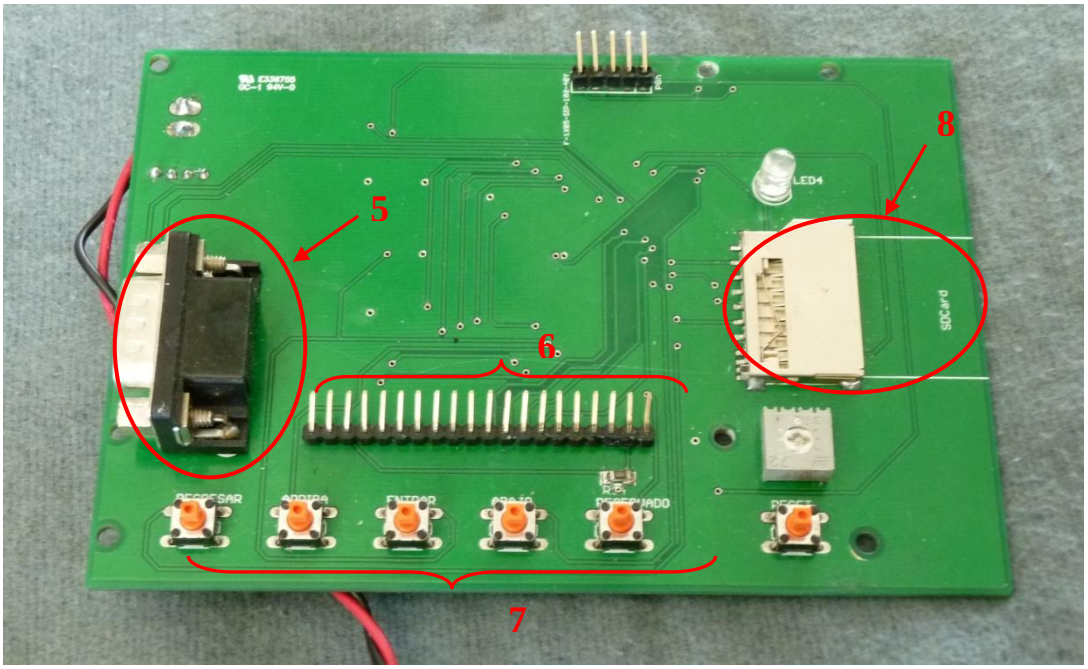


Figura 4.48. Lado superior de la tarjeta final de la Interfaz de registro.

A continuación se nombran las partes señaladas en las figuras anteriores:

1. Transceptor para el estándar RS-232
2. Microcontrolador PIC 18F452
3. Módulo UB 232R para la comunicación USB
4. Regulador de voltaje de 3.3V
5. Conector DB-9 para la comunicación serie RS-232
6. Conector para el display gráfico
7. Teclado
8. Conector para la inserción de las tarjetas de memoria SD

Cabe señalar que al prototipo en circuito impreso se le realizaron las mismas pruebas que al sistema en tarjetas *protoboard*, obteniéndose los mismos resultados.

En las siguientes figuras se muestra el funcionamiento del prototipo en PCB de la Interfaz de registro. En la figura 4.49 se muestra a la Interfaz de registro acoplada a la unidad sísmica SR04. En la figura 4.50 se puede observar el menú principal desplegado en el prototipo de la Interfaz de registro. En la figura 4.51 se presenta una estampa de tiempo obtenida. Por último, en la figura 4.52 se tiene como ejemplo la gráfica del canal 3 de la unidad sísmica SR04.

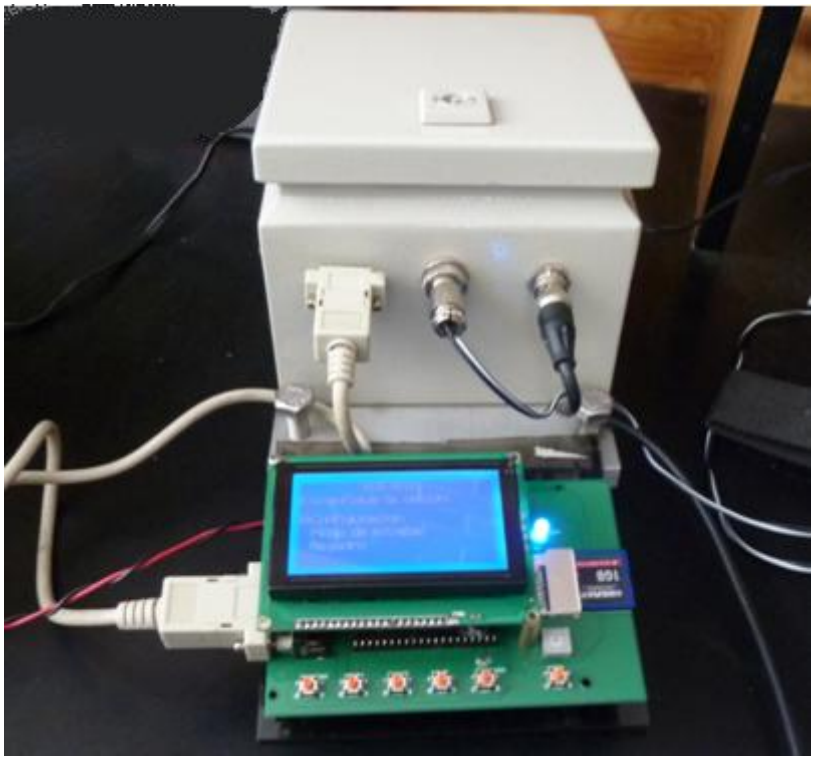


Figura 4.49. Interfaz de registro acoplada a la unidad sísmica SR04.

Figura 4.50. Menú principal de la Interfaz de registro en el prototipo en PCB.

Figura 4.51. Obtención del paquete de tiempo de la unidad sísmica SR04.

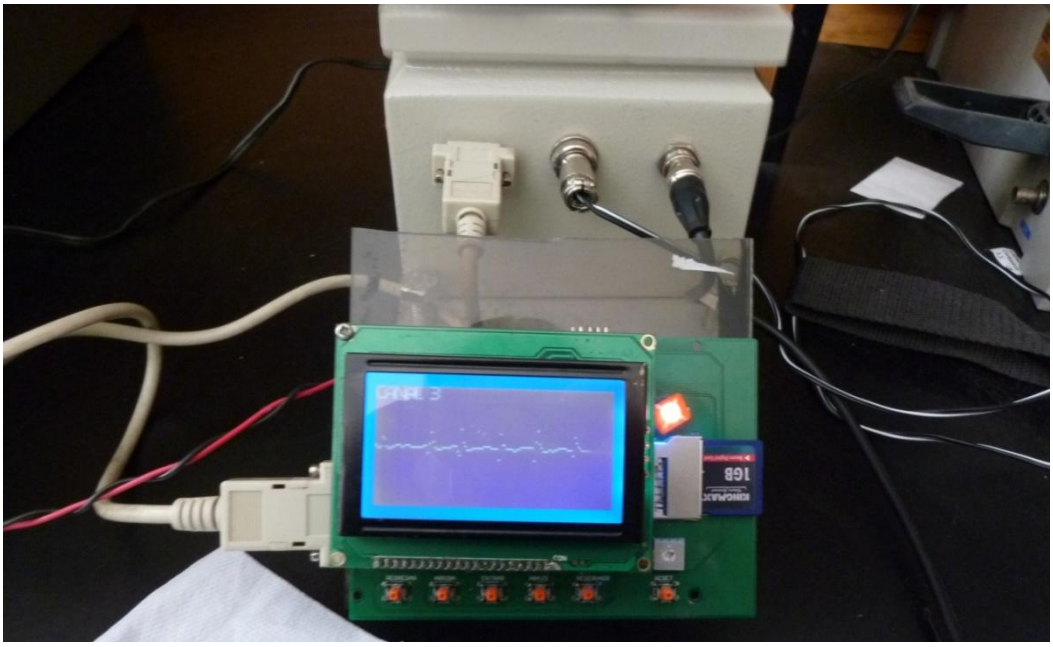


Figura 4.52. Gráfica del canal 2 de la unidad sísmica SR04.

5. RESULTADOS Y CONCLUSIONES

En este capítulo se darán los resultados generales que se obtuvieron a través del desarrollo de la interfaz de registro de datos para la unidad sísmica SR04. También se proporcionarán las conclusiones finales y se señalará una serie de recomendaciones para la mejora y actualización del sistema en cuestión.

5.1. Resultados

Los resultados más importantes obtenidos a partir del desarrollo de la Interfaz de registro son:

- El sistema desarrollado cuenta con un display gráfico y un teclado, que en conjunto permiten seleccionar distintas funciones, como son:
  - Configuración de la frecuencia de muestreo
  - Probar el comportamiento de los sensores conectados a cada canal de la unidad SR04
  - Mostrar la fecha y hora GMT actuales
  - Programar el tiempo en el que la interfaz realizará el registro de datos
- Utilizando la opción de configuración se puede establecer la frecuencia de muestreo a la cual trabajará la unidad sísmica, y escoger entre tres distintos valores de frecuencia de muestreo: 50, 100 y 200 mps.
- Mediante la función de Prueba se puede seleccionar uno de los 3 canales de la unidad SR04, de modo que la interfaz se comunica con dicha unidad, obtiene la información generada en este canal y grafica la forma de onda de la señal obtenida. También se muestra la fecha y la hora actual generada por el GPS.

- A través de la opción de Registro se puede elegir la cantidad de tiempo en el que la interfaz estará registrando los datos provenientes de la unidad sísmica.

- El registro de datos se hace por archivos, los cuales pueden ser leídos por cualquier equipo de cómputo que soporte los sistemas de archivos FAT. La información registrada es confiable y no ocurren pérdidas durante el proceso. Los archivos grabados pueden ser leídos en una PC y mediante un procesamiento básico se puede obtener la información que le es útil a los expertos en el área de sismología.

- El sistema en cuestión es capaz de registrar hasta 30 días de información a 200 mps, requiriendo para ello una tarjeta de memoria de 8 GB. En las tablas 4.2 a 4.5 del capítulo anterior se muestra la relación entre el tiempo de registro, la cantidad de bytes a registrar y las tarjetas soportadas en cada caso.

- La interfaz no intentará registrar una cantidad de bytes más grande que la soportada, indicando en el display que la memoria SD insertada es insuficiente. En los casos en los que la tarjeta en la que se pretende realizar el registro no sea compatible porque tiene formato del sistema de archivos FAT12 o porque no cumple con las especificaciones técnicas en su versión 2.0, se detendrá el proceso de registro y se mostrará un mensaje indicando la causa del error.

- Adicionalmente, la interfaz desarrollada cuenta con un módulo USB mediante el cual se puede comunicar con una PC, sirviendo de enlace entre la unidad sísmica y el equipo de cómputo. A través de dicho módulo, la computadora pude establecer la frecuencia de muestreo a la que trabajará la unidad sísmica.

5.2. Conclusiones

El objetivo del presente trabajo ha sido el diseño y desarrollo de una Interfaz de registro de datos para la unidad sísmica SR04, con el propósito de darle autonomía a dicha unidad para realizar el registro de datos sin necesidad del uso de un equipo de cómputo. Para lograr este objetivo, se propuso el diseño y desarrollo de un sistema basado en microcontrolador, que fuera capaz de configurar la frecuencia de muestreo de la unidad SR04, visualizar el comportamiento de sus canales a través de gráficas trazadas en un display gráfico y registrar la información proveniente de dicha unidad en tarjetas de memoria flash SD, implementando los sistemas de archivos FAT16 y FAT32.

El prototipo final desarrollado permite integrar las funciones mencionadas a la unidad SR04, facilitando la tarea de los especialistas en ingeniería sismológica, al no requerir de un equipo de cómputo para llevar a cabo el registro de datos por grandes periodos de tiempo ni para configurar la frecuencia de muestreo, además de que permite

comprobar el buen funcionamiento de la unidad sísmica SR04 y de los sensores conectados a la misma.

Como conclusión, podemos comentar que se cumplió con el objetivo inicial, al desarrollar un sistema que cubre las necesidades planteadas al inicio de este trabajo.

Finalmente y de manera personal, en el desarrollo de este proyecto tuve la oportunidad de aplicar los conocimientos adquiridos durante la carrera para diseñar y desarrollar un sistema electrónico que le da solución a un problema de ingeniería, en este caso en el campo de la sismología. También, durante el desarrollo del presente proyecto enriquecí mi formación con nuevas experiencias, responsabilidades y grandes retos.

5.3. Recomendaciones

Durante el desarrollo del la Interfaz de registro se hicieron muchas recomendaciones, observaciones y sugerencias, que permitieron hacer cambios para mejorar el diseño inicial; sin embargo, siempre se puede mejorar o actualizar algunos aspectos, mismos que se presentan a continuación.

En lo que respecta al *hardware* algunas propuestas son:

- Utilizar otro microcontrolador con mayor capacidad de procesamiento, con lo cual se podría incrementar el desempeño de algunas de las funciones de *software* o integrar otras, además, algunos de los microcontroladores, como los que se mencionaron en el capítulo 2, tienen periféricos integrados que pueden manejar directamente ciertos dispositivos como son: el display gráfico, la comunicación USB e inclusive las memorias SD. Otro motivo para cambiar de microcontrolador es que el código de las funciones desarrolladas llena casi por completo la memoria de programa, por lo que un aspecto a considerar es que el microcontrolador sustituto cuente con más memoria de programa.

- También podría utilizarse otro tipo de display para tener mejor resolución en las gráficas de los sensores, de tal manera que den una mejor idea de la información para las pruebas que se realizan con la unidad SR04.

- Sería muy recomendable expandir la función del módulo USB, que fue integrado en este trabajo como un elemento extra y de poco impacto en las tareas que desempeña la Interfaz. Dicho módulo fue pensado originalmente para enviar directamente la información registrada en las tarjetas SD hacia la PC y evitar el uso de un lector de tarjetas. Debido a que el módulo USB utilizado funciona como un puerto que sólo adapta la señal serie que genera el microcontrolador a los niveles de voltaje y protocolo USB, la velocidad a la que trabaja este puerto es la programada para las funciones de comunicación con la unidad sísmica, es decir, 38 400 baudios. Por lo

anterior, la transferencia de información sería muy lenta y no convendría utilizar dicho módulo para la función descrita.

En cuanto al software:

- Convendría optimizar el código de los programas desarrollados, con el fin de disminuir el tamaño del programa del microcontrolador, además de hacer más eficiente al mismo.

- De manera ideal, el registro de los datos en las tarjetas SD debería darse registrando los canales por separado, es decir, registrar un archivo por cada canal de la unidad SR04, añadiendo la estampa de tiempo correspondiente, de forma que sea más fácil procesar la información para su análisis en un equipo de cómputo.

GLOSARIO

ADC. *Analog to Digital Converter*. Dispositivo que convierte una señal analógica a una representación digital.

ALU. *Arithmetic and Logic Unit*. Componente de la unidad central de procesamiento encargada de realizar las operaciones aritméticas y también de las operaciones lógicas.

ASCII. *American Standard Code for Information Interchange*. Código de caracteres, que permite representar numéricamente el texto en equipos de cómputo.

BIOS. *Basic Input Output System*. Código de sistema que se ejecuta para iniciar el sistema operativo de una PC.

Bus. Conductor o conjunto de conductores común a varios dispositivos que permite distribuir corrientes de alimentación.

Clúster. Unidad de asignación según la terminología de *Microsoft*, es un conjunto contiguo de sectores que componen la unidad más pequeña de almacenamiento de un disco.

CPU. *Central Processing Unit*. Es la unidad central de procesamiento en un sistema de cómputo.

CRC: Cyclic Redundancy Code. Código de detección de errores en la transmisión de información entre la tarjeta de memoria SD y el microcontrolador.

Display. Dispositivo de ciertos aparatos electrónicos, como los teléfonos y las calculadoras, destinado a la representación visual de información.

EEPROM. *Electrically-Erasable Programmable Read-Only Memory*. Memoria no volátil de solo lectura, que se puede borrar y programar eléctricamente.

FAT. *File Allocation Table*. Nombre de los sistemas de archivos de los sistemas operativos Windows que utilizan la tabla de asignación de archivos.

Firmware. Es el código fijo de un sistema electrónico que contiene las tareas programadas para el mismo.

GMT. *Greenwich Mean Time*. Es el tiempo medio solar en el Observatorio Real de Greenwich, en Greenwich Inglaterra, que por convención se encuentra en el meridiano cero.

GPS. *Global Positioning System*. Es un sistema global de navegación basado en satélites que permite localizar la posición de un objeto.

Handshake. Señal dentro de un protocolo de comunicación entre dispositivos, que permite a éstos verificar si el otro está listo para iniciar la transferencia de datos.

Hardware. Conjunto de los componentes que integran la parte material de un sistema.

Host. Dispositivo anfitrión que provee servicios a otros dispositivos con los que se comunica.

I2C. *Inter-Integrated Circuit*. Protocolo de comunicación serie, utilizado para comunicar dispositivos a través de dos terminales.

MBR: *Master Boot Record*. Primer sector localizado en unidades de almacenamiento, tales como discos duros. Contiene código para inicializar al equipo de cómputo anfitrión.

MSSP. Master *Synchronous Serial Port*. Módulo de un microcontrolador que permite la comunicación a través del bus SPI y también por I2C.

NMEA. Es un protocolo de la *National Marine Electronics Association* que permite comunicar dispositivos marinos con sistemas GPS.

OCR. *Operation Condition Register*. Registro de la tarjeta de memoria SD que permite conocer las condiciones de operación de dicha tarjeta.

RAM. *Random Access Memory*. Memoria volátil de acceso aleatorio.

ROM. *Read Only Memory.* Memoria no-volátil de solo lectura.

SDHC. *SD High Capacity*. Nomenclatura que hace referencia a las tarjetas de memoria flash *Secure Digital* de con capacidad de almacenamiento de 2 a 32 GB.

SO. *Sistema Operativo.* Conjunto de programas para el funcionamiento y explotación de un equipo de cómputo, encargado de controlar la unidad central de procesamiento, la memoria y los dispositivos de entrada y salida.

Software. Conjunto de programas, instrucciones y reglas informáticas para ejecutar tareas en una computadora.

SPI. *Serial Peripheral Interface*. Protocolo de comunicación serie entre un dispositivo maestro y uno esclavo que utilizan cuatro terminales para transmitir información entre ellos.

USART. *Universal Synchronous Asynchronous Receiver Transmitter*. Módulo que proporciona al microcontrolador la capacidad para comunicarse a través de las interfaces seriales RS–485 y RS–232.

USB. *Universal Serial Bus*. Protocolo de comunicación serie para transferir datos entre dispositivos.

ANEXOS

HOJAS DE ESPECIFICACIONES:

- ***SD Specifications Part 1: Physical Layer Simplified Specification Version 2.00***
- **Microsoft Extensible Firmware Initiative FAT32 File System Specification**
- **SR04 24 bit Seismic Unit**
- **PIC18F452 Datasheet**
- **GLCD JHD12864**
- **LF33CV Voltage regulator**
- **MAX3222 RS-232 transceiver**
- **UB 232 –R USB – Serial Development module**

SD Specifications

Part 1

Physical Layer

Simplified Specification

Version 2.00

September 25, 2006

SD Group
Matsushita Electric Industrial Co., Ltd. (Panasonic)
SanDisk Corporation
Toshiba Corporation

Technical Committee
SD Card Association

Microsoft

Hardware White Paper

Designing Hardware for Microsoft® Operating Systems

Microsoft Extensible Firmware Initiative FAT32 File System Specification

FAT: General Overview of On-Disk Format

Version 1.03, December 6, 2000
Microsoft Corporation

The FAT (File Allocation Table) file system has its origins in the late 1970s and early1980s and was the file system supported by the Microsoft® MS-DOS® operating system. It was originally developed as a simple file system suitable for floppy disk drives less than 500K in size. Over time it has been enhanced to support larger and larger media. Currently there are three FAT file system types: FAT12, FAT16 and FAT32. The basic difference in these FAT sub types, and the reason for the names, is the size, in bits, of the entries in the actual FAT structure on the disk. There are 12 bits in a FAT12 FAT entry, 16 bits in a FAT16 FAT entry and 32 bits in a FAT32 FAT entry.

Contents

Microsoft, MS_DOS, Windows, and Windows NT are trademarks or registered trademarks of Microsoft Corporation in the United States and/or other countries. Other product and company names mentioned herein may be the trademarks of their respective owners.

© 2000 Microsoft Corporation. All rights reserved.

SEISMOSTACK seismic equipment boards and instruments series

SR04 - 24 BIT SEISMIC UNIT

The SR04 is a rugged and compact seismic digitizer designed to recording signals produced by a wide range of sensors from geophones to accelerometers.

It can be equipped with up to three 4.5Hz geophones.

Overview

The SR04 is an instrument with all the electronic components needed to digitize an analogue seismic signal in digital form to be recorded using a PC. It embeds a sigma-delta high speed and high precision analogue to digital converter, a GPS receiver for accurate timekeeping and sincronization and can embeds 1 or 3 compact 4.5Hz sensors.

The system is intended to be used with a personal computer running the proper data logging software. The native software for this unit is Seismowin and Seismowin-Pro. The unit is also supported by Seislog and Seiscomp.

Technical features

Analogue channels:	3
Anti alias filter:	1 poles 20Hz low-pass filter
Band-pass:	standard DC to 8.8Hz (customizable)
A/D converter:	24 bit sigma delta
Type:	Differential input
Gain:	fixed gain
Input range:	+/- 1V
Overvoltage protec.:	zener diode up to 1kV for few mS
Damping:	geophones internally damped, external sensor damping with external resistors
Input Impedance:	Typically >= 1 Mohm
Noise level:	typically < 2.5 counts at 100 SPS
Crosstalk rejection:	> 140dB
Skew time:	zero (simultaneously sampling on all 3 channel)
Dynamic range:	140dB at 25 SPS
Clock:	10ppm stability within -20/+50 °C temperature range
Precision:	better than 1ppm at 25 °C
Sincronization:	GPS receiver included
GPS Antenna:	Amplified antenna with 10mt of coaxial cable and BNC connector
Comunication:	1 RS232 port at 38400 baud
Protocol:	binary proprietary supported by SEISLOG and SEISMOWIN
Sample frequency:	10, 20, 25, 50, 100, 200 SPS
Power supply:	10-25Vdc - 4.5W
Operating temp.:	-20/+70 °C
Cabling:	RS232 and power cable provided with the unit
Weight:	1,5 Kg

Applications

- Seismic networks
- Seismic arrays
- Personal Seismograph
- Training

SARA di Mariotti Gabriele & C s.n.c - 06129 – Perugia – Via A.Mercuri, 4 – ITALY
Tel. +39 075 5051014 Fax +39 075 5006315 www.sara.pg.it info@sara.pg.it

PIC18FXX2
Data Sheet

High-Performance, Enhanced Flash Microcontrollers with 10-Bit A/D

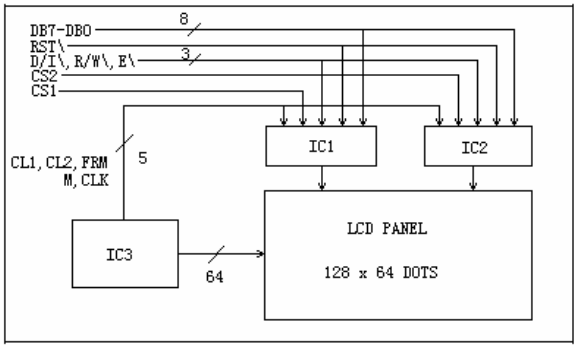
© 2006 Microchip Technology Inc. DS39564C

JHD12864A

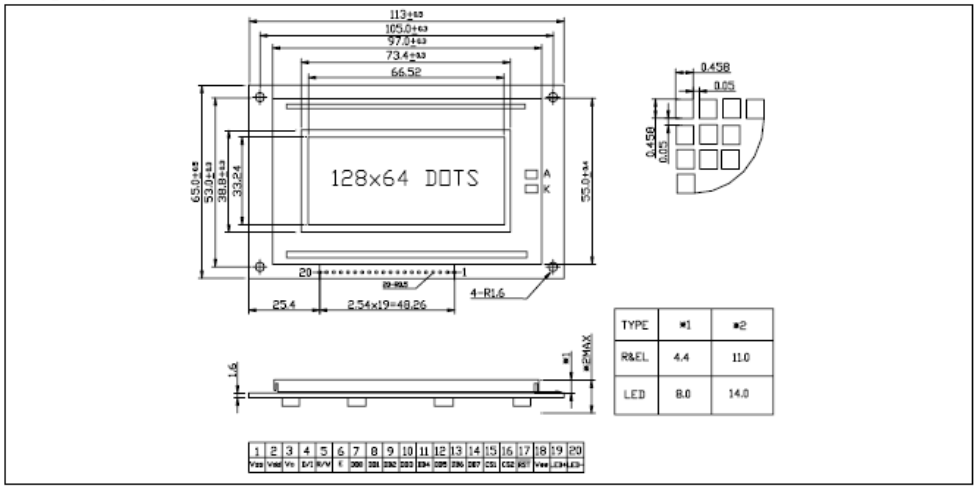
CHARACTERISTICS

Display Content:128*64 dots
Driving Mode: 1/64D
Available Types:
- STN (Yellow Green, Grey, B/W)
- Reflective with El or Led
- El/100VAC, 400HZ
- LED/4.2VDC

APPLICATION CIRCUIT



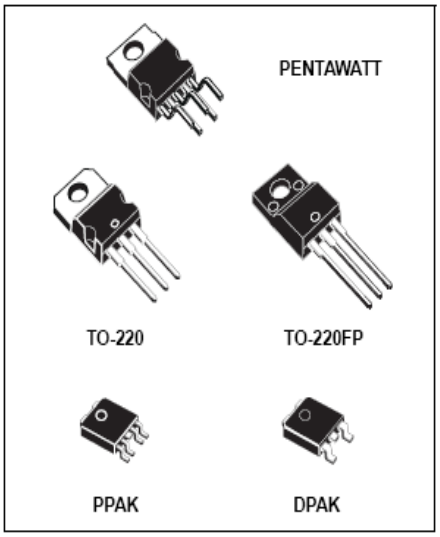
Display Content



LF00 SERIES

VERY LOW DROP VOLTAGE REGULATORS WITH INHIBIT

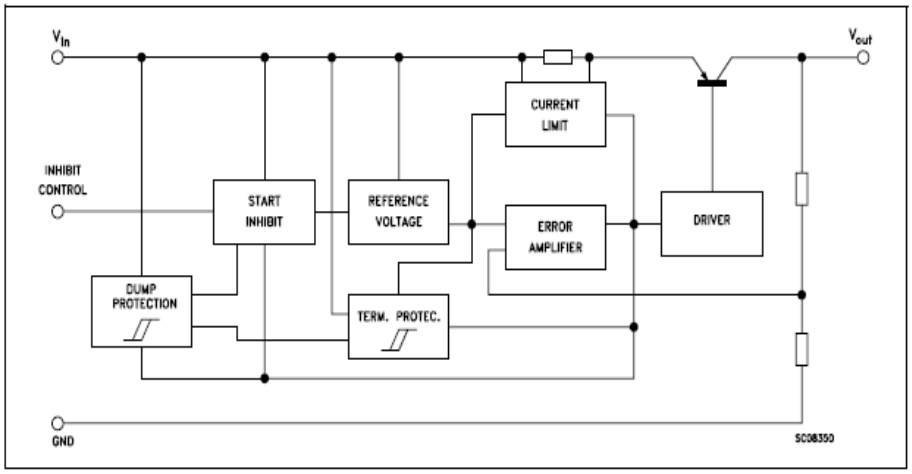
- VERY LOW DROPOUT VOLTAGE (0.45V)
- VERY LOW QUIESCENT CURRENT (TYP. 50 µA IN OFF MODE, 500 µA IN ON MODE)
- OUTPUT CURRENT UP TO 500 mA
- LOGIC-CONTROLLED ELECTRONIC SHUTDOWN
- OUTPUT VOLTAGES OF 1.25; 1.5; 1.8; 2.5; 2.7; 3; 3.3; 3.5; 4; 4.5; 4.7; 5; 5.2; 5.5; 6; 8; 8.5; 9; 12V
- INTERNAL CURRENT AND THERMAL LIMIT
- ONLY 2.2 µF FOR STABILITY
- AVAILABLE IN ± 1% (AB) OR ± 2% (C) SELECTION AT 25 °C
- SUPPLY VOLTAGE REJECTION: 80db (TYP.)
- TEMPERATURE RANGE: -40 TO 125 °C



DESCRIPTION

The LF00 series are very Low Drop regulators available in PENTAWATT, TO-220, TO-220FP, DPAK and PPAK package and in a wide range of output voltages.
The very Low Drop voltage (0.45V) and the very low quiescent current make them particularly suitable for Low Noise, Low Power applications and specially in battery powered systems.
In the 5 pins configuration (PENTAWATT and PPAK) a Shutdown Logic Control function is available (pin 2, TTL compatible). This means that when the device is used as a local regulator, it is possible to put a part of the board in standby, decreasing the total power consumption. In the three terminal configuration the device has the same electrical performance, but is fixed in the ON state. It requires only a 2.2 µF capacitor for stability allowing space and cost saving.

SCHEMATIC DIAGRAM



19-0273; Rev 7; 1/07

MAXIM

3.0V to 5.5V, Low-Power, up to 1Mbps, True RS-232 Transceivers Using Four 0.1µF External Capacitors

General Description

The MAX3222/MAX3232/MAX3237/MAX3241 transceivers have a proprietary low-dropout transmitter output stage enabling true RS-232 performance from a 3.0V to 5.5V supply with a dual charge pump. The devices require only four small 0.1µF external charge-pump capacitors. The MAX3222, MAX3232, and MAX3241 are guaranteed to run at data rates of 120kbps while maintaining RS-232 output levels. The MAX3237 is guaranteed to run at data rates of 250kbps in the normal operating mode and 1Mbps in the MegaBaud™ operating mode, while maintaining RS-232 output levels.

The MAX3222/MAX3232 have 2 receivers and 2 drivers. The MAX3222 features a 1µA shutdown mode that reduces power consumption and extends battery life in portable systems. Its receivers remain active in shutdown mode, allowing external devices such as modems to be monitored using only 1µA supply current. The MAX3222 and MAX3232 are pin, package, and functionally compatible with the industry-standard MAX242 and MAX232, respectively.

The MAX3241 is a complete serial port (3 drivers/5 receivers) designed for notebook and subnotebook computers. The MAX3237 (5 drivers/3 receivers) is ideal for fast modem applications. Both these devices feature a shutdown mode in which all receivers can remain active while using only 1µA supply current. Receivers R1 (MAX3237/MAX3241) and R2 (MAX3241) have extra outputs in addition to their standard outputs. These extra outputs are always active, allowing external devices such as a modem to be monitored without forward biasing the protection diodes in circuitry that may have VCC completely removed.

The MAX3222, MAX3237, and MAX3241 are available in space-saving TSSOP and SSOP packages.

Applications

Notebook, Subnotebook, and Palmtop Computers

High-Speed Modems

Battery-Powered Equipment

Hand-Held Equipment

Peripherals

Printers

Typical Operating Circuits appear at end of data sheet.

Next Generation Device Features

- **For Smaller Packaging: MAX3228E/MAX3229E: +2.5V to +5.5V RS-232 Transceivers in UCSP™**
- **For Integrated ESD Protection: MAX3222E/MAX3232E/MAX3237E/MAX3241E*/ MAX3246E: ±15kV ESD-Protected, Down to 10nA, 3.0V to 5.5V, Up to 1Mbps, True RS-232 Transceivers**
- **For Low-Voltage or Data Cable Applications: MAX3380E/MAX3381E: +2.35V to +5.5V, 1µA, 2 Tx/2 Rx RS-232 Transceivers with ±15kV ESD-Protected I/O and Logic Pins**

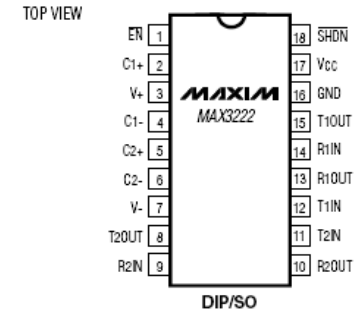
Ordering Information

PART	TEMP RANGE	PIN-PACKAGE	PKG CODE
MAX3222CUP+	0°C to +70°C	20 TSSOP	U20+2
MAX3222CAP+	0°C to +70°C	20 SSOP	A20+1
MAX3222CWN+	0°C to +70°C	18 SO	W18+1
MAX3222CPN+	0°C to +70°C	18 Plastic Dip	P18+5

+Denotes lead-free package.

Ordering Information continued at end of data sheet.

Pin Configurations



Pin Configurations continued at end of data sheet.

MegaBaud and UCSP are trademarks of Maxim Integrated Products, Inc.
**Covered by U.S. Patent numbers 4,636,930; 4,679,134; 4,777,577; 4,797,899; 4,809,152; 4,897,774; 4,999,761; and other patents pending.*

For pricing, delivery, and ordering information, please contact Maxim/Dallas Direct! at 1-888-629-4642, or visit Maxim's website at www.maxim-ic.com.

Future Technology Devices International Ltd.

UB232R

USB Mini-B FT232R Evaluation Module

Datasheet

Document Reference No.: FT_000055

Version 1.02

Issue Date: 2010-02-05

Future Technology Devices International Ltd (FTDI)

Unit 1, 2 Seaward Place, Centurion Business Park, Glasgow, G41 1HH, United Kingdom

Tel.: +44 (0) 141 429 2777 Fax: + 44 (0) 141 429 2758

E-Mail (Support): support1@ftdichip.com

Web: http://www.ftdichip.com

Neither the whole nor any part of the information contained in, or the product described in this manual, may be adapted or reproduced in any material or electronic form without the prior written consent of the copyright holder. This product and its documentation are supplied on an as-is basis and no warranty as to their suitability for any particular purpose is either made or implied. Future Technology Devices International Ltd will not accept any claim for damages howsoever arising as a result of use or failure of this product. Your statutory rights are not affected. This product or any variant of it is not intended for use in any medical appliance, device or system in which the failure of the product might reasonably be expected to result in personal injury. This document provides preliminary information that may be subject to change without notice. No freedom to use patents or other intellectual property rights is implied by the publication of this document. Future Technology Devices International Ltd, Unit 1, 2 Seaward Place, Centurion Business Park, Glasgow, G41 1HH, United Kingdom. Scotland Registered Number: SC136640

BIBLIOGRAFÍA

Libros

[1] Boylestad R., Nashelsky L., *Electrónica: Teoria de circuitos y dispositivos electrónicos*, Octava edición, Pearson Prentice Hall, México, 2003.

[2] García E., *Compilador C CCS y simulador PROTEUS para microcontroladores PIC*, primera edición, Ed. Alfaomega, México 2008.

[3] Kernighan W., Ritchie M., *El lenguaje de programación C*, segunda edición, Pearson Prentice Hall, México, 1991.

[4] Palacios E., Remiro F., López L., *Microcontrolador PIC16F84 Desarrollo de Proyectos*, primea edición, Ed. Alfaomega, México 2004.

[5] Stallings W., *Operating Systems: Internals and Design Principles*, Ed. Prentice Hall, Estados Unidos, 2009.

Artículos y Tesis

[1] Hernández A., *Registro de datos en tarjetas de memoria SD CARD implementando los sistemas de archivos FAT16 y FAT32*, Tesis de licenciatura, UNAM, 2009.

[2] Stoffregen P., *Understanding FAT32 File Systems*: http://www.pjrc.com/tech/8051/ide/fat32.html

[3] Pavan P., Bez R., Olivo P., Zanoni E., *Flash Memory Cells—An Overview*, Proceedings of the IEEE, vol. 85, no. 8, august 1997.

[4] Santiago L., Hernández H., Hernández A., *Diseño e implementación de una interfaz de registro de datos en tarjetas de Memoria flash SD Card para la unidad sísmica SR04*, Memorias del Congreso, V Semana Nacional de Ingeniería Electrónica 2009, ISBN 978-607-477-073-5.

[5] Tapia D., *Adquisición y procesamiento de datos provenientes de la unidad sísmica SR04*, Tesis de licenciatura, UNAM, 2009.

Manuales y Hojas de especificaciones

[1] *3.0V to 5.5V, Low-Power, Up to 1Mbps, True RS-232 Transceivers Using Four 0.1μF External Capacitors (MAX3222) datasheet:* http://www.datasheetcatalog.org/datasheet/maxim/MAX3222-MAX3241.pdf

[2] *8-bit PIC® Microcontroller (PIC18F452) datasheet*: http://ww1.microchip.com/downloads/en/DeviceDoc/39564c.pdf

[10] Display Gráfico de 128 x 64 puntos con LED de iluminación de pantalla JHD12864J hoja de especificaciones. http://www.ftdichip.com/Support/Documents/DataSheets/Modules/DS_UB232R.pdf

[3] *Microsoft Extensible Firmware Initiative FAT32 File System Specification FAT: General Overview of On-Disk Format*. Version 1.03, December 6, 2000 Microsoft Corporation.

[4] *MPLAB® C18 C Compiler Getting started*, Microchip Technology Inc. 2005

[5] *MPLAB® C18 C Compiler Libraries*, Microchip Technology Inc. 2005.

[6] *SADC10/18/20/30-Communication protocol*. Document revision 23rd March 2004 SARA di Mariotti Gabriele & C snc – Perugia

[7] *SD Specifications Part 1: Physical Layer Simplified Specification Version 2.00*, SD Group, 2006.

[8] *USB Mini-B FT232R Evaluation Module (UB232R) Datasheet*, Future Technology Devices International Ltd:

[9] *Very low drop voltage regulators with inhibit (LF33CV) datasheet*: http://www.datasheetcatalog.org/datasheet/stmicroelectronics/2574.pdf

Páginas de Internet

[1] http://es.wikipedia.org/wiki/Memory_Stick

[2] http://geovirtual.cl/EXPLORAC/TEXT/0400Asismolog.htm

[3] http://sismologia.cicese.mx/resnom/principal/FAQ.php#sismologia

[4] http://www.compactflash.org/

[5] http://www.compactflash.org/faqs/faq.php#capacities

[6] http://www.pcmcia.org/

[7] http://www.pcmcia.org/about.htm

[8] http://www.w3counter.com/globalstats.php