Corsham Technologies, LLC

www.corshamtech.com

617 Stokes Road, Suite 4-299

Medford, NJ 08055

SD Card System

Last update: 2017

Table of Contents

Introduction

When we first developed our SS-50 6800 CPU board and motherboard, it was quickly evident that some sort of mass storage would soon be required. Originally we prototyped a conventional disk drive controller, but mass storage technology has advanced quite a bit since the mid 1970s so it seemed fitting to adopt modern technology to these older computers.

Looking around at available technologies, I remembered having an SD card shield for an Arduino, so work began using that. The code quickly outgrew the limited RAM on the ATMega128 processor, so the step up to the Arduino Mega seemed the next logical step.

Features

- Drivers provided for complete FLEX support.
- FLEX can be booted from the SD card.
- Supports mounting DSK format files as FLEX disk drives.
- Has a real-time clock on-board for providing time and date data.
- Open source based on the Arduino Mega; all source code is provided.
- User expandable; new devices can be added.
- Has a real-time clock on-board for providing time and date data.

Installation

Parallel Card

The software is written assuming the parallel card is in slot 6. There are no jumpers that need to be set on this board, so simply plug in the keyed ribbon cable, then plug the board into slot 6.

SD Shield

If you purchased the entire Arduino and SD shield from us, then all the software is loaded and ready to go. Plug the other end of the ribbon cable into the shield. Then install the micro SD card.

If you're using a recent version of our 6800 board then the version of xSWTBUG should be 1.2 or greater. Look at the label on the EPROM on the upper right corner of the board to see what version you have. If you've got a version prior to 1.2 then you should have gotten a new EPROM with the SD card that needs to be installed.

Now turn on your computer. The green LED on the SD Shield should turn on solid for several seconds, then begin turning on/off about once a second, indicating the SD software is running.

At the xSWTBUG monitor, press X and it should respond with a "$$" prompt. At this prompt, press "B" and FLEX should boot (everything we typed in is blue):

```
Extended SWTBUG v1.2 by Corsham Technologies
www.corshamtech.com

$$ ?
X = Return to SWTBUG
B = Boot from SD card
P = Ping SD controller
D = Do directory of SD card
T = Type file from SD card
L = Load SREC file from SD card
N = Number guessing game
M = Memory tester
O = Othello

$$ Ping... success!

$$ B

Booting from SD...
Loading FLEX.........................

FLEX 2.0

DATE (MM,DD,YY)? 1,1,1   ← This value is fixed later

20:03:19, Saturday, August 22, 2015
+++
```

How It Works

The SD/RTC Shield that plugs into the Arduino Mega has a socket for a Micro SD card, as well as a hardware real-time clock (RTC).

The SD card uses files that can be read and written on a personal computer, possibly with the use of an adaptor card. Besides regular files, there are also special files that are images of FLEX disks, and almost always have an extension of DSK. The software on the Arduino makes these files appear as disk drives to FLEX, allowing

FLEX to operate using with them. You "insert" a disk by mounting a DSK format file to a drive from 0 to 3, and then FLEX can access those files.

When you are done using a file, you unmount it, which is the equivalent to removing the disk from the disk drive.

We have included a micro SD card with two files on it:

- SD-BOOT.DSK
- CT-UTILS.DSK

The first disk is usually mounted as FLEX's drive 0. It contains a bootable image of FLEX, many standard FLEX utilities and Corsham Tech written utilities that make it easier to work with the new capabilities of the SD card. These new commands are documented in the 6809 Software

xSBUG (shipped with all our 6809 boards) has a command to boot Flex but no other SD card related commands.

D – Boot from SD

Pressing D will start the boot sequence.

New FLEX Commands section.

The second disk contains source code to the utilities we've provided. They show how to write utilities that make use of the SD card's capabilities, so feel free to start with them if you want to write your own utilities, add functionality, or fix bugs.

SD.CFG

One other file you might want to explore and modify is SD.cfg. It tells the Arduino software which DSK files to mount on which drives at start-up. It is a text file that can be edited with any text editor, such as notepad (Windows), textedit (OS X), vi/emacs (Linux). Do not use a word processor, as they usually include formatting information which will be misinterpreted by the Arduino code.

The software on the Arduino is not sophisticated when it comes to parsing and interpreting the contents of the SD.cfg file. The format must be followed exactly or things may not work as expected.

```
# Sample SD.cfg file
#
# Any line starting with a "#" is a comment.
#
# Valid lines start with a digit from 0 to 3
# indicating the drive number.
#
# Next must be either an 'R' for read-only,
# or a colon.  If the R is present, then it
# must be followed by a colon.  If the R is
# not present then the drive is read-write.
#
# After the colon is the name of the file to
# mount.  It must be the full filename,
# including any extension.
#
0:SD00BOOT.DSK
1:CT_UTILS.DSK
2R:DANGER.DSK
3:PLAY.DSK
```

By default, SD00BOOT is installed as drive 0 and CT_UTILS is installed as drive 1. Depending on what we feel like on any particular day, additional disks might be installed.

LEDs

The shield has three LEDs that are under software control. This is the current meaning of each:

GREEN – turns on/off every second to indicate the main software loop is running.

AMBER – Turns on at the start of a command from the host, and turns off when the response has been sent. This is like the "activity light" on most disk drives. If you are modifying code or writing your own low-level functions, if the amber LED is on for long periods of time, it can indicate some software problems.

RED – Indicates a problem. Currently, this comes on when the SD card is removed and turns off when an SD is re-inserted.

6800 Software

This project started to support our 6800 CPU board, so this is where most of the development was done and most of the utilities were written. Our 6800 CPU Board with xSWTBUG version 1.2 and higher have built-in commands and functions for working with the SD card.

Monitor Commands

xSWTBUG v1.2 and higher have commands to directly work with the SD card. Currently, the default slot for the parallel card must be slot 6. If this interferes with other boards on your system the slot can be changed with a change of one constant, rebuilding the monitoring and burning a new EPROM. If you need this done, please contact us for information.

B – Boot Operating System

This command reads track 0, sector 0, disk 0, into memory at address $A700 and jumps to it. For most disks this will load the bootloader that knows how to read in the rest of the operating system and transfer control to it once fully loaded.

D – Do a directory of the SD filesystem

This will display a list of all files at the top level (root directory) of the SD card. Filenames are all converted to upper case and forced to the 8.3 format. There is no way to look inside other directories.

T – Type a file

This is a way to see the contents of a file. It should only be used with text files.

L – Load an SREC file

This was a quick way to use the SD card before we had FLEX booting. This will prompt you for the name of an SREC format file (often with the extension S19) and load it into memory.

6809 Software

xSBUG (shipped with all our 6809 boards) has a command to boot Flex but no other SD card related commands.

D – Boot from SD

Pressing D will start the boot sequence.

New FLEX Commands

All of our SD related software runs on both 6800 and 6809 based machines, so each of these commands should work for both.

DATE

This is a standard utility but the one we include has been modified to get the current date and time from the SD card's real time clock (RTC). If you use the command line arguments to set the date, it will not affect the hardware real time clock. Use the SETCLOCK command to set the clock instead.

Our version of date also displays more than the original, including date, time, and day of the week.

```
+++DATE
20:03:47, Saturday, August 22, 2015
+++
```

DATE Bugs/"Features"

- You can't set the hardware clock using this command. Use SETCLOCK instead.
- If you specify a date on the command line, it is ignored.

MOUNT

This is a new command we've added to help manage mounted drives. With this utility, you can see what files are mounted to which drives, unmounts DSK images, and mount DSK images.

```
+++MOUNT

MOUNT version 1

Options:
   D = Directory of DSK files
   L = List mounted drives
   M = Mount drive
```

```
   U = Unmount drive
   X = Exit back to FLEX

Enter your command: List

  0 = SD00BOOT.DSK
  1 = CT_UTILS.DSK
  2 = BLANK.DSK
  3 = 6800BAS.DSK

Options:
   D = Directory of DSK files
   L = List mounted drives
   M = Mount drive
   U = Unmount drive
   X = Exit back to FLEX

Enter your command: U Unmount

Enter drive number to unmount (0-3) or Enter to cancel: 2

Options:
   D = Directory of DSK files
   L = List mounted drives
   M = Mount drive
   U = Unmount drive
   X = Exit back to FLEX

Enter your command: List

  0 = SD00BOOT.DSK
  1 = CT_UTILS.DSK
  2 =
  3 = 6800BAS.DSK

Options:
   D = Directory of DSK files
   L = List mounted drives
   M = Mount drive
   U = Unmount drive
   X = Exit back to FLEX

Enter your command: Mount

Enter drive number to mount (0-3) or Enter to cancel: 2
Enter filename to mount: BLANK.DSK

Options:
   D = Directory of DSK files
   L = List mounted drives
   M = Mount drive
   U = Unmount drive
```

```
   X = Exit back to FLEX

Enter your command: List
  0 = SD00BOOT.DSK
  1 = CT_UTILS.DSK
  2 = BLANK.DSK
  3 = 6800BAS.DSK

Options:
   D = Directory of DSK files
   L = List mounted drives
   M = Mount drive
   U = Unmount drive
   X = Exit back to FLEX

Enter your command: Directory
  SD00BOOT.DSK
  WOMBAT.DSK
  WHATEVER.DSK
  CT_UTILS.DSK
  BLANK.DSK
  TOAD.DSK
  6800BAS.DSK

Options:
   D = Directory of DSK files
   L = List mounted drives
   M = Mount drive
   U = Unmount drive
   X = Exit back to FLEX

Enter your command: Exit

+++
```

MOUNT Bugs/"Features"

- When mounting a file, the name is not checked for validity. Ie, a name of "I have 3 cats" is accepted. Use names given by the "D" command.
- There is no check to verify a file is mounted only once.
- When mounting a file, the full filename must be given, including the ".DSK" extension.
- No support for read-only file systems. There is no way to mount a drive read-only, nor does the list of mounted files indicate if any are read-only.

NEWDISK

This command is used to create new DSK format files that can be mounted. The FLEX operating system handles different sized disks extremely well; a major improvement over other operating systems of the time! Each disk has a single sector that has information about the disk parameters, so the NEWDISK command allows the user to format a number of different sized disks and then write the appropriate information to the disk so FLEX can use it.

The source code for NEWDISK is included, so you can add your own custom disk formats, but the default formats are:

- 88K (34 tracks, 10 sectors per track)
- 256K (32 tracks, 32 sectors per track)
- 500K (76 tracks, 26 sectors per track)
- 1.4M (79 tracks, 72 sectors per track)

<<SAMPLE SESSION, format a disk, mount it>>

NEWDISK Bugs/"Features"

- Input is not validated. Filenames are not checked to be valid, volume names are not checked, etc.
- It does not verify if the new file exists already or not. Ie, it's possible to damage an existing file without warning.
- Actually an Arduino bug, but if the name of existing file is provided, the file is not deleted first, but the new format is added to the end. The extra space is not usable.

SDDIR – not done yet!

Does a directory of the SD disk.

SDTYPE – not done yet!

Displays the contents of an SD file. Best used on text files, not binary files.

SETCLOCK

This utility's sole purpose is to set the hardware real time clock. When executed, it will get the current time, date and day-of-week from the RTC display the values, and allows you to change the values. This is not used very often, usually when the system is first set up for a new time zone.

The hardware real time clock is a Dallas DS3231 which has a very stable time base, so it should drift very little over extended periods of time. Since FLEX only uses the date, this clock should be more than sufficient accuracy for most applications.

Note that FLEX only knows the current date and will not automatically query the RTC to get the current date. If you use the DATE command, then leave your system running overnight, FLEX won't know to update the current date unless you use the DATE command again. Every time I sit down in front of my system, I do a DATE command just to make sure the date is accurate.

```
+++SETCLK

Set Time Utility

Current time and date: 20:03:52 8/22/15
For each field you can either type a new value or use the
default value
(in parenthesis) by simply pressing the return key.

Month (8): 8
Day of month (22): 22
Last two digits of year (15): 15
Hour - 24 hour time (20):
Minutes (3): 4
Seconds (52): 0
Day of week.  1=Sunday, 2=Monday, etc. (7):

The new time & date have been set

+++DATE
20:04:02, Saturday, August 22, 2015
+++
```

SETCLOCK Bugs/"Features"

- It does not validate data entry. Ie, a month of 999 is accepted but might produce strange results.
- If the RTC board is removed, there is no warning.

Uh-Oh, I Did Something BAD!

We all make mistakes. Bad things happen. So how do you recover?

First, we strongly recommend copying the contents of the SD card to a directory on your personal computer before you even install the SD card system. If something gets damaged on the card, you can recover back the original files from your backups.

Something else to keep in mind is that it's easy to create a new blank disk with the NEWDISK command, mount it with the MOUNT command, and then copy files from the questionable disk to the new one.

As a last resort, the images of the DSK files are always on our website www.corshamtech.com for you to download.

There is nothing special about the micro SD card, so you can buy almost any micro SD, install the DSK files and SD.CFG file and it is now a bootable disk for the SD card system. The Arduino does not boot from the card, it just reads data files from it.

Porting to non-Corsham Tech Systems

Please reference The Remote Disk Protocol Guide for complete information on how the interface between the main processor and Arduino works.

Porting to Another 6800 or 6809 System

This is definitely the easiest case.

It is generally easiest to develop the interface functions using a RAM based test environment, then transfer the debugged code to EPROM so the functions are readily available for your OS and applications to access. We have found a jump table to these routines makes it easiest to maintain compatibility as changes are being made to the code.

The file PARPROTO.INC has equates for command codes, responses, error codes, as well as the vectors for the jump table to common routines. You will definitely need to change the base address of the jump tables to suit where your code is located in memory, but the rest of the file should remain unchanged.

The subroutines in PARIO.ASM need to be ported to your particular hardware, and then placed at a fixed location in memory so higher level code knows where to find them. The included PARIO.ASM is based on a 6821 PIA using four memory locations. Functions in this file are aware of hardware and physical access to it.

The subroutines in DISKFUNC.ASM are higher level functions that perform complete actions as opposed to simply moving bytes to/from the Arduino. They call the functions in PARIO.ASM but do not contain any references to specific hardware. They can also be ported to any location, but they need to link to the low-level subroutines in PARIO.ASM. You might wish to add or remove functionality in this file to suit your needs but the sector read function is usually needed by the operating system loader to bring in the operating system.

All of the software written by uses a jump table that is at a known location. This is used by the FLEX boot sector and all utilities. The location is in ROM for our board and is always at address E40A.

<<INSERT JUMP TABLE HERE>>

Porting to a Non-6800/6809 System

This section is meant to assist someone wishing to port the existing code to another system that is not based on the 6800 or 6809 processors. Since the Arduino code knows nothing about the main processor, it should be fairly easy to follow the 6800 or 6809 code examples and use them as guides when developing your code.
The lowest level that needs to be matched is the hardware interface to the Arduino, which uses 8 bi-directional signals and three control lines. Two of the control lines are always output from the main processor and the other control line is always an input. The eight data lines change direction depending on which processor is sending data.

Generally speaking, you'll need to develop five low-level functions that directly drive the hardware:

INIT – This initializes the hardware so it can be used for communication to the remote disk controller.

SET WRITE – Prepares the hardware for sending a message to the remote disk controller. This effectively indicates the start of an outbound message that is called before each message is sent. When the interface is idle, it should be in write mode since the host starts all transactions, saving the time needed to switch back to write mode at the start of each transaction.

SET READ – This prepares the hardware for receiving a message. This is called after a message has been sent and the host is preparing to receive the response from the remote disk controller.

WRITE BYTE – Sends a single byte to the remote disk processor. Does not return until the byte is either sent or queued for transmission. This should never be called unless SET WRITE has been called first.

READ BYTE – Reads a single byte from the remote disk processor. Does not return until a byte is received. Note that this should never be called unless a SET READ has been done first.

Once the low level drivers are done, an easy test is to send the PING command and get the PONG response back. This simple exchange of two bytes proves that the interface is working properly. It is highly recommended to start with PING/PONG for debugging, and not move to other high level functions until this works repeatedly.

We put a number of higher level functions into the file DISKFUNC.ASM so the user can call known-good functions rather than creating their own using low level functions.

For booting a higher level operating system, a function to read a sector is usually required so this is a probably the next function to write. Beyond that, what functions are provided by the library are up to the developer.

ARDUINO Code

The complete Arduino source code is in the directory title “Arduino Mega Code” and can be read into your Arduino development environment. Please read the comments in the source code.

Please reference The Remote Disk Protocol Guide for more information.

ARDUINO Code Bugs/”Features”

- Only supports 256 byte sector sizes. Some commands indicate a sector size and this should be properly set, but any value other than the one for 256 bytes will not work.
- When opening a file for writing (WRITE_FILE command), if there is an existing file with the same name, it will be appended to rather than deleted and recreated.
- Support for read-only disk images is limited and should not be relied upon.
- The code is currently single threaded with no interrupts, so if a disk operation hangs, the software timers (needed for OS/9) will stop ticking.

Getting More Information and Additional FLEX DSK Images

FLEX, despite being a rather old operating system, has a devoted following, mostly at the FLEX User Group website:

www.flexusergroup.com

A huge collection of DSK images is available on their FTP site, but you do need a username (flexuser) and password (flex) to access it:

ftp://ftp.flexusergroup.com

Many other sites have FLEX format disk images, so search around to find more.

Revision History

Documentation

Version	Changes
1	Initial Beta.
A	First official release.

Circuit Board

Version	Changes
1	Initial Beta.
A	First official release.
3	Stable release.
4	Adds option switches and lines to support timer interrupts from the Arduino back to the main processor.

Errata

R1, R2, and R3 are now 150 ohms, not 100 ohms as mislabeled on the PCB.

Parts List – INCOMPLETE!

Part	Number	Description
PCB	1	Printed Circuit Board (Corsham Tech)
R1, R2, R3	3	150 ohm, ¼ watt resistor
R4	1	10K, ¼ watt resistor
C1	1	.1uf disc capacitor
LED1	1	Red 3mm LED
LED2	1	Yellow 3mm LED
LED3	1	Green 3mm LED
IC1	1	74HC4050
	1	16 pin socket for IC1
S1	1	Reset button (need better description)
J1	1	2x13 male connector (need part numbers)
J2	1	Molex 1040310811 SD socket

RTC	1	Raspberry Pi DS3231 RTC module
	1	1x5 connector for RTC
H1	1	Arduino Mega connector set