

DOSonCHIP™ Host Interface Specification

Revision History

Date	Document Version	Firmware Version	Notes
February 23, 2009	0.1	2.02	Initial Draft

DOSonCHIP protocol	3
<i>Interfacing over SPI</i>	4
<i>Interfacing over UART</i>	4
<i>Protocol basics</i>	4
Conventions and common values	6
Command dictionary	7
<i>Basic Commands</i>	7
DOS_DOS_CMD_GET_ID	7
DOS_CMD_GET_VERSION	7
DOS_CMD_GET_TIME	8
DOS_CMD_SET_TIME	8
DOS_CMD_SET_TIME_ON_OFF	8
DOS_CMD_MOUNT	8
DOS_CMD_GET_FREE_SECTORS	8
DOS_CMD_DIR	9
DOS_CMD_DIR_GET_PROPERTY	9
DOS_CMD_MAKE_DIR	9
DOS_CMD_SET_DIR	10
DOS_CMD_DELETE	10
DOS_CMD_SET_HANDLE	10
DOS_CMD_OPEN_READ	10
DOS_CMD_OPEN_WRITE	10
DOS_CMD_SEEK	11
DOS_CMD_WRITE_PREALLOCATE	11
DOS_CMD_CLOSE	11
<i>Complex commands</i>	12
DOS_CMD_SET_NAME	12
DOS_CMD_GET_NAME	12
DOS_CMD_READ	13
DOS_CMD_WRITE	14
Appendix: DOSprotocol.h (version 2.02)	15

DOSonCHIP protocol

The DOSonCHIP supports interfacing over UART or SPI mode. All communication with the DOSonCHIP module uses a common packet-based protocol, regardless of the host interface chosen. Details of the physical connections are provided in the data sheet document.

The protocol described in this document is implemented in the DOSonCHIP Host API. When this document conflicts with the Host API C source, the source code is correct.

Throughout this document various constant values are referred to by their name, denoted in all caps with a prefix of “DOS_”. The values for these constants are found in the DOSonCHIP Host API source code in the file named DOSprotocol.h.

All communication with the DOSonCHIP uses big-endian format. The convention used in this document is that four byte sequences are composed of bytes numbered 3 through 0, where 3 is the first byte transmitted to or received from the DOSonCHIP and 0 is the last byte.

Interfacing over SPI

The DOSonCHIP operates as a SPI slave mode device. SPI mode interfacing is performed with SPI mode 0, 8 bits per word, most significant bit first. See the datasheet for details on supported clock rates. All SPI mode communication is half-duplex; when reading a byte from the DOSonCHIP, the host must send zeros.

Interfacing over UART

See the datasheet for details on supported baud rates. Hardware flow control may be used (again see the datasheet for specifics), but software flow control is not supported.

After connecting to the DOSonCHIP for the first time, send two carriage returns (ASCII character 0xD) with a short delay between them to establish the baud rate.

Protocol basics

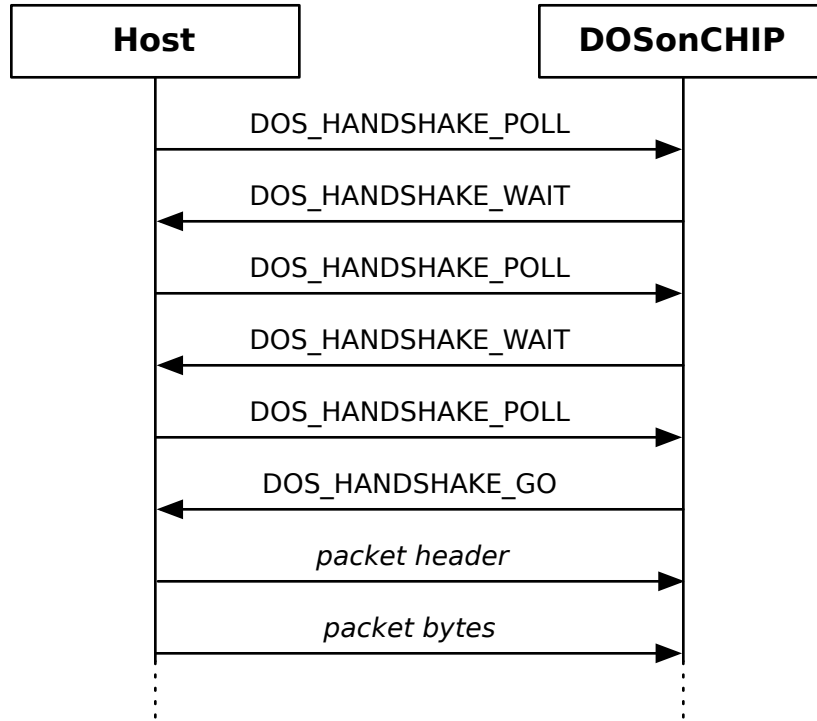
All communication with the DOSonCHIP is initiated by the host. To initiate any communication (whether a read or a write), the host must send the `DOS_HANDSHAKE_POLL` byte to the DOSonCHIP. The host must then read a byte from the DOSonCHIP device. If the byte read is `DOS_HANDSHAKE_WAIT`, the host must send `DOS_HANDSHAKE_POLL` again and read another response byte until a valid response is read.

When the DOSonCHIP returns with `DOS_HANDSHAKE_GO`, the host is expected to initiate a packet write by sending a command byte followed by some number of argument bytes. Otherwise, the value returned is the command byte of a packet to be read, and the host must read the argument bytes associated with that kind of packet.

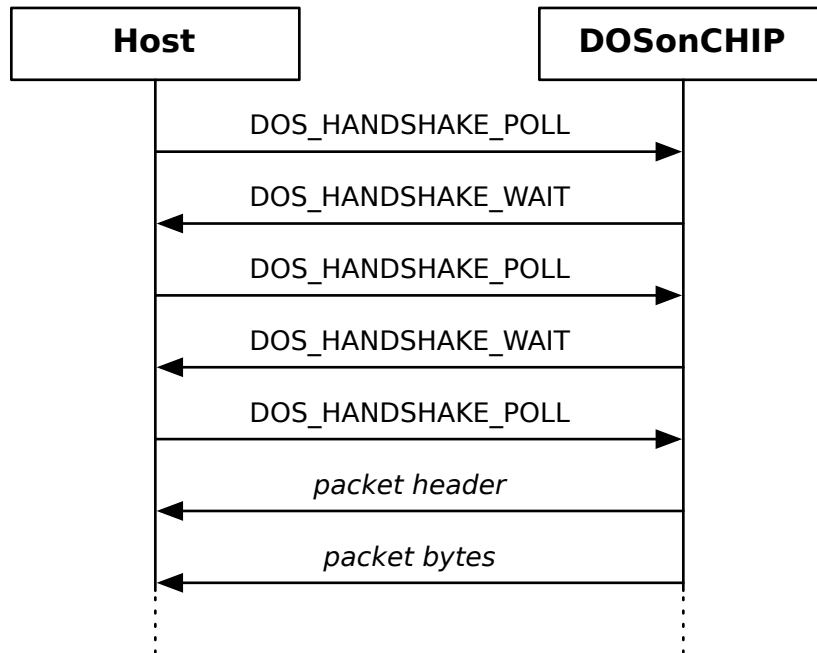
Some commands such as card mounting may take several seconds to process. It is recommended to set a timeout of no less than 30 seconds when waiting to receive a response packet from the DOSonCHIP.

All multi-byte values sent to or returned from the DOSonCHIP are in big-endian format.

The following diagrams show the process of sending a packet or receiving a packet from the DOSonCHIP. All communication with the DOSonCHIP uses this polling sequence. When diagrams appear later in this document using unfilled arrow ends on communication lines, each line represents a use of the following handshake sequence.



Writing a packet to the DOSonCHIP



Reading a packet from the DOSonCHIP

Conventions and common values

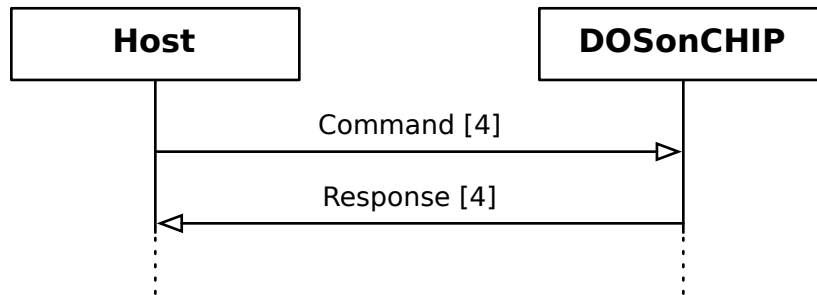
There are several important pieces of state carried by the DOSonCHIP which affect the interpretation of many commands. These are:

- **The mounted state of the card:** If the card has not been mounted, many commands will fail to operate.
- **The current date and time:** The current date and time is used when writing files to the card and can be retrieved from the DOSonCHIP. All time values on the DOSonCHIP are represented as a number of seconds since 12:00 A.M. January 1st, 1970. Due to the nature of the FAT filesystem, times stored on disk are interpreted in the local time zone, not the universal time. Thus, when setting the time on the DOSonCHIP or interpreting file modification or creation times returned from the DOSonCHIP, all times must be treated as relative to the local time zone.
- **The current file handle:** There are four file handles (0 through 3) available for use on the DOSonCHIP for simultaneous file access. Before opening, reading, writing, seeking, or closing a file, the `DOS_CMD_SET_HANDLE` command must be used to select a handle for the operation. The current setting of the handle persists across commands.
- **The current file name:** The DOSonCHIP has a single name buffer which is used when opening files, changing directories, and deleting files. It is set by the DOSonCHIP when iterating through the contents of a directory, and can be set or retrieved by user command. The file name is represented as a full 8.3 file name including the period, e.g. "FILENAME.TXT".
- **The current directory:** The current working directory is set to the root of the card when the card is mounted, and may be changed via the `DOS_CMD_SET_DIR` command.
- **The directory iterator:** The directory iterator is used to list the contents of a directory. It must be initialized before it can be used via the `DOS_CMD_DIR` command with the `DOS_FIRST` argument. The `DOS_CMD_DIR` command with the `DOS_NEXT` argument is used to set the iterator to the next file in the directory. While using the iterator, the `DOS_CMD_DIR_GET_PROPERTY` and `DOS_CMD_GET_NAME` commands may be used to retrieve information about the current entry in the iterator. The file pointed to by the iterator may be opened, but doing so will destroy the current state of the iterator.
- **Seek pointer:** Each file opened for reading has a current seek pointer, which is the position where read operations will start in the file. When the file is opened, the seek pointer is set to the beginning of the file. The `DOS_CMD_SEEK` command may be used to set the seek pointer in the file.

Command dictionary

Basic Commands

All basic commands consist of a single byte command identifier followed by a four byte command argument. The response to a basic command consists of a response byte indicating success or failure followed by a four byte response argument. Successful responses return DOS_RES_NOERROR followed by a four byte argument which is specific to the command. Unsuccessful responses return an error code followed by a four byte error value which may be useful for debugging.



The high-level interaction of an ordinary command, showing a four byte command argument and response argument. Unfilled arrows indicate that the protocol for sending or receiving packets is being used to transmit these values.

DOS_DOS_CMD_GET_ID

Returns the silicon version of the DOSonCHIP IC.

Command argument	Ignored
Response argument	Bytes 3-1: Ignored Byte 0: DOSonCHIP silicon version

DOS_CMD_GET_VERSION

Returns the bootloader and firmware version of the DOSonCHIP IC.

Command argument	Ignored
Response argument	Byte 3: DOSonCHIP bootloader major version Byte 2: DOSonCHIP bootloader minor version Byte 1: DOSonCHIP firmware major version Byte 0: DOSonCHIP firmware minor version

DOS_CMD_GET_TIME

Returns the current time value of the DOSonCHIP IC. Only has meaning when the date and time has been set and the real time clock has been enabled (via the DOS_CMD_SET_TIME_ON_OFF command).

Command argument	Ignored
Response argument	Bytes 3-0: The current date and time

DOS_CMD_SET_TIME

Sets the current time value of the DOSonCHIP IC. Will be updated automatically by the DOSonCHIP if the real time clock has been enabled by the DOS_CMD_SET_TIME_ON_OFF command.

Command argument	Bytes 3-0: The current date and time
Response argument	Ignored

DOS_CMD_SET_TIME_ON_OFF

Enables or disables the DOSonCHIP real time clock. By default, it is not enabled. Before enabling the clock, use the DOS_CMD_SET_TIME command to set the current date and time.

Command argument	Bytes 3-1: Ignored Byte 0: DOS_ON or DOS_OFF
Response argument	Ignored

DOS_CMD_MOUNT

Mounts the inserted SD/SDHC card, and finds free sectors (512 bytes) which may be used for writing files to the card. The argument limits the number of free sectors which will be found during the mount process. If the card will not be written to, pass an argument of 0. If the card will be used for writing, pass 0xFFFFFFFF for best performance when writing files. Returns a unique ID which may be used to identify the filesystem on the card.

Command argument	Bytes 3-0: Number of free sectors to find for use in writing files
Response argument	Bytes 3-0: Unique filesystem identifier of the card

DOS_CMD_GET_FREE_SECTORS

Returns the number of free sectors found on the card. Each sector is equal to 512 bytes.

Command argument	Ignored
Response argument	Bytes 3-0: The number of free sectors found on the card

DOS_CMD_DIR

Initializes or increments the directory iterator. If DOS_FIRST is supplied, the iterator is initialized to the first entry in the directory. If DOS_NEXT is supplied, the iterator (which must have already been initialized) is incremented to point at the next entry in the directory. If there are no more entries or if the iterator is initialized in an empty directory, DOS_RES_DIR_END is returned.

Command argument	Bytes 3-1: Ignored Byte 0: DOS_FIRST
Response argument	Bytes 3-1: Ignored Byte 0: The attributes of the current file, represented as mask of the DOS_ATTR values specified in the protocol header file

DOS_CMD_DIR_GET_PROPERTY

Retrieves a property of the current entry pointed to by the directory iterator.

Command argument	Bytes 3-1: Ignored Byte 0: The property to fetch: one of DOS_PROPERTY_SIZE, DOS_PROPERTY_TIME_CREATED or DOS_PROPERTY_TIME_MODIFIED
Response argument	For DOS_PROPERTY_SIZE: The size of the file in bytes (0 for a directory) For DOS_PROPERTY_TIME_CREATED and DOS_PROPERTY_TIME_MODIFIED: the creation or modification time of the file or directory

DOS_CMD_MAKE_DIR

Creates an empty directory on the card in the current directory. The name of the directory is specified via the name buffer.

Command argument	Ignored
Response argument	Ignored

DOS_CMD_SET_DIR

Sets the current directory to the directory named by the name buffer. If the name is “..”, the directory will be set to the parent of the current directory. If the name is “\”, the current directory will be set to the root directory of the card.

Command argument	Ignored
Response argument	Ignored

DOS_CMD_DELETE

Deletes the file named by the value of the name buffer. Deletion of directories is not currently supported.

Command argument	Ignored
Response argument	Ignored

DOS_CMD_SET_HANDLE

Sets the current file handle. Must be used before opening or closing a file or reading, writing, or seeking within an open file.

Command argument	Bytes 3-1: New handle value (0-3)
Response argument	Ignored

DOS_CMD_OPEN_READ

Open the file whose name is specified by the name buffer for reading, and assigns it to the current file handle. The seek pointer is initialized to the beginning of the file.

Command argument	Bytes 3-0: 0
Response argument	Bytes 3-0: Size of the file in bytes

DOS_CMD_OPEN_WRITE

Opens the file whose name is specified by the name buffer for writing, and assigns it to the current file handle. If the file does not exist, it is created.

Command argument	Bytes 3-0: 0
Response argument	Bytes 3-0: Size of the file in bytes

DOS_CMD_SEEK

Increments the seek pointer in the file by the specified number of bytes, or resets it to the beginning of the file if an increment of 0 is given. If the seek pointer would point past the end of the file, it is set to the end of the file. Seeking is not supported on files open for write.

Command argument	Bytes 3-0: Seek increment, or 0 to reset to beginning of file
Response argument	Ignored

DOS_CMD_WRITE_PREALLOCATE

Allocates space to a file opened for write. Use this operation before initiating a large write to ensure that the necessary sectors are assigned to the file before data is written. Space is allocated to the file immediately, but its contents are unspecified until a write command is used to fill the space.

Command argument	Bytes 3-0: Number of bytes to allocate to the file
Response argument	Bytes 3-0: New size of the file in bytes

DOS_CMD_CLOSE

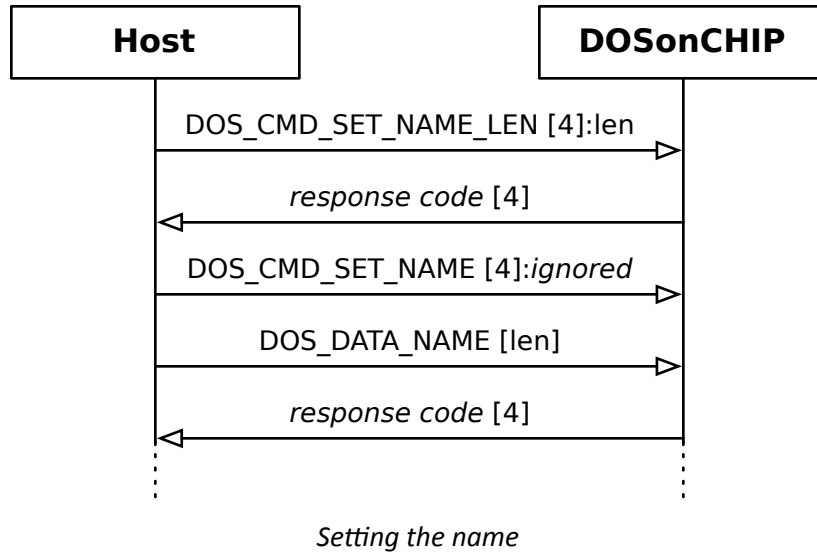
Closes the file specified by the current file handle.

Command argument	Ignored
Response argument	Ignored

Complex commands

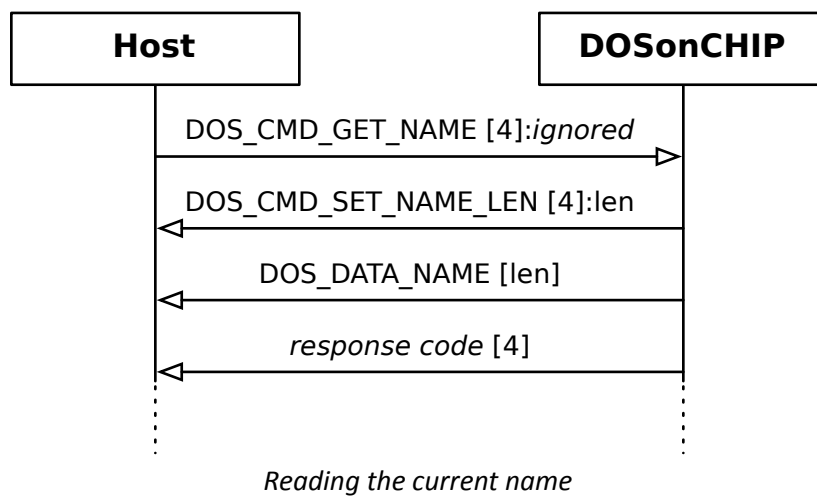
DOS_CMD_SET_NAME

Sets the value of the current name buffer. Before sending the name, the host must inform the DOSonCHIP of the length of the name to send (maximum of 12 bytes for a 8.3 file name).



DOS_CMD_GET_NAME

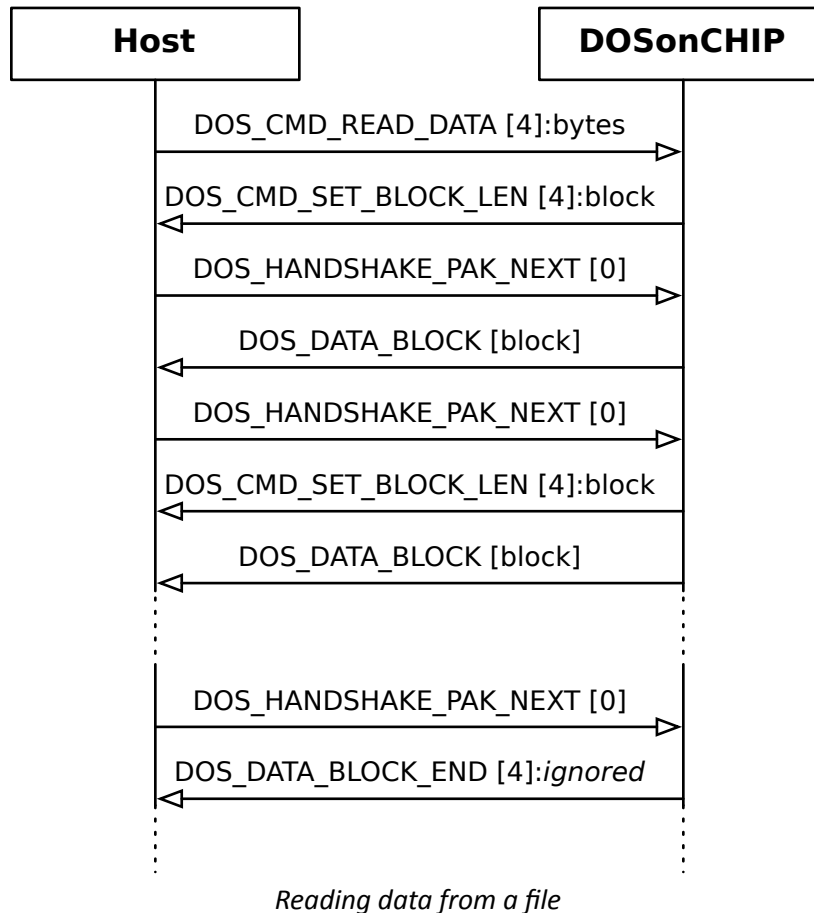
Retrieves the value of the current name buffer. Before receiving the name, the DOSonCHIP informs the host of the length of the name it will be sending (maximum of 12 bytes for a 8.3 file name).



DOS_CMD_READ

Reads data from a file open for reading (indicated by the current handle). The DOSonCHIP sends data in packets with variable size payload. Before sending the first block, the DOSonCHIP will inform the host of the size of the packet it will be sending. Thereafter it will inform the host when the size of the packet changes, but it may send several packets in a row without changing the size.

To receive packets, the host sends DOS_HANDSHAKE_PAK_NEXT and receives in response a data block or a DOS_DATA_BLOCK_END packet indicating the end of data. If any other response is received, it indicates an error during reading.

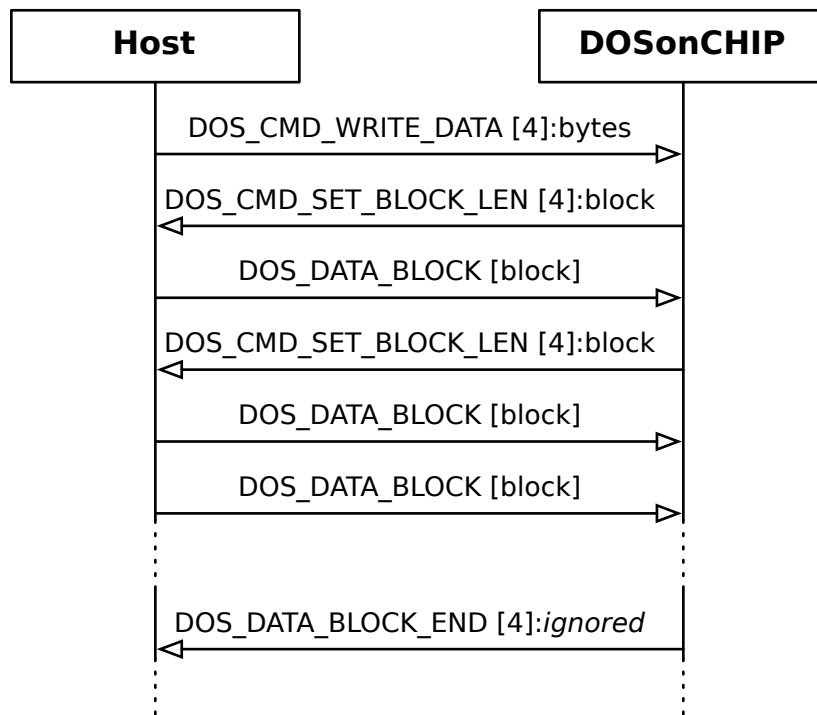


DOS_CMD_WRITE

Writes data to a file open for writing (indicated by the current handle). The host sends the data to the DOSonCHIP in a series of packets. Before any data is written, the DOSonCHIP tells the host the size of the data packet it is expecting. Thereafter it will inform the host when the size of the packet changes, but it may receive several packets in a row without changing the size.

When sending data to the DOSonCHIP, the host polls the DOSonCHIP via the normal handshake sequence. If it receives DOS_HANDSHAKE_GO, it sends a data block using the current block size. If it receives DOS_CMD_SET_BLOCK_LEN, it must read four bytes of response argument to determine the current block length before polling again to send a block.

Note: The upper two bytes of the block length must be ignored.



Writing data to a file

Appendix: DOSprotocol.h (version 2.02)

```

/*****
Copyright 2008, 2009 Wearable Inc.; An Illinois, United States Corporation

Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at

    http://www.apache.org/licenses/LICENSE-2.0

Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
*****/
/*****
Name:          DOSonCHIP(TM) Host API Definitions
File Name:     DOSprotocol.h
Description:   Host API Protocol Definitions
*****/

#ifndef DOSPROTOCOL_H
#define DOSPROTOCOL_H

/* DOSONCHIP IC CONSTANTS */
#define DOS_FIRMWARE_MAJOR      0x02 /* expected firmware (major) version */
#define DOS_FIRMWARE_MINOR     0x02 /* expected firmware (minor) version */
#define DOS_MAX_HANDLES        4    /* Firmware 2.x limit */

/* PACKET CONSTANTS */
#define DOS_PACKET_LENGTH_HEADER      1 /* single header byte */
#define DOS_PACKET_LENGTH_PAYLOAD_MAX 512 /* default & maximum DATA_BLOCK payload length */
#define DOS_PACKET_LENGTH_PAYLOAD_NAME 12 /* (8+1+3) default & maximum DATA_NAME packet payload length */
#define DOS_PACKET_LENGTH_PAYLOAD_CMDRES 4 /* command & response payload length */
#define DOS_PACKET_LENGTH_PAYLOAD_HANDSHAKE 0 /* handshake payload length */
#define DOS_PACKET_LENGTH_FOOTER      0 /* no footer at this time */

/* PACKET HEADER DEFINITIONS */
#define DOS_HANDSHAKE_OFF      0xFF
#define DOS_HANDSHAKE_WAIT    0x00
#define DOS_HANDSHAKE_POLL    DOS_HANDSHAKE_WAIT
#define DOS_HANDSHAKE_GO      0xEA
#define DOS_HANDSHAKE_PAK_NEXT 0xE0
#define DOS_HANDSHAKE_PAK_ABORT 0xE1

#define DOS_DATA_NAME          0x7B
#define DOS_DATA_BLOCK         0x7C
#define DOS_DATA_BLOCK_END     0x7D

#define DOS_COMMAND_CARD_ACCESS_MIN 0x5B /* NON-COMMAND: minimum command header byte value for a command requiring card access */

#define DOS_CMD_GET_ID          0x40 /* "@" : get the DOSonCHIP silicon version */
#define DOS_CMD_UPDATE          0x42 /* "B" : update firmware via bootloader */
#define DOS_CMD_GET_NAME        0x47 /* "G" : get current file/directory name */
#define DOS_CMD_SET_HANDLE      0x48 /* "H" : set the current handle number */
#define DOS_CMD_SET_BLOCK_LEN   0x4C /* "L" : set the byte length for transferring data in a single packet transfer */
#define DOS_CMD_SET_NAME_LEN    0x4D /* "M" : set the byte length for the file/directory name */
#define DOS_CMD_SET_NAME        0x4E /* "N" : set the current file/directory name */
#define DOS_CMD_SET_TIME_ON_OFF 0x52 /* "R" : enable/disable the real-time clock to commence incrementing */

```

```

#define DOS_CMD_SET_TIME                0x53 /* "S" : set the real-time clock date & time
*/
#define DOS_CMD_GET_TIME                0x54 /* "T" : get the real-time clock date & time
*/
#define DOS_CMD_GET_VERSION            0x56 /* "V" : get the DOSonCHIP firmware versions
(bootloader & filesystem) */
#define DOS_CMD_SHUTDOWN                0x5A /* "Z" : place the DOSonCHIP into minimal
power shut down */
#define DOS_CMD_DIR                    0x5C /* "\" : retrieve directory entry */
#define DOS_CMD_MOUNT                  0x5F /* "-" : initialize the card file system */
#define DOS_CMD_WRITE_PREALLOCATE      0x61 /* "a" : add bytes to current file open for
writing */
#define DOS_CMD_CLOSE                  0x63 /* "c" : flush any pending writes to the file,
update the modification time stamp, and clear the handle number */
#define DOS_CMD_DELETE                 0x64 /* "d" : delete the current directory entry */
#define DOS_CMD_DIR_GET_PROPERTY       0x69 /* "i" : get the current directory entry
property (used with DOS_CMD_DIR) */
#define DOS_CMD_SET_DIR                0x6C /* "l" : set the current directory */
#define DOS_CMD_MAKE_DIR               0x6D /* "m" : make a new directory within the
current directory */
#define DOS_CMD_OPEN_WRITE             0x6F /* "o" : open a file for writing */
#define DOS_CMD_OPEN_READ             0x70 /* "p" : open a file for reading */
#define DOS_CMD_READ_DATA              0x72 /* "r" : read data from a file */
#define DOS_CMD_SEEK                   0x73 /* "s" : change the current position pointer
in a file for subsequent reading or writing */
#define DOS_CMD_GET_FREE_SECTORS       0x75 /* "u" : get the amount of available but
unused sectors for the current file system */
#define DOS_CMD_WRITE_DATA             0x77 /* "w" : write data to a file */

/* COMMAND PARAMETER DEFINITIONS */
#define DOS_OFF                        0x00 /* Disable feature */
#define DOS_ON                         0x01 /* Enable feature */
#define DOS_FIRST                      0x00 /* Point to first valid directory entry */
#define DOS_NEXT                       0x01 /* Point to next directory entry */

/* FILE ATTRIBUTES (not mutually exclusive) */
#define DOS_ATTR_READ_ONLY             0x01 /* if set, directory entry is read only */
#define DOS_ATTR_HIDDEN                0x02 /* if set, directory entry is hidden */
#define DOS_ATTR_SYSTEM                0x04 /* directory entry system tag bit */
#define DOS_ATTR_DIRECTORY             0x10 /* if set, directory entry is directory (not a
file) */
#define DOS_ATTR_ARCHIVE               0x20 /* directory entry archive tag bit */

/* DIRECTORY ENTRY PROPERTIES (for DOS_CMD_DIR_GET_PROPERTY) */
#define DOS_PROPERTY_SIZE               0x00 /* return the current entry's size of file */
#define DOS_PROPERTY_TIME_CREATED       0x01 /* return the current entry's creation date/
time stamp */
#define DOS_PROPERTY_TIME_MODIFIED     0x02 /* return the current entry's last modified
date/time stamp */

/* RESPONSE HEADER DEFINITIONS */
#define DOS_RES_NOERROR                0x80 /* no error */

#define DOS_RES_CARD_ERROR              0x81 /* card error (card error in payload) */
#define DOS_RES_CARD_NOT_DETECTED      0x82 /* card not detected */
#define DOS_RES_CARD_INIT_FAILURE      0x83 /* card could not be initialized */
#define DOS_RES_CARD_BLOCK_LENGTH_FAILURE 0x84 /* either card block length is improper or
card could not have block length set */
#define DOS_RES_CARD_VOLTAGE_OUT_OF_RANGE 0x85 /* card required voltage not within allowable
range */
#define DOS_RES_CARD_NOT_MOUNTED       0x86 /* card volume is not mounted */

#define DOS_RES_INVALID_COMMAND        0x90 /* invalid command */
#define DOS_RES_INVALID_PACKET_TYPE    0x91 /* improper packet type than expected */
#define DOS_RES_INVALID_PARAMETER      0x92 /* parameter out of range */
#define DOS_RES_INVALID_OPERATION      0x93 /* cannot perform this operation--not allowed
*/

#define DOS_RES_DISK_FORMAT_INCOMPATIBLE 0x98 /* card file system is not supported (please
reformat card) */
#define DOS_RES_DISK_FULL              0x99 /* disk full error */
#define DOS_RES_DISK_ROOT_DIR_FULL     0x9A /* root directory full error (FAT16) */

```

```
#define DOS_RES_NAME_ERROR                0xA0 /* invalid directory name/not a directory/
filename error */
#define DOS_RES_NAME_NOT_FOUND            0xA1 /* file/directory not found; entry does not
exist in specified dir */
#define DOS_RES_NAME_DUPLICATE            0xA2 /* duplicate name error; file/directory
already exists */

#define DOS_RES_HANDLE_INVALID            0xA8 /* invalid handle/handle out of range */
#define DOS_RES_HANDLE_PREVIOUSLY_ASSIGNED 0xA9 /* handle previously assigned */

#define DOS_RES_FILE_END                   0xB0 /* end-of-file */
#define DOS_RES_FILE_READ_ONLY            0xB1 /* read-only error/write access is denied */
#define DOS_RES_FILE_OPEN_PREVIOUS        0xB2 /* file already open */
#define DOS_RES_FILE_NOT_OPEN_FOR_READ    0xB3 /* not opened for read operation error ;should
this be generic 'cannot apply this operation' ??? */
#define DOS_RES_FILE_NOT_OPEN_FOR_WRITE   0xB4 /* not opened for write operation error */
#define DOS_RES_FILE_CANNOT_DELETE        0xB5 /* open file cannot be deleted error */
#define DOS_RES_FILE_TOO_LARGE           0xB6 /* file would be too large */

#define DOS_RES_DIR_END                   0xC0 /* directory iterator at end */
#define DOS_RES_DIR_NOT_INITIALIZED       0xC1 /* directory iterator is not initialized */

/* Highest allowable error code is 0xDF */

#endif
```