Reference Design

Mini-Z® USB Design Board

RD003002-0814

Overview

Zilog's Mini-Z® USB Design Board provides a reference design to incorporate USB host and peripheral functionality with Zilog's portfolio of Mini-Z® modules. Its hardware design uses a Max3421E USB transceiver to provide an interface to the USB communications with supporting power-limiting and connection circuitry. The design also incorporates a Secure Digital (SD) card using its SPI interface. The Mini-Z® USB Design Board firmware includes host functionality for USB mass storage devices and Secure Digital (SD) card support.

The Mini-Z® USB Design Board includes all of the hardware necessary to develop prototypes and projects incorporating the host and peripheral functions on Zilog's wide selection of MCUs.

The Mini-Z® USB Design Board, shown in Figure 1, is designed to be operated by either a 9V battery or an external power supply.

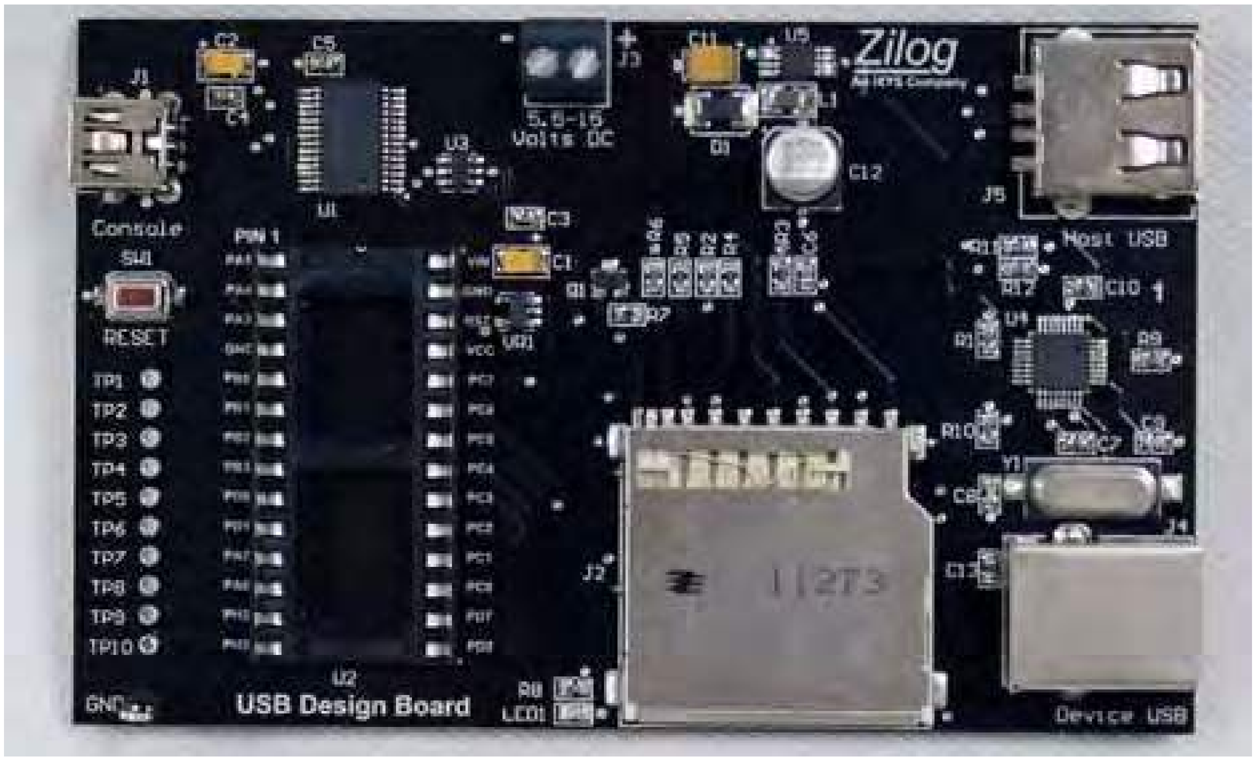

Figure 1. The Mini-Z USB Design Board

Note: The source code file associated with this application note, RD0030-SC01, is available free for download from the Zilog website. This source code has been tested with version 5.0.1 of ZDS II for ZNEO MCUs. Subsequent releases of ZDS II may require you to modify the code supplied with this application.

Mini-Z USB Design Board Features

The key features of the Mini-Z® USB Design Board are:

- Low/full speed USB host capability
- Secure Digital (SD) card
- Full-speed USB peripheral capability
- USB serial communications

Potential Applications

The Mini-Z® USB Design Board can be used to develop a number of applications; the brief list below offers a few ideas.

- Secure Digital-to-USB converter
- Adding removable storage to Zilog MCU projects
- Adding USB Human Interface Device (HID) input for Zilog MCU projects

Shell Application

This section describes how to install the USB Host Shell and become familiar with its commands.

Installing the Shell Application

Every module in Zilog's Mini-Z® product mix is preloaded with a boot loader and control shell. The shell application provides access to the module and to the design board that the module attaches to.

Observe the following brief procedure to install the shell application.

1. Connect the USB cable (A male to Mini-B male, provided in your kit) from the PC to the USB Design Board's Mini-USB port. This Mini-USB port is located adjacent to the reset switch on the opposite side of the USB-A and USB-B connectors, and is labeled CONSOLE.

2. Launch a terminal emulation program (such as HyperTerminal or TeraTerm) and establish a connection to the Mini-Z® USB Design Board. The terminal application's communication settings should reflect the following values:
   - 57600 baud rate
   - 8 data bits
   - No parity
   - 1 stop bit
   - No flow control

Note: This document assumes HyperTerminal to be the selected terminal emulation program.

3. Connect the Mini-Z® USB Design Board to the appropriate power source. When power is applied, the following command prompt should appear on your terminal:

   ```
   Z16Mini-Z>
   ```

Note: This `Z16Mini-Z>` prompt will appear if you are working with Zilog's Mini-Z® ZNEO 28-Pin Module. If you are working with Zilog's Mini-Z® WLAN Module, you'll instead see this prompt portrayed as `WLANMini-Z>`. With the Mini-Z® Z-PAN® 28-Pin Module, this prompt will appear as `BTMini-Z>`.

4. Enter the following command in the terminal console:

   ```
   Flashapp
   ```

5. Press the Enter key. The console will prompt you to transfer a hex file. From the **Transfer** menu in HyperTerminal, select **Send Text File... (no modem protocol)**.
6. The Send Text File dialog will appear in Windows. In this dialog, navigate to and select the `USBHost_app.hex` file, and click **OK** to flash the code to the Module. You may be prompted to change the **Files of Type** selection to **All files (*.*)** at the bottom of the Send Text File window.
7. After the Module has been flashed, the console will return to the command prompt. Enter the following command to execute the application and start the USB host:

   ```
   Execapp
   ```

Installation of the shell application is now complete. However, bear in mind that the next time you power on the Mini-Z® USB Design Board, you must enter the `Execapp` command to once again execute the application and be able to have the USB Mini-Z® commands available to you. For additional details about this shell application, see the Mini-Z Shell and Flash Loader Reference Manual (RM0061).

USB Host Shell Application Commands

The USB Host Shell application provides the ability to access USB mass storage devices (using FAT16/FAT32 file systems) and SD/MMC cards from the shell.

Drives are numbered (1,2, etc.) and are dynamic. Directories are referenced with a diagonal character ("/").

Table 1. Mini-Z USB Design Board Shell Command Description

Command	Description	Usage Example
List	Lists the drives that are currently accessible.	Z16MiniZ>list Drive 1: SD Card Total Size: 7757824K -FAT32 Drive 2: USB Drive Total Size: 1004031K -FAT32 Total Drives Found: 2
ChgDrive	Changes the default drive.	Z16MiniZ>chgdrive 2: Drive changed.
Dir	Lists the directories for the drive. By specifying the drive number followed by a colon, you can get a directory listing of the non default drive. You can also specify a subfolder to get a listing of the sub folder.	Z16MiniZ>dir Drive 1 Directory ./ UNDERW~1.AVI 1316579328 UNDERW~2.AVI 1268234240 UNDERW~3.AVI 1013458944 FFDSHO~1.EXE 4560237 INSTAC~1.EXE 1645248 EXTACC~1.TXT 54 Total Files 6 Total Directories 0 Optional usage: Z16MiniZ>dir 2:smrtntky Drive 2 Directory smrtntky .. < dir > DEVICE < dir > WSETTING.WFC 803 MESSAGEB.TXT 285 FCW .ICO 22486 WSETTING.TXT 628 Total Files 4 Total Directories 2
Copy	Copies a specified file. No wild cards are supported. You can copy between drives by specifying drive numbers.	Z16MiniZ>copy EXTACC~1.txt Mytxt.txt Copying...Total bytes transferred: 54 Optional Usage: Z16MiniZ>copy 2:hello.asp 1: Copying...Total bytes transferred: 2824
Del	Allows the ability to delete a file from the drive.	Z16MiniZ>del mytxt.txt Deleting file... File: mytxt.txt has been deleted.

Table 1. Mini-Z USB Design Board Shell Command Description (Continued)

Command	Description	Usage Example
CD	Allows the ability to change directories within the drive. By specifying the drive number followed by a colon, you can change the directory on the other drive. The directory location remains until either the drive is removed or the CD command is issued.	Z16MiniZ>cd 2:smrtntky Directory changed.
Type	Displays the specified text file on the console. You can press Ctrl-C to stop the display.	Z16MiniZ>type test.txt
Hello	A test text file.	Z16MiniZ>
Dump	Displays the bytes of the file in hexadecimal format. You can press Ctrl-C to stop the display.	Z16MiniZ>dump test.txt 0x48 0x65 0x6C 0x6C 0x6F 0x20 0x0D 0x0A 0x54 0x68 0x69 0x73 0x20 0x69 0x73 0x20 0x61 0x20 0x74 0x65 0x73 0x74 0x20 0x74 0x65 0x78 0x74 0x20 0x66 0x69 0x6C 0x65 0x2E 0x0D 0x0A Z16MiniZ>

Technology

This section discusses the technology of the Universal Serial Bus as well as SD card technology, its file systems, and hardware.

USB

The Universal Serial Bus (USB) is a host-controlled bus. There can only be one host per bus, but up to 127 devices. The devices are connected in a tiered-star topology. In USB 2.0 and below, all communications are host driven, including interrupt transfers that have to wait for the host to poll the device. (NOTE: The Mini-Z USB Design Board is USB 2.0 compatible, so we are not addressing USB 3.0). The communications are handled via transactions with several layers of protocols. Each USB transaction consists of a Token packet, data packet (optional) and a status packet (acknowledgement/error correction).

There is also the Start of Frame Packet that is sent by the host at regular intervals, for synchronization, which the MAX3421E takes care of, so it will not be discussed.

There are four different types of transfers that are defined in the USB protocol. They are:

Control Transfers. Used for command and status operations.

Isochronous Transfers. Transfers that are continuous at a specific rate

Interrupt Transfers. Small packets that are queried for at specific intervals

Bulk Transfers. Used for large data that is not time dependent, such as burst data.

Because a host cannot detect which device has been plugged into the USB port, the USB protocol has defined how a device must report which type of device it is. A device will therefore use descriptors which the host can retrieve through a control transfer mechanism. These descriptors are diagrammed in Figure 2.

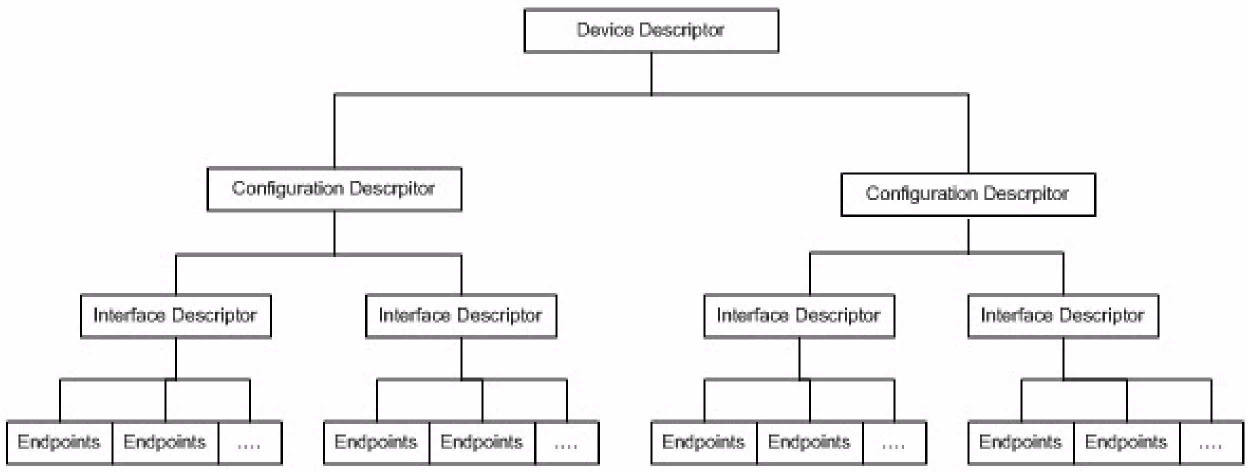

Figure 2. USB Descriptors

Each device can have only one Device descriptor, and each Device descriptor includes the following device information:

- USB Version supported
- Device Class (if 0, then located in the interface Descriptors)
- Device SubClass
- Device Protocol
- Max Packet size
- Vendor ID
- Product ID
- Device Release Number
- Manufacture String Descriptor Index
- Product String Descriptor Index
- Serial Number String Descriptor Index
- Number of Configurations

Each device descriptor can have one or more Configuration descriptors, which define the power usage and the number of interfaces for a particular configuration. The host must activate the configuration that it will use, and is only allowed one active configuration at a time. Configuration descriptors include the following information:

- Number of interfaces
- Configuration string descriptor index
- Power attributes:
  - Self-powered
  - Bus-powered
  - Remote wakeup
- Maximum power (in 2mA units)

Each configuration descriptor can have one or more Interface descriptor, which is a grouping of endpoints that provides a feature for the device. The host may enable as many of these interfaces as is appropriate. An Interface descriptor includes the following information:

- Number of endpoints
- Interface class code (USB class codes)
- Interface subclass code (USB subclass codes)
- Interface protocol (USB protocol code)
- Interface string descriptor index

Each interface descriptor can have one or more Endpoints descriptors, which are used to specify the type of transfer, direction, and polling interval for the host to use. There is a special endpoint, Endpoint 0, that has no descriptor; this particular endpoint is the default control endpoint and is always active. Endpoints descriptors include the following information:

- Endpoint address (includes a direction bit)
- Endpoint attributes (includes transfer type, synchronization info, endpoint type)
- Maximum packet size
- Polling interval (in frame counts)

Not shown in Figure 2 are String descriptors, which can include strings such as manufacturer, product name, etc., and can be accessed by an index of all descriptors. These String descriptors allow for different spoken languages.

For a host to acquire device information and configure a device for operation, communication starts with a set-up packet on Endpoint 0. This set-up packet includes information such as the type, direction, *recipient* (device, interface, endpoint), request, and associated data necessary for the request. There are seven standard device requests:

- Get_Status
- Clear_Feature
- Set_Feature
- Set_Address
- Get_Descriptor
- Set_Descriptor
- Get_Configuration
- Set_Configuration

When a Get_Descriptor request is issued for the Configuration descriptor, the Interface and Endpoints descriptors are included in the response. The typical sequence for acquiring information is:

1. Retrieve a Device descriptor to determine the number of Configuration descriptors.
2. Set a unique bus address for the device.
3. For each configuration:
   a. Retrieve Configuration descriptors.
   b. Parse Interface and Endpoints descriptors from the reply.

After the descriptors have been processed, the host knows what the device is and what its function is. If the device is something the host can support, the host will select and activate an appropriate configuration and start interfacing with the endpoints.

Device communication is based on logical channels or pipes. These pipes are logical connections between a host and its endpoints; they can be opened or closed. Endpoints are part of the device, so they always exist. A device can have up to 32 endpoints. There are two types of endpoints: *stream* and *message (control)*. Message endpoints are bidirectional and are used for control transfers. The *stream* endpoint is a unidirectional pipe.

The Mini-Z USB Design Board's firmware currently supports only USB mass storage devices. These types of storage devices use bulk transfer endpoints to transfer data in and out via the SCSI protocol. The SCSI protocol can support up to seven *Luns* (different disks) per device, though typically there is only one Lun on a USB device.

To transfer data on a mass storage device, the mass storage unit must also be initialized. This initialization includes identifying how many Luns are available on the device, determining the capacity of the drive, and waiting for it to be ready.

All communication with the SCSI includes a header packet of 31 bytes (i.e., a Command Block Wrapper, or CBW) followed by the data, and completed with a status packet of 13 bytes (i.e., a Command Status Wrapper, or CSW). All of these packets are sent/received on the bulk transport of the associated endpoints. There only a limited set of SCSI commands that are required to be supported for USB mass storage devices. The typical commands used for mass storage devices are:

GetReadyStatus. Identify when the SCSI subsystem of the device is ready

GetCapacity. Returns the capacity of the drive

Read10. Bulk Read Block command

Write10. Bulk Write Block command

There are two USB application-specific commands that are supported and sent through the control pipe:

RequestLUN. Request the Lun count (although some devices do not support)

MSBulkReset. Resets the SCSI subsystem on the Bulk endpoint

SD Card

A Secure Digital (SD) card is a nonvolatile memory card developed by Toshiba, Matsushita Electric and SanDisk from the MultiMedia Card specifications. An SD card offers features such as digital rights management (DRM), higher density, and faster speeds. The SD Card Association maintains the Secure Digital Standard.

There are currently three types of SD cards:

SDSC. Standard, a maximum of 2GB capacity

SDHC. High Capacity, a maximum of 32GB

SDXC. eXtended Capacity, a maximum of 2TB

The Mini-Z USB Design Board only supports SDSC and SDHC cards, as well as MMC cards.

An SD card can operate in two modes: SD Mode and SPI Mode. The Mini-Z USB Design Board employs the 4-wire SPI Mode. The host communicates with the SD card with commands to identify and control the cards. The commands include information commands, configuration commands and reading/writing commands. SD cards are block-addressable storage devices that require read and writes to be in full blocks (typically 512 bytes). The Mini-Z USB Design Board communicates with the SD cards in SPI Mode.

To access an SD card, an initialization sequence must be followed to both determine the type of card that it is (SDSC, SDHC or MMC), then set the card to be in a ready state to receive commands. After this initialization sequence is completed, the host can read and write blocks.

File Systems

The file systems supported with the Mini-Z USB Design Board firmware is the FAT16 and FAT32. These file systems contain the Master Boot Record, which contains the partition information, and a Volume Boot Record, which contains the information for the volume. Some devices may only have a Volume Boot Record on the first sector (sector 0) instead of a Master Boot Record.

The Volume Boot Record contains all of the information for accessing the partition (volume), including the size of the sector, the number of sectors in a cluster, the maximum number of clusters, the location for the File Allocation Tables, the type of FAT being used, etc.

The FAT file system breaks up a space into clusters, and clusters are further broken into sectors. A sector will typically be 512 bytes (although there are new standards that increase sector size to handle more space). The File Allocation Table keeps track of all of the clusters that are being used and the chain of clusters for files that exceed a single cluster.

There is a specific area in a FAT file system that stores directory information. On a FAT16 file system, the Directory sectors occur right before the start of the clusters; on a FAT32 system, the Directory entries are allocated clusters. Each Directory entry is 32 bytes in length; an entry contains information about a file such as its attributes, starting cluster, size, 8.3 filename, etc. Additionally, long file name (LFN) directory entries allow long file names. These are additional Directory entries that contain double-byte characters describing a name as well as the original directory entry with an 8.3 filename. LFNs retain a specific attribute that serves to identify the type of entry it is.

The Mini-Z USB Design Board firmware only supports DOS names, not Long File Names. To allow for long file name entries, all entries that contain the LFN attribute (`0x0F`) are skipped.

Hardware

The Mini-Z USB Design Board hardware design uses a Max3421E USB transceiver to provide an interface to USB communications. The MAX3421E contains the digital logic and analog circuitry necessary to implement a full-speed USB peripheral or a full-/low-speed host compliant to USB Specifications Revision 2.0. An internal serial interface engine handles the low-level USB protocol details such as error checking, bus retries, and frame packets. The interface to the MAX3421E is the SPI interface.

This hardware includes both circuitry for peripheral implementation, as well as the host (though the system can only be in either host or peripheral mode at any given time). The host side also incorporates a current limit switch (MAX1776) to limit current draw from the Host port, thereby allowing power to be supplied to any USB devices without drawing excess power.

The SD card circuitry includes an analog circuit to only apply power to the SD card when there is a card detected. This circuit is also monitored by the MCU to determine when a user adds or removes an SD card.

Electrical Specifications

Stresses greater than those listed in Table 2 may cause permanent damage to the device. These ratings are stress ratings only. Operation of the device at any condition outside those indicated in the operational sections of these specifications is not implied. Exposure to absolute maximum rating conditions for extended periods affects device reliability. For improved reliability, unused inputs must be tied to one of the supply voltages (V_{DD} or V_{SS}).

Table 2 describes the electrical characteristics of the Mini-Z USB Design Board and reflects all available data as a result of testing prior to qualification and characterization. As such, the data presented in Table 2 is subject to change.

Table 2. Electrical Specifications for the Mini-Z USB Design Board

Parameter	Min	Max	Units	Notes
V_{IN} range	5.5	17	Volts	Module-dependent.
Max voltage range, all other pins	–0.3	5.5	Volts	I/O pins and Reset.
Max current for I/O pin connection points	–25	25	Milliamps (mA)	
Max V_{IN} current		1	Amps	
V_{OUT} USB Host	4.8	5.2	Volts	
Ambient temperature	–40	105	Degrees Celsius	
Storage temperature	–65	150	Degrees Celsius	

Table 3 describes the electrical characteristics of the Mini-Z USB Design Board's terminal blocks.

Table 3. Terminal Block Specifications

Rated Voltage	150 Volts
Rated Current	6 Amps
Wire Size	AWG 28-16
Test Voltage	2.0K volts
Rated Torque/Screw Size	0.4 Nm/M2
Wire:	
Max rated cross section:	
Solid Wire	1.5mm

Table 3. Terminal Block Specifications

Stranded Wire	1.5mm
Stranded wire with Ferrules	.75mm
Stripped Length	5.5mm

Packaging

Figure 3 displays an assembly diagram of the Mini-Z USB Design Board.

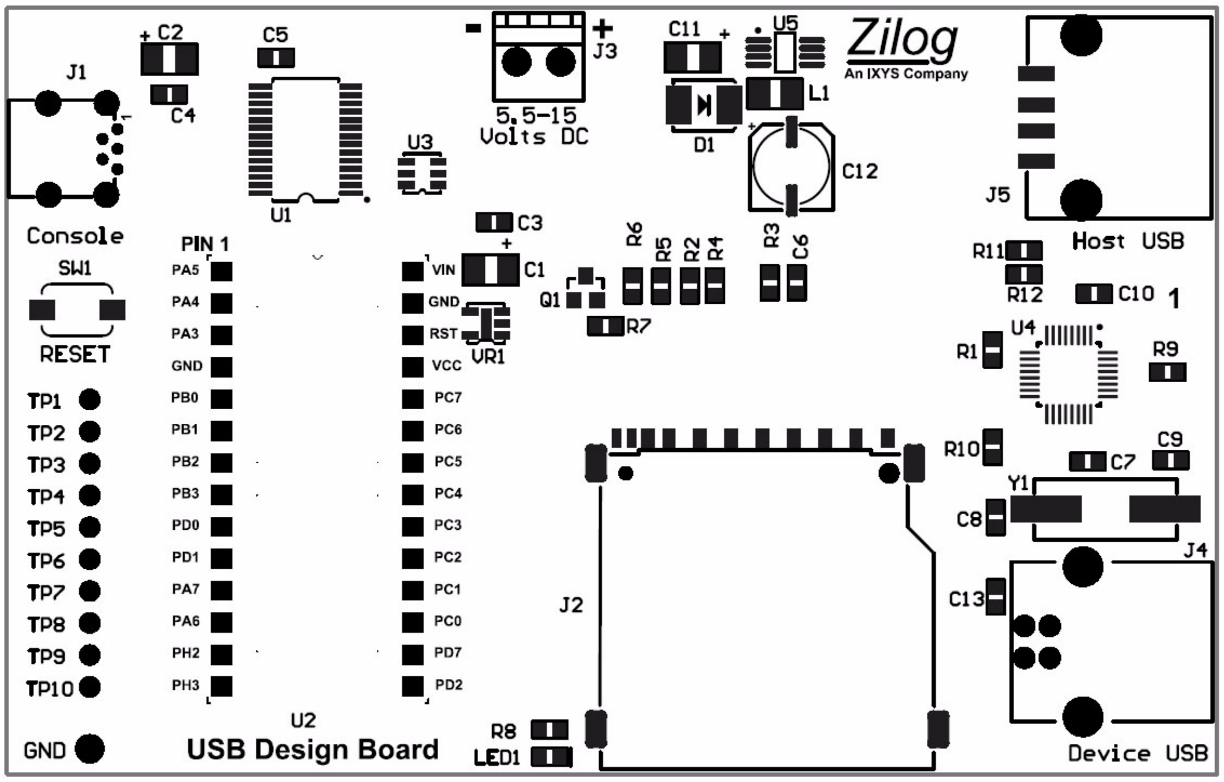

Figure 3. Mini-Z USB Design Board Assembly Diagram

Bill of Materials

Figure 4 lists all parts that comprise the Mini-Z USB Design Board.

ITEM #	Designator	Value	Footprint	Description	Manufacturer	Mfg. Part Number
1	C1	10uF 6.3V	CAP 1206	CAP TANT	AVX	TAJA106K006RN
2	C2	4.7uF	CAP 1206	CAP 4.7uF 16V A	AVX	
3	C3, C4, C5, C6	.1uF	0603	CAP CER 25V 0603	Murata	GRM188F51E104ZA01D
4	C7, C10, C13	1uF	0603	CAP 1uF 16V 0603	Murata	GRM188F51C105ZA01D
5	C8, C9	18pF	0603	CAP CER 18pF 50V 0603	Murata	GRM1885C1H180JA01D
6	C11	10uF	CAP 1206	TANT 16V	ROHM	TCA1C106M8R
7	C12	100uF 10V	CAP ELECT F55 7.2x6.6mm	CAP 100uF 10V 6.3mmDia SMD	United ChemiCon	EMVA100ADA101MF55G
8	D1	SS3P3-M3	DO-214AA_SMB	Schottsky Diode (3A/30V)	Vishay	SS3P3-M3/84A
9	J1	Mini B	USB MiniB TH	USB MiniB 5 ThroughHoe	WURTH ELECT	651005136521
10	J2	SD Card	SD CARD TE 2041021	SD Card holder	3M	
11	J3	POWER	TermBlock2 3.5mm	TERM BLOCK 125V 6A	On Shore Tech	ED555/2DS
12	J4	USB B 4PIN	USB_B	USB B 4 PIN		
13	J5	USB Type A	USB Type A	USB Type A Connector	Adam Tech	
14	L1	10uH	1206	100 Ohm 3A		
15	LED1	GRN	LED0603	LED GRN 2.03V 10mA 0603	Panasonic	LNJ314G83RA
16	Q1	DMP2215L	SOT23 DMP2215L	FET PCH 2V 2.7A SOT23	Diodes Inc	DMP2215L-7
17	R1, R2, R6, R9, R10	100K	0603	RES 100K 0603		
18	R3, R4, R5	56K	0603	RES 56K 1% 1/10W		
19	R7	1K	0603	RES 1K 0603	Vishay	CRCW06031K00FKEA
20	R8	2K	0603	RES 2.0K 1%		
21	R11, R12	33	0603	RES 33 Ohm 0603	Yageo	RC0603FR-0733RL
22	SW1	RESET	SW SMD 6.8x3.7MM	SW PB SMD 6.1x3.7mm	C&K	PTS635SK25SMTR LFS
23	GND	TP	TEST POINT 063	Test Point	Keystone Elect	5011
24	U1	FT232RL	SSOP28	IC USB TO SERIAL UART SSOP28	FTDI	FT232RL-REEL
25	U2 --Dip Socket--	ZNEO STAMP	DIP28x.6	ZNEO STAMP28 SOCKET		
26	U3	SN74LVC2GU04	SOT23-6	Inverter Dual Sot23-6	TI	SN74LVC2GU04DBVR
27	U4	MAX3421E	TQFP32L	USB Transciever	Maxim	
28	U5	MAX1776	MAXu8	Buck Converter with Current Limit	Maxim	
29	VR1	REG102NA-3.3	SOT23-5L	VOLT REG 3.3V 400mA SOT23-5	TI	REG102NA-3.3/250
30	Y1	12Mhz Crystal	HCM49		ABRACON	

Figure 4. Mini-Z USB Design Board Parts List

Mechanical Profile

Figure 5 displays the dimensions of the Mini-Z USB Design Board.

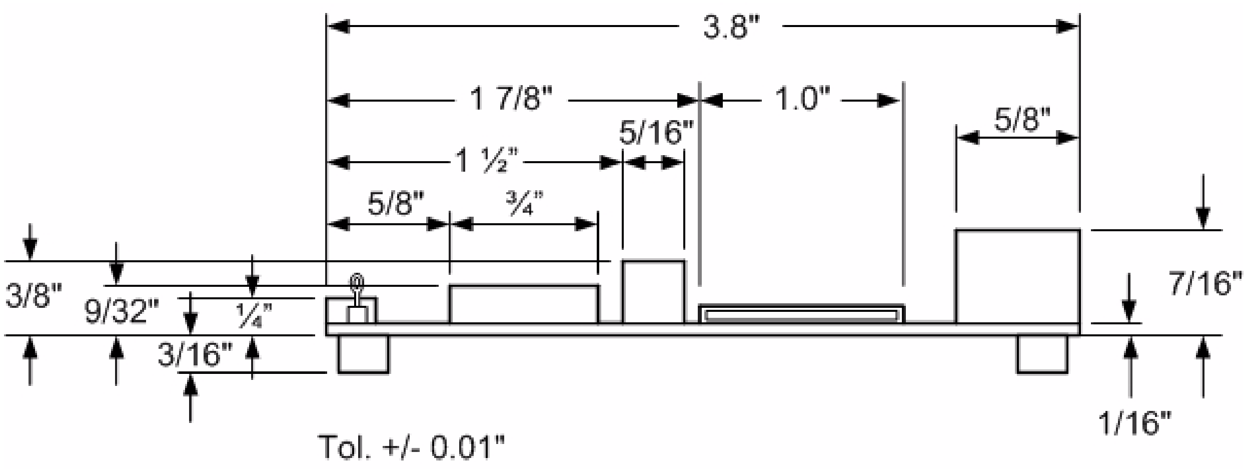

Figure 5. Mini-Z USB Design Board Dimensions

Kit Contents

The Mini-Z USB Design Board Kit contains the following items:

- Mini-Z ZNEO 28-Pin Module
- Mini-Z USB Design Board
- USB SmartCable
- Mini-Z to standard debug adapter
- USB cable (A male to Mini-B male)
- DIP Package Extractor

Ordering Information

The products associated with the Mini-Z USB Design Board are available individually or as a kit and can be purchased from the Zilog Store – simply click the Store Product IDs listed in Table 4.

Table 4. Mini-Z USB Design Board Ordering Information

Part Number	Description	Store Product ID
Z16F28UB100ZRDG	Mini-Z USB Design Board	RD10024
Z16F28UB100KITG	Mini-Z USB Design Board Kit	RD10025
Z16F2800100MODG	Mini-Z ZNEO 28-Pin Module	RD10002
Z16F28WF100MODG	Mini-Z WLAN 28-Pin Module	RD10003
Z16F28ZP100MODG	Mini-Z Z-PAN 28-Pin Module	RD10008

Related Documentation

The documents associated with the Mini-Z USB Design Board are listed in Table 5, and supporting manufacturers' documentation is listed in Table 6. Each of these documents can be obtained from the Zilog website (except where noted in Table 6) by clicking the link associated with its Document Number.

Table 5. Mini-Z USB Design Board Documentation

Document Number	Description
RD0030	This Mini-Z USB Design Board Reference Design document.
RD0030-SC01	Source code for the Mini-Z USB Design Board Reference Design.
RD0006	Mini-Z ZNEO 28-Pin Module Reference Design document.
RD0006-SC01	Mini-Z Library.
RM0061	Mini-Z Shell and Flash Loader Reference Manual.
PS0220	ZNEO Z16F Series Product Specification.
UM0188	ZNEO CPU Core User Manual.
UM0181	USB Smart Cable User Manual.

Table 6. Supporting Manufacturers' Documentation

Document Number	Description
MAX3124E	MAX3421E Datasheet.
MAX1776	MAX1776 Current Limit Step-Down Converter.
SD Card Spec	SD Specifications Part 1 Physical Layer Simplified Specification.

Table 6. Supporting Manufacturers' Documentation (Continued)

Document Number	Description
USB 2.0 Spec	USB 2.0 Specifications.
USBmasbulk	USB Mass Storage Class Bulk-Only Transport Specification.

Appendix A. Schematic Diagram

Figure 6 shows a schematic diagram of the Mini-Z USB Design Board.

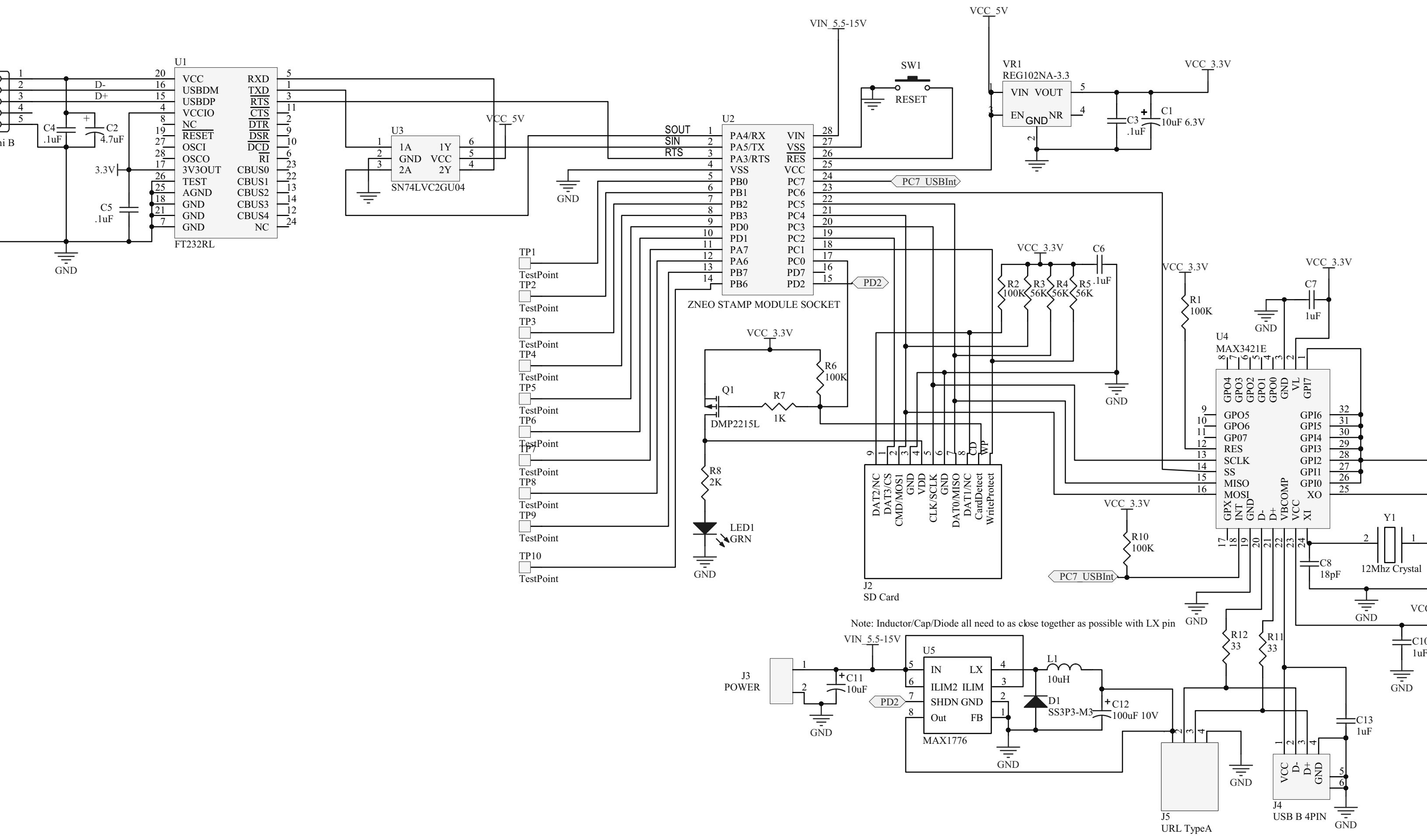

Figure 6. Mini-Z USB Design Board Schematics

Appendix B. Firmware Layout

The firmware consists of a layered approach to isolating the hardware from its interfaces. This approach, as described in Tables 7 and 8, allows a developer to add/remove hardware and to add/remove volume support.

Table 7. Upper Layer Firmware Programs

Program	Description
MAIN.C	This file contains the application initialization and command handling routines. The InitApplication function is the application entry code for the Mini-Z console to call when the user executes the application. This function initializes specific required ports and initializes the hardware buses to be used. The command handling function is next registered with the console; additionally, the callback function (Monitor) is called every 300 ms. Ports used by the application that are not available for other commands are then added to the exclusion list; the routine then exits, returning control to the console. Every 300 ms, the Monitor function is called by the console. This function checks to see if an SD card or USB device has been added or removed. If a device has been added, the device is initialized; if a device has been removed, the device is removed from the list. When a command is entered, the console passes control to the command-handling routine that is registered to perform the command.
USBCMDS.C	This routine contains the implementation for the different commands that are added to the console to access the Mini-Z USB Design Board.
FILEIO.C	This file provides the implementation for standard file I/O functions such as fopen, fclose, fread, fwrite, etc.
DIRIO.C	This file provides the implementation for Directory access for drives.
FAT.C	This file provides the implementation for FAT16 and FAT32 file allocation tables.
DRIVE.C	This file contains the implementation to segregate volumes into logical drives.

Table 8. Lower Layer Firmware Programs

Program	Description
DEVICE.C	This file provides an interface to the upper layers for reading and writing devices, regardless of the hardware. As a result, the upper layers can perform requests for reads and writes without having to recognize the hardware.
USBMASSSTORAGE.C	This file contains the mass storage handing routine for the USB devices. USB mass storage devices use a SCSI interface to retrieve data.
USBHOST.C	This file contains the USB host functionality for the MAX3721E device with USB protocols. Control and endpoint communications are provided for the USB device as well as to monitor the bus for the insertion and removal of USB devices.

Table 8. Lower Layer Firmware Programs (Continued)

Program	Description
SD.C	This file contains the implementation of the interface routine to the SD card; it includes the initialization, reading and writing routines for the SD card.
SPI.C	This file contains the interface to the SPI bus on the ZNEO microcontroller. The SPI bus is shared between the USB transceiver hardware and the SD card. These devices are transaction-oriented devices, and will track if the SPI bus is in use by the other device, thereby completing the transaction on one device before starting a transaction on the other device. The low-level access routine to the SD card and USB transceiver can also be found in this file.

Appendix C. Flow Charts

This appendix displays flow charts that diagram the major functions of the Mini-Z USB Design Board. Figure 7 illustrates the flow of the application initialization routine.

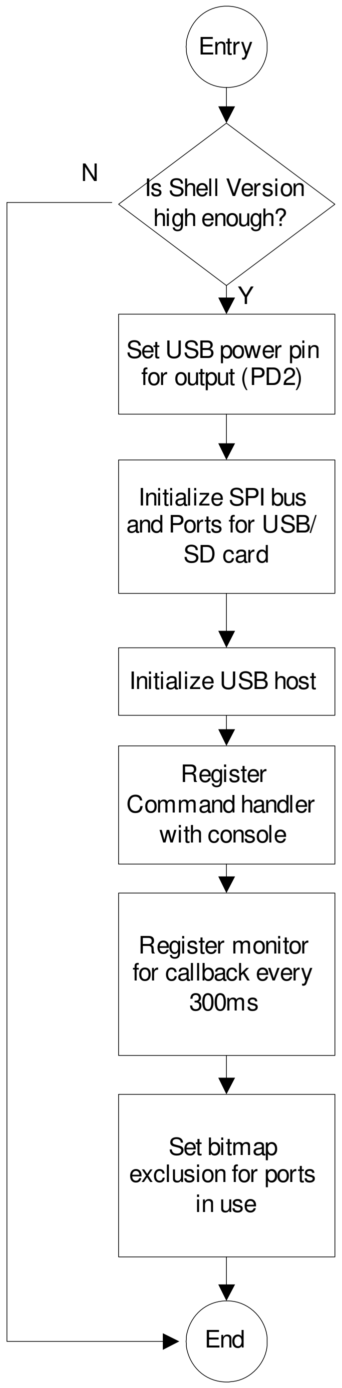
Entry
N
Is Shell Version high enough?
Y
Set USB power pin for output (PD2)
Initialize SPI bus and Ports for USB/ SD card
Initialize USB host
Register Command handler with console
Register monitor for callback every 300ms
Set bitmap exclusion for ports in use
End

Figure 7. Application Initialization Routine

Figure 8 illustrates the flow of the monitoring routine.

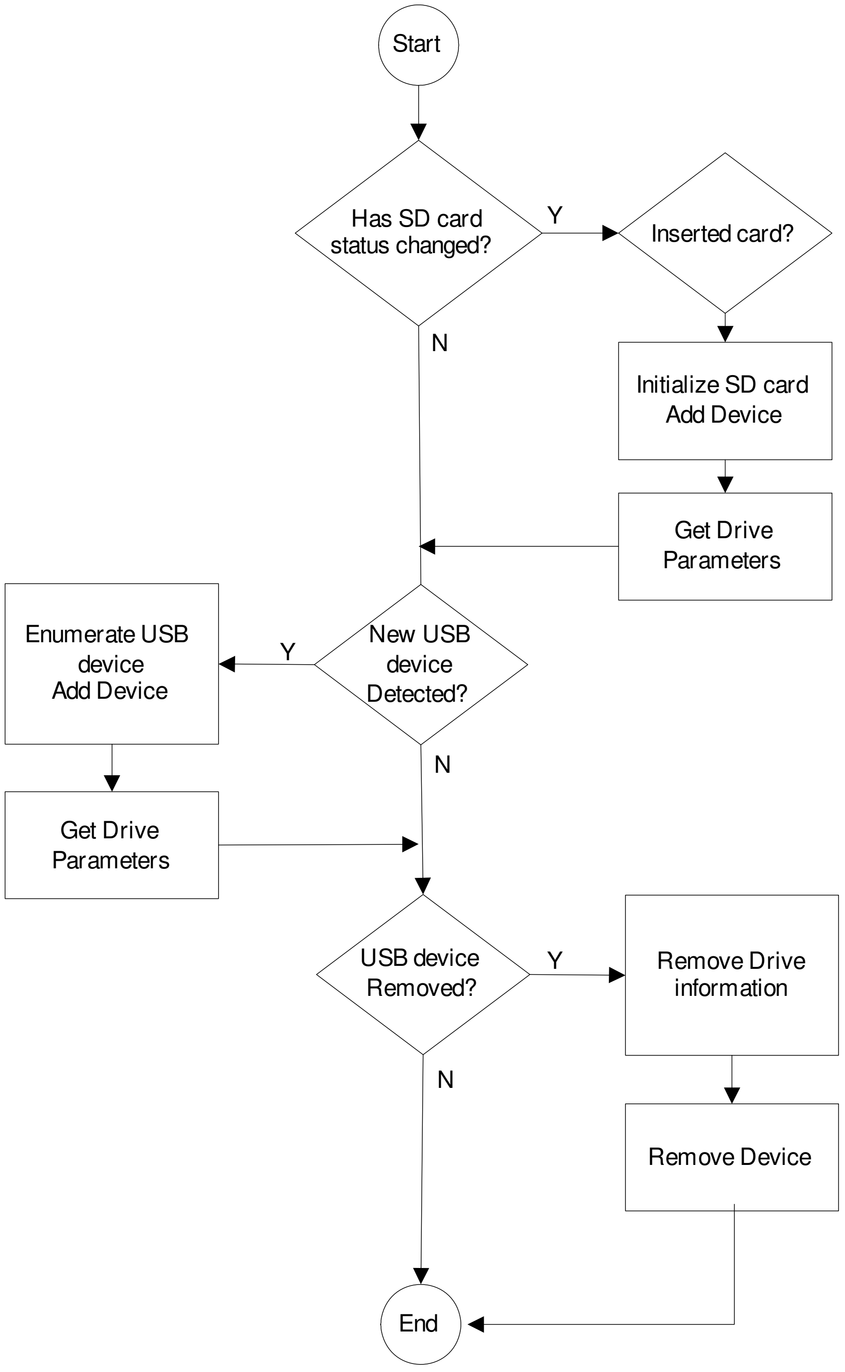

Figure 8. Monitoring Routine

Figure 9 illustrates the flow of the SD Card initialization routine.

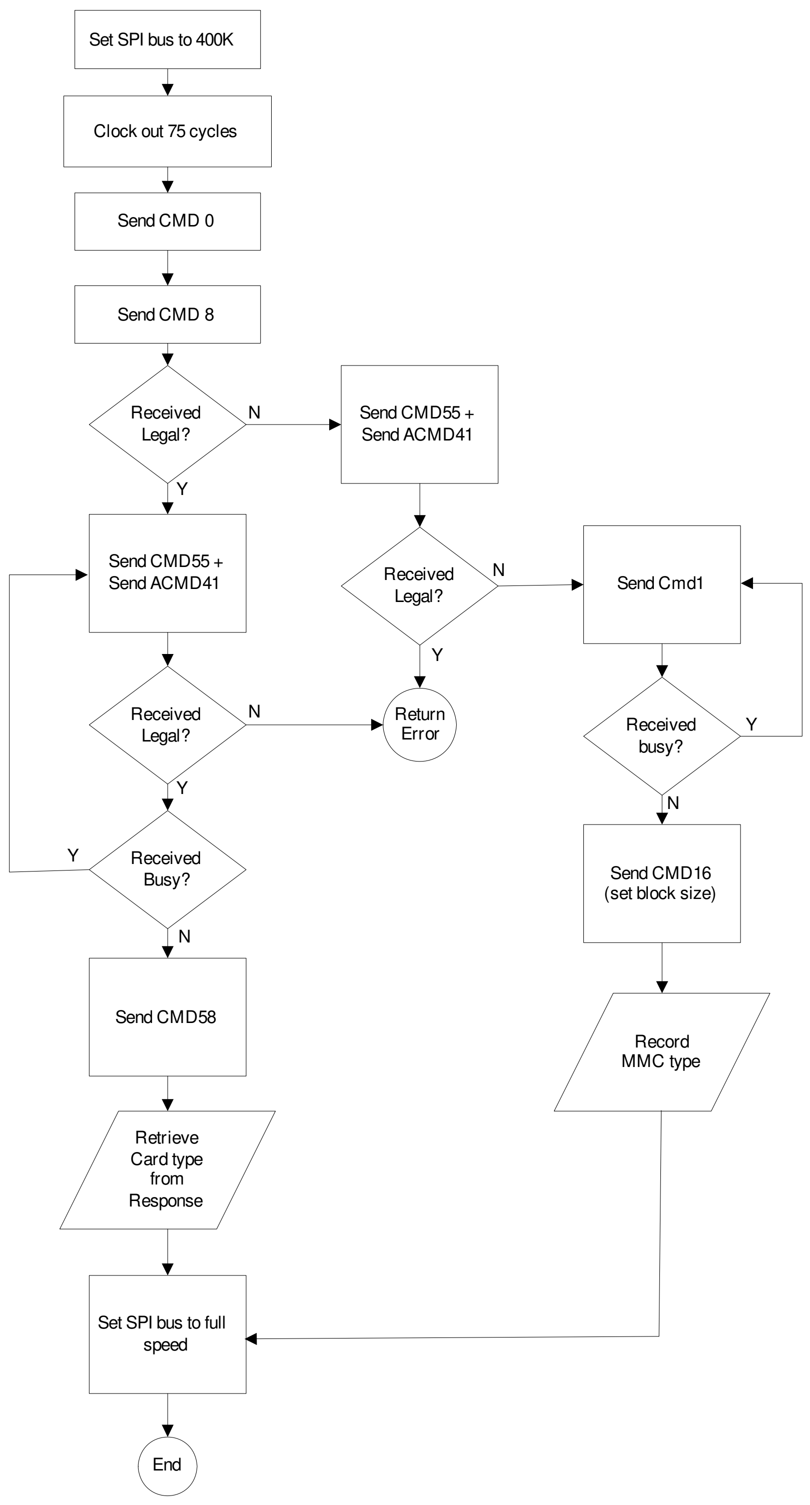

Figure 9. SD Card Initialization Routine

Figure 10 illustrates the flow of the USB device enumeration routine.

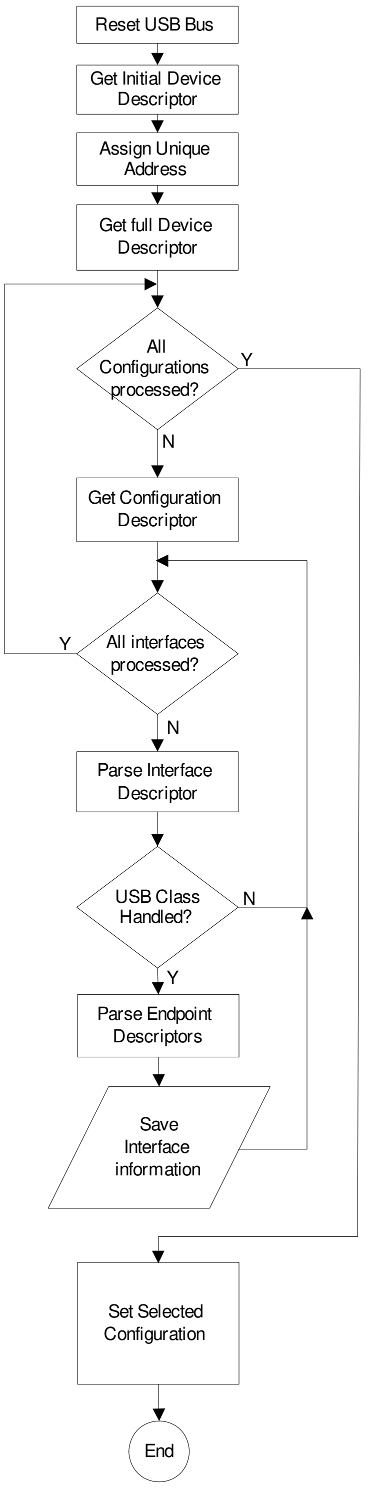
Reset USB Bus
Get Initial Device Descriptor
Assign Unique Address
Get full Device Descriptor
All Configurations processed?
Y
N
Get Configuration Descriptor
All interfaces processed?
Y
N
Parse Interface Descriptor
USB Class Handled?
N
Y
Parse Endpoint Descriptors
Save Interface information
Set Selected Configuration
End

Figure 10. USB Device Enumeration Routine

Figure 11 illustrates the flow of the drive information routine.

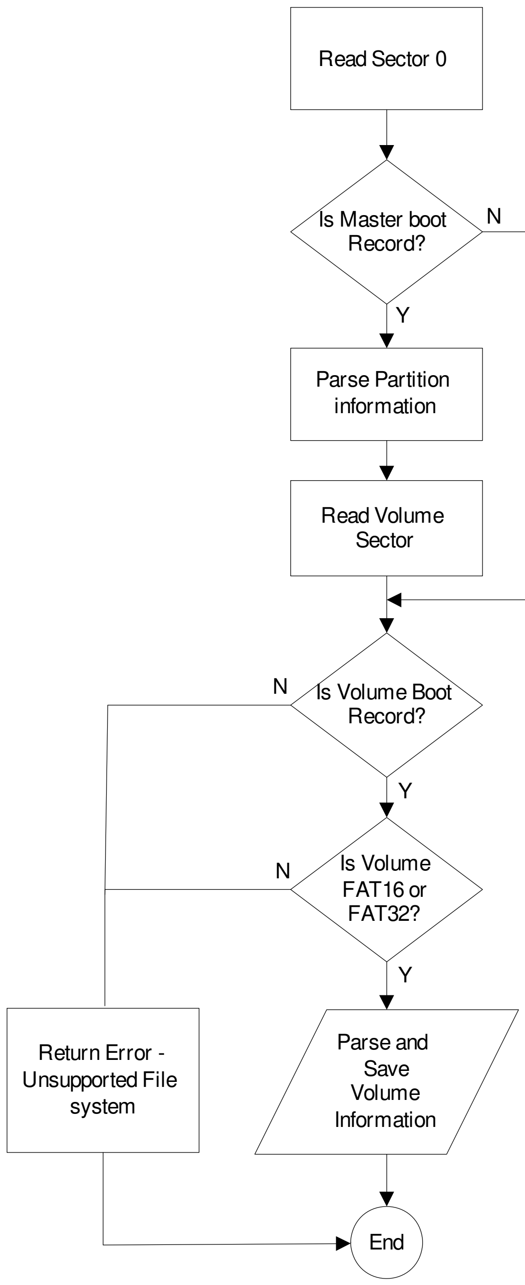
Read Sector 0
Is Master boot Record?
N
Y
Parse Partition information
Read Volume Sector
N
Is Volume Boot Record?
Y
N
Is Volume FAT16 or FAT32?
Y
Return Error - Unsupported File system
Parse and Save Volume Information
End

Figure 11. Drive Information Routine

Customer Support

To learn more about this product, find additional documentation, get your technical questions answered or report issues, please contact esales@zilog.com.

Warning: DO NOT USE THIS PRODUCT IN LIFE SUPPORT SYSTEMS.