sd2iec - a controller/interface adapting storage devices to the CBM serial bus

Parts based on code from others, see comments in main.c for details.
JiffyDos send based on code by M.Kiesel
Fat LFN support and lots of other ideas+code by Jim Brain
crc7.c generated by pycrc, see comments in it for pycrc licence.
Final Cartridge III fastloader support by Thomas Giesel

FIXME: This file still needs to be expanded. A lot.

Introduction

sd2iec is firmware, used in hardware designs like MMC2IEC, SD2IEC, or uIEC, that allows the Commodore serial bus to access removable storage devices (MMC, SD, CF) - think of it as a 1541 with a modern storage medium instead of disks. The project was inspired by (and uses a few bits of code from) MMC2IEC[1] by Lars Pontoppidan and once ran on the same hardware before it grew too big for the ATmega32 used there.

Currently, the firmware provide good DOS and file-level compatibility with CBM drives, but much work remains. Unless specifically noted, anything that tries to execute code on the 1541 will not work, this includes every software fastloader.

[1] Homepage: http://pontoppidan.info/lars/index.php?proj=mmc2iec

Please note: Whenever this file talks about "D64 images" the text applies to all Dxx image types, i.e. D64/D71/D81 unless specifically noted.

Supported commands

- General notes:
  Any command not listed below is currently not supported.

- Directory filters:
  To show only directories, both **=B** (CMD-compatible) and **=D** can be used.
  On a real Commodore drive D matches everything.
  To include hidden files in the directory, use ***=H** - on a 1541 this doesn't do anything.
  sd2iec marks hidden files with an H after the lock mark, i.e. "**PRG<H**" or "**PRG H**".
  CMD-style "short" and "long" directory listings with timestamps are supported ("**$=T**"), including timestamp filters. Please read a CMD manual for the syntax until this file is updated.

- Partition directory:
  The CMD-style partition directory ($=P) is supported, including filters ($=P:S*). All partitions are listed with type "FAT", although this could change to "NAT" later for compatibility.

- **CD** / **MD** / **RD**
  Subdirectory access is compatible to the syntax used by the CMD drives, although

drive/partition numbers are completely ignored.
Quick syntax overview:
CD:_ …changes into the parent dir (_ is the left arrow ← on the C64)
CD_ …ditto
CD:foo …changes into foo
CD/foo …ditto
CD//foo …changes into \foo
CD/foo/:bar …changes into foo\bar
CD/foo/bar …ditto

You can use wildcards anywhere in the path. To change into an M2I or D64 image the image file must be named after the **:**, it will not be recognized otherwise.

MD uses a syntax similar to CD and will create the directory listed after the colon (**:**) relative to any directory listed before it.

MD/foo/:bar …creates bar in foo
MD//foo/:bar …creates bar in \foo

RD can only remove sub-directories of the current directory.
RD:foo …deletes foo

CD is also used to mount/unmount image files. Just change into them as if they were a directory and use **CD:_** (**left arrow** on the C64) to leave. Please note that image files are detected by file extension and file size and there is no reliable way to see if a file is a valid image file.

- **CP**, **C<Shift-P>**
  This changes the current partition, see "Partitions" below for details.

- **C**
  File copy command, should be CMD compatible. The syntax is
  C[partition]path]:targetname=[[partition][path]:]sourcename[,[[p][p]:]sourcename...]
  You can use this command to copy multiple files into a single target file in which case all source files will be appended into the target file. Parsing restarts for every source file name which means that every source name is assumed to be relative to the current directory.

  You can use wildcards in the source names, but only the first file matching will be copied.

  Copying REL files should work, but isn't tested well. Mixing REL and non-REL files in an append operation isn't supported.

- **D**
  Direct sector access, this is a command group introduced by sd2iec. Some Commodore drives use D for disk duplication between two drives in the same unit, an attempt to use that command with sd2iec should result in an error message.

  D has three subcommands: **DI** (Info), **DR** (Read) and **DW** (Write).
  Each of those commands requires a buffer to be opened (similar to U1/U2), but due to the larger sector size of the storage devices used by sd2iec it needs to be a large

buffer of size 2 (512 bytes) or larger. The exception is the **DI** command with page set to 0, its result will always fir into a standard 256 byte buffer.
If you try to use one of the commands with a buffer that is too small a new error message is returned, "78,BUFFER TOO SMALL,00,00".
In the following paragraphs the secondary address that was used to open the buffer is called "bufchan".

- **DI**
  In BASIC notation the command format is "DI"+chr$(bufchan)+chr$(device)+chr$(page)
  "device" is the number of the physical device to be queried,
  "page" the information page to be retrieved. Currently the only page implemented is page 0 which will return the following data structure:
  1 byte: Number of valid bytes in this structure
  This includes this byte and is meant to provide backwards compatibility if this structure is extended at a later time. New fields will always be added to the end so old programs can still read the fields they know about.
  1 byte: Highest diskinfo page supported
  Always 0 for now, will increase if more information pages are added (planned: Complete ATA IDENTIFY output for IDE and CSD for SD)
  1 byte: Disk type
  This field identifies the device type, currently implemented values are:
  0 IDE
  2 SD
  3 Dataflash
  1 byte : Sector size divided by 256
  This field holds the sector size of the storage device divided by 256.
  4 bytes: Number of sectors on the device
  A little-endian (same byte order as the 6502) value of the number of sectors on the storage device. If there is ever a need to increase the reported capacity beyond 2TB (for 512 byte sectors) this field will return 0 and a 64-bit value will be added to this diskinfo page.

  If you want to determine if there is a device that responds to a given number, read info page 0 for it. If there is no device present that corresponds to the number you will see a DRIVE NOT READY error on the error channel and the "number of valid bytes" entry in the structure will be 0.

  Do not assume that device numbers are stable between releases and do not assume that they are continuous either. To scan for all present devices you should query at least 0-7 for now, but this limit may increase in later releases.

- **DR** / **DW**
  In BASIC notation the command format would be
  "DR"+chr$(bufchan)+chr$(device)
  +chr$(sector AND 255)
  +chr$((sector/256) AND 255)
  +chr$((sector/65536) AND 255)
  +chr$((sector/16777216) AND 255)
  (or "DW" instead of "DR) but this won't work on the C64 because AND does not accept parameters larger than 32767. The principle should be clear though, the last four

bytes are a 32 bit sector number in little-endian byte order.

DR reads the sector to the buffer, **DW** writes the contents of the buffer to the sector. Both commands will update the error channel if an error occurs, for **DR** the 20,READ ERROR was chosen to represent read errors; for write problems during **DW** it sets 25,WRITE ERROR for errors and 26,WRITE PROTECT ON if the device is read-only.

- **G-P**
  Get partition info, see CMD FD/HD manual for details. The reported information is partially faked, feedback is welcome.

- **P**
  Positioning doesn't just work for REL files but also for regular files on a FAT partition. When used for regular files the format is
  "P"+chr$(channel)+chr$(hi)+chr$(mid)+chr$(lo)
  which will seek to the 0-based offset hi*65536+256*mid+lo in the file.

- **N**
  Format works only if a D64 image is already mounted. This command is limited to 1541 disk images, usually known as .D64.

- **R**
  Renaming files should work the same as it does on CMD drives, although the errors flagged for invalid characters in the name may differ.

- **S**
  Name matching is fully supported, directories are ignored. Scratching of multiple files separated by , is also supported with no limit to the number of files except for the maximum command line length (usually 100 to 120 characters).

- **T-R** and **T-W**
  If your hardware features RTC support the commands **T-R** (time read) and **T-W** (time write) are available. If the RTC isn't present, both commands return 30,SYNTAX ERROR,00,00; if the RTC is present but not set correctly T-R will return 31,SYNTAX ERROR,00,00.

  Both commands expect a fourth character that specifies the time format to be used, **T-W** expects that the new time follows that character in exactly the format returned by **T-R** with the same format char.

  The possible formats are:
  - "A"SCII: "SUN. 01/20/08 01:23:45 PM"+CHR$(13)
    The day-of-week string can be any of "SUN.", "MON.", "TUES", "WED.", "THUR", "FRI.", "SAT.". The year field is modulo 100.
  - "B"CD or "D"ecimal:
    Both these formats use 9 bytes to specify the time. For BCD everything is BCD-encoded, for Decimal the numbers are sent/parsed as-is.
    Byte 0: Day of the week (0 for sunday)
    1: Year (modulo 100 for BCD; -1900 for Decimal, i.e. 108 for 2008)
    2: Month (1-based)

3: Day (1-based)
4: Hour (1-12)
5: Minute (0-59)
6: Second (0-59)
7: AM/PM-Flag (0 is AM, everything else is PM)
8: CHR$(13)

When the time is set a year less that 80 is interpreted as 20xx.

- **U0**
  Device address changing with "U0>"+chr$(new address) is supported, other U0 commands are currently not implemented.

- **U1** / **U2** / **B-R** / **B-W**
  Block reading and writing is fully supported while a D64 image is mounted.

- **B-P**
  Supported, not checked against the original ROM at all.

- **UI+** / **UI-**
  Switching the slightly faster bus protocol for the VC20 on and off works, it hasn't been tested much though.

- **UI** / **UJ**
  Soft/Hard reset - **UI** just sets the "73,..." message on the error channel, **UJ** closes all active buffers but doesn't reset the current directory, mounted image, swap list or anything else.

- **U<Shift-J>**
  Real hard reset - this command causes a restart of the AVR processor (skipping the bootloader if installed). **<Shift-J>** is character code 202.

- **X**
  Extended commands. If you use JiffyDOS, you can send them by using **@**"**X...**" - without quotes you'll just receive an error.

- **Xenum**
  Sets the "file extension mode". This setting controls if files on FAT are written with an x00 header and extension or not.
  Possible values for num are:
  0: Never write x00 format files.
  1: Write x00 format files for SEQ/USR/REL, but not for PRG
  2: Always write x00 format files.
  3: Use SEQ/USR/REL file extensions, no x00 header
  4: Same as 3, but also for PRG
  If you set mode 3 or 4, extension hiding is automatically enabled. This setting can be saved in the EEPROM using XW, the default value is 1.

  For compatibility with existing programs that write D64 files, PRG files that have D64, D41, D71, D81 or M2I as an extension will always be written without an x00

header and without any additional PRG file extension.

- **XE+ / XE-**
  Enable/disable extension hiding. If enabled, files in FAT with a PRG/SEQ/USR/REL extension will have their extension removed and the file type changed to the type specified by the file extension - e.g. APPLICATION.PRG will become a PRG file named "APPLICATION", "README.SEQ" will become a SEQ file named "README".
  This flag can be saved in the EEPROM using **XW**, the default value is disabled (**-**).

- **XB+ / XB-**
  Enable/disable free block calculation for FAT32 drives.
  As the free block calculation for FAT32 can take a lot of time it can be disabled using **XB-**. If it is disabled, sd2iec will always report "1 BLOCKS FREE" for FAT32 drives. The free block calculation on FAT12/FAT16 isn't affected because it takes just two seconds at most. This flag can be saved in the EEPROM using **XW**, the default value is enabled (**+**).

- **XJ+/XJ-**
  Set or reset the JiffyDOS protocol support flag. This flag can be saved in the EEPROM using **XW**, the default value is enabled (**+**).

- **X*+ / X*-**
  Enable/disable 1581-style ***** matching. If enabled, characters after a ***** will be matched against the end of the file name. If disabled, any characters after a ***** will be ignored. This flag can be saved in the EEPROM using **XW**, the default value is enabled (**+**).

- **Xcnum**
  Set oscillator calibration value to num (must be between 0 and 255). Default is whatever your chip defaults to. This value can be saved in the EEPROM using **XW**.

- **Xddrv=val**
  Configure drives. On ATA-based units or units with multiple drive types, this command can be used to enable or reorder the drives. **drv** is the drive slot (0-7), while val is one of:
  0: Master ATA device
  1: Slave ATA device
  4: Primary SD/MMC device
  5: Secondary SD/MMC device
  6: Dataflash device
  15: no device

  Note that only devices supported by the specific hardware can be selected. Unsupported device types will return an error if requested. Also, note that you cannot select a device in multiple drive slots. Finally, while it is possible to re-order ATA devices using this functionality, it is strongly discouraged. Use the master/slave jumpers on the ATA devices instead. To reset the drive configuration, set all drive slots to "no device". This value can be permanently saved in the EEPROM using **XW**.

- **XD?**

View the current drive configuration. Example result:
"03,D:00=04:01=00:02=01,10,01". The track indicates the current device address, while the sector indicates extended drive configuration status information.

- **X**
  X without any following characters reports the current state of all extended parameters via the error channel, similar to DolphinDOS. Example result:
  "03,J-:C152:E01+:B+:*+,08,00"
  The track indicates the current device address.

- **XS:name**
  Set up a swap list - see "Changing Disk Images" below.
  XS …Disable swap list

- **XW**
  This commands stores the current configuration in the EEPROM. It will automatically be read when the AVR is reset, so any changes you made will persist even after turning off the hardware.

  The stored configuration includes the oscillator calibration value, the JiffyDOS protocol support flag, the extension mode and the current device address. If you have changed the device address by software, sd2iec will power up with that address unless you have changed the device address jumpers (if available) to a different setting than the one active at the time the configuration was saved. You can think of this feature as changing the meaning of one specific setting of the jumpers to a different address if this sounds logical enough to you.

  The "hardware overrides software overrides hardware" priority was chosen to allow accessing sd2iec even when it is soft-configured for a device number that is already taken by another device on the bus without having to remove that device to reconfigure sd2iec (e.g. when using a C128D).

- **X?**
  Extended version query. This commands returns the extended version string which consists of the version, the processor type set at build time and the suffix of the configuration file (usually corresponds to the short name of the hardware sd2iec was compiled for).

- **M-R**, **M-W**, **M-E**
  Memory reading returns random data of the requested length. Memory writing knows about the address used for changing the device address on a 1541 and will change the address of sd2iec to the requested value. It will also check if the transmitted data corresponds to any of the known software fastloaders so the correct emulation code can be used when **M-E** is called.

- **E-R**, **E-W**
  Both commands work like **M-R** and **M-W**, but instead of reading/writing RAM they allow access to a user-area of the EEPROM. This area currently holds 512 bytes and accesses beyond its end will result in a 32 SYNTAX ERROR. It is strongly recommended to work on a protocol for sharing this area between multiple applications that want to store their configuration in there, but that is beyond the

scope of this project.

As the contents of the EEPROM have to be copied to RAM before they can be sent to the computer it is not possible to read more data with a single command than the error message buffer (default size: 36 bytes) can hold. Similarly, writing is restricted by the size of the command buffer (at least 42 bytes for compatibility, expected to be at least 100 bytes in release versions).

The user-area does not interfere with the stored configuration (**XW**) in any way.

Large buffers

To support commands which directly access the storage devices support for larger buffers was added. A large buffer can be allocated by opening a file named "##<d>" (exactly three characters" with <d> replaced by a single digit specifying the number of 256-byte buffers to be chained into one large buffer - e.g. "##2" for a 512 byte buffer, "##4" for 1024 bytes etc. Unlike a standard buffer where the read/write pointer is set to byte 1, a large buffer will start with the r/w pointer pointing to byte 0 because that seems to be more sensible to the author.

If there aren't enough free buffers to support the size you requested a 70,NO CHANNEL message is set in the error channel and no file is opened. If the file name isn't exactly three bytes long a standard buffer ("#") will be allocated instead for compatibility.

The B-P command supports a third parameter that holds the high byte of the buffer position, for example, "B-P 9 4 1" positions to byte 260 (1*256+4) of the buffer on secondary address 9.

No command is implemented at this time which uses large buffers.

Long File Names

Long file names (i.e names not within the 8.3 limits) are supported on FAT, but for compatibility reasons the 8.3 name is used if the long name exceeds 16 characters. If you use anything but ASCII characters on the PC or their PETSCII equivalents on the Commodore you may get strange characters on the other system because the LFN use unicode characters on disk, but sd2iec parses only the low byte of each character in the name.

Partition

sd2iec features a multi-partition support similiar to that of the CMD drives. The partitions (which may be on separate drives for some hardware configurations) are accessed using the drive number of the commands sent from the computer and are numbered starting with 1. Partition 0 is a special case: Because most software doesn't support drive numbers or always sends drive number 0, this partition points to the currently selected partition. By default, accesses to partition 0 will access partition 1, this can be changed by sending "CP<num>" over the command channel with <num> being an ASCII number from 1 to 255. "C<Shift-P" (0x42 0xd0) works the same, but expects a binary partition number as the third character of the command.

Software fastloaders

Turbodisk
Turbodisk is detected by the CRC of its 493 byte long floppy code and the M-E address 0x0303. The same code seems to be used under various names, among them "Turbodisk" (both 2.1 and 2.2) and "Fast-Load". Unfortunately the timing requirements are extremely tight and cannot be met with the internal RC oscillator of the AVR even if calibrated. You really need an external 8MHz crystal or the data read by the C64 will be gibberish. It is not known if there is an NTSC-compatible version of this fastloader.

Final Cartridge III
Both the fast loader and the fast saver of Final Cartridge III are supported. The fast loader seems to work without a crystal, the fast saver was not tested. It is not known if there is an NTSC-compatible version using the same drive code, but at least for the fastloader the FC3 ROM seems to contain two different C64-side implementations.

The slightly different fastloader used for files freezed with the FC3 is also supported.

EXOS V3 and **The Beast System**
Both supported, the loader used by these kernals is very similar to the FC3 fast loader.

Action Replay 6
The AR6 reads a byte from the drive rom to check which fastloader it should use. sd2iec returns a value that should force the cartridge to use the standard kernal loader instead of its (still unsupported) fastloader.

Dreamload
Dreamload uses direct track/sector access, so it is only supported on D64 or similiar disk image formats. As sd2iec has to wait for commands from the C64 constantly the disk change buttons may become unresponsive, try multiple times if you need to. Dreamload is a "captive" fastloader, sd2iec stay in Dreamload mode until it receives a "quit loader" command from the C64. To force sd2iec to resume normal operation, hold the disk change button until the red LED turns on (just like sleep mode).

Please note that Dreamload does not work with more than one device on the serial bus due to the way it uses the ATN line.

JiffyDOS
The JiffyDOS protocol has very relaxed timing constraints compared to Turbodisk, but still not as relaxed as the standard Commodore IEC protocol. Jiffy seems to tolerate slightly mis-tuned RC oscillators, but you still shouldn't expect it to work without oscillator calibration. If the frequency error is too big you WILL get wrong data which usually manifests as a FILE NOT FOUND error because the name the drive received was already garbled.

x00 files
P00/S00/U00/R00 files are transparently supported, that means they show up in the directory listing with their internal file name instead of the FAT file name. Renaming them only changes the internal name. The XE command defines if x00 extensions are used when writing files, by default sd2iec uses them for SEQ/USR/REL files but not for PRG. Parsing

of x00 files is always enabled even when writing them is not.

x00 files are recognized by checking both the extension of the file (P/S/U/R with a two-digit suffix) and the header signature.

Disk Images
Disk images are recognized by their file extension (.D64, .D41, .D71, .D81) and their file size (must be one of 174848, 175531, 349696, 351062, 819200). If the image has an error info block appended it will be used to simulate read errors. Writing to a sector with an error will always work, but it will not clear the indicated error. D81 images with error info blocks are not supported.

M2I files
M2I files are fully supported. sd2iec supports SEQ and USR files in this format in addition to PRG and DEL which were already implemented in MMC2IEC. For compatibility reasons the file type is not checked when opening files. Inside an M2I file the files are always shown as 0 (DEL) or 1 blocks because calling stat for every file was slowing down the directory listing too much. For compatibility with existing M2I files the data files do not use P00 headers even when the file type is SEQ or USR.

REL files
Partial REL file support is implemented. It should work fine for existing files, but creating new files and/or adding records to existing files may fail. REL files in disk images are not supported yet, only as files on a FAT medium. When x00 support is disabled the first byte of a REL file is assumed to be the record length.

Changing Disk Images
Because some programs require more than one disk side there is support for changing the currently mounted disk image with a button connected to the disk change pin.

If your circuit doesn't have a disk change pin/button you might be able to add it yourself:

- For the original MMC2IEC and the NKC MMC2IEC:
  Connect a button from PA4 to ground. PA4 is pin 36 on the DIL version of the controller or pin 33 on the surface-mount version.
- For Shadowolf's MMC2IEC 1.x PCBs:
  Connect a button from PC4 to ground. PC4 is pin 23 on the DIL version of the controller or pin 23 on the surface-mount version.
- Any other circuit without disk change pin on a convenient connector somewhere and no button dedicated to that function:
  Please check with the supplier of the board and read config.h in the sources to find out how to connect it.

To use this functionality, create a text file that lists the file names of all disk images you want to swap between, one per line. The file names are parsed in the same way as the **CD** command, so you can include a path to the image if desired.

Examples:
=== example 1 ===
FOO.D64
BAR.D64
BAZ.D64

=== end of example 1 ===

=== example 2 ===
//NEATGAME/:DISK1A.D64
//NEATGAME/:DISK1B.D64
//NEATGAME/:DISK2A.D64
//NEATGAME/:DISK2B.D64
=== end of example 2 ===

The swap list is enabled by sending "XS:filename" over the command channel with filename being the name of the image list you created, parsed in the same way as any other file name.

After sending **XS** the first image in the list is automatically mounted. To switch to the next image in the list, push the button. If the new image was mounted successfully both LEDs will blink twice. When you've reached the last image in the list pushing the button will mount the first image again. All of this is completely compatible with normal image mounting/unmounting, so you can unmount the disk image any time you want and resume the mount cycle later by pushing the button.

Due to the way this feature is implemented you are not limited to a swap list containing just D64 images, M2I and even FAT directories will work too.

FIXME: Does that still work?

If you press the button when no list has been set before or when the previous list was cleared by sending **XS** the software will look for a file called **AUTOSWAP.LST** in the current (FAT-)directory and use this as the current swap list until you deactivate it or manually change the directory (otherwise an **AUTOSWAP.LST** in the new directory would be ignored until you send **XS**, killing the nice "it just works" feeling).

FIXME: Integrate the following into the preceding

The second disk change button is on

- PA5 for LarsP/NKC
- PC3 for Shadowolf's MMC2IEC 1.x
- PC2 for Shadowolf's sd2iec 1.x ("Reserve" on the header)
- PG4 for uIEC

Either of those buttons will trigger the use of AUTOSWAP.LST.
If a swap list is already active, the first button will switch to the next image in the list, the second button will switch to the previous image in the list and pushing both buttons together will switch to the first image in the list.

The confirmation blink is red+green followed by green for "next", by red for "previous" and by red+green for "first".

Sleep Mode

If you hold the disk change button down for two seconds, sd2iec will enter "sleep mode". In this mode it doesn't listen to the bus at all until you hold down the disk change button for

two seconds again which resumes normal operation. Sleep mode allows you to keep sd2iec connected to the serial bus even when you load something from a different drive that uses a fast loader that doesn't work with more than one device on the bus.

While sleep mode is active, the red LED will be on and the green LED will be off.

Other important notes

- When you hold down the disk change (forward) button during power on the software will use default values instead of those stored in the EEPROM.

- File overwrite (@foo) is implemented by deleting the file first.

- File sizes in the directory are in blocks (of 254 bytes), but the blocks free message actually reports free clusters. It is a compromise of compatibility, accuracy and code size.

- If known, the low byte of the next line link pointer of the directory listing will be set to (filesize MOD 254)+2, so you can calculate the true size of the file if required. The 2 is added so it can never be mistaken for an end marker (0) or for the default value (1, used by at least the 1541 and 1571 disk drives).

- If your hardware supports more than one SD card, changing either one will reset the current partition to 1 and the current directory of all partitions to the root drive. Doing this just for the card that was changed would cause lots of problems if the number of partitions on the previous and the newly inserted cards are different.

Compilation notes

sd2iec requires avr-libc version 1.6.x.

sd2iec is set up to be compiled in multiple configurations, controlled by configuration files. By default the Makefile looks for a file named 'config', but you can override it by providing the name on the make command line with "make CONFIG=filename". If you are using *BSD you may have to edit the Makefile to use another awk implementation instead of gawk - unfortunately WinAVR compatibility requires using gawk in there.

An example configuration file named "config-example" is provided with the source code, as well as abridged files corresponding to the release binaries. If you want to compile sd2iec for a custom hardware you may have to edit config.h too to change the port definitions.