# CCARD MCDISK Card Utility

## General Information

The CCARD utility was created for to be able to access all types of PCMCIA Cards (PC Cards) that can be used with a MCDISK. The MCDISK is a PCMCIA Card Reader/Writer with a SCSI interface that allows systems that have a SCSI Interface to access PCMCIA Cards. With a MCDISK and the CCARD utility, one can format, read, write, compare and test SRAM, Flash, ATA and I/O Cards.
The CCARD utility is a commandline-tool only, no graphical user interface exists.

## Technical Features

- Write, read and verify memory card data
- Very short data transfer time
- Card access via SCSI bus and MCDISK
- FLASH ROM card read / write capability
- Card / device information can be displayed
- Defineable offsets and size in file and card
- Automatic selection of SCSI ID (can be overwritten)
- Card examining with hexdump
- Card format / erase capabilities
- JEDEC ID selectable
- Card test functions

# 1. Table of Contents

## 2. Installation

The CCARD utility may be obtained with or without an install program. The install program simply copies the manual (this file) and the CCARD utility in a separate folder. To use this utility you have either to work in the folder where the CCARD resides or you have to copy the utility to a folder which is included in your PATH variable.
This utility needs the **SCSI ASPI Manager**, which is either a part of your hostadapter software or can be obtained from the homepage of Adaptec: www.adaptec.com
Depending on your Operating System you may need some additional registry settings. Please consult our homepage www.mpl.ch for further information.

## 3. Using the CCARD utility

This utility is a **Commandline program** which means you have to work in a Commandline prompt window. For more information about the parameters consult the “MCDISK Technical Reference Manual” which can be obtained on our homepage www.mpl.ch. Please note, that all following examples are described for Windows Systems, if you are using DOS, type “ccard” instead of “ccard32”.

### 3.1 CCARD Help

With the Command `ccard32 -?` all Command Line Switches are listed with a short description:

```
C:\>ccard32 -?
# Usage:    ccard32 [options] <file>
# Function: data transfer between a file and a memory card with MCDISK
# Options:
#   -c              : display card/device information
#   -s <offs>       : start offset within input file (default=0)
#   -d <offs>       : start offset on card (default=0)
#   -z <size>       : number of bytes to copy (default=32k)
#   -i <ID>         : SCSI ID of MCDISK, (default=look for lowest ID#)
#   -l <LUN>        : SCSI LUN to be used, (default=7, raw access)
#   -n <HAN>        : Host adapter number,(default=0, first)
#   -e              : Erase a portion of the card (use -z and -d options)
#   -t              : translate input file (S01/S09 or Intel Hex format)
#                     Use -f Option for filling pattern (default $FF)
#   -r              : read from card to file
#   -v              : verify card against file
#   -p              : print hex dump of card to standard output
#   -f <byte>       : fill area with <byte>
#   -q              : quiet operation (no progress information)
#   -u <speed>      : Set access speed code ($24 for 150nS, 0=default=250nS)
#   -o              : Set access mode to I/O
#   -w              : Override write protect (USE WITH CARE!)
#   -a              : Use non-counting access (for special I/O accesses)
#   -x              : issue REZERO UNIT before any other operation
#   -format         : format/erase card. Options for formatting are:
#      -m <mode>    : format mode: 0 : disk-like PCMCIA Lvl2 format (default).
#                                    (CIS mode=3, Format mode=0)
#                                  1 : raw format, no Lvl2 CIS in common space.
#                                    (CIS mode=1, Format mode=255)
#      -j <JEDEC>   : JEDEC ID for formatting. Note that JEDEC ID is
#                     normally a hex number, so use $ prefix!
#                     WARNING: USING INCORRECT JEDEC ID MAY DESTROY A CARD!
#      -z <size>    : Set card size (only when MCDISK cannot determine size)
#      -t           : Do card test during format (takes a while)
# Notes:
#   - numbers can be decimal or $hex.
#   - numbers can be k-Bytes M-Bytes or 512-blocks (k,M or b suffix).
#   - card offset and size must be multiple of 512
# V2.0 (c) 1993-2001 by MPL AG, Switzerland
# Product number: MEV-10001-003
```

## 3.2 Display card information

The option –c display information of the inserted card. Please note, that the CCARD utility access takes place on LUN7 which is a raw Device. Normaly this does not make much sense with this option. So use the option –c in conjunction with the option –l 0. This accesses the card in the lower slot. The following print-out shows the card information of a ATA Harddisk:

```
C:\>ccard32 -c -l 0
# Found MPL      MC-DISK-FA 0.3  MEV-10038-010 on SCSI ID# 3.
# Parameters of device addressed trough LUN #0:
#               Device type  =  $E0 (PC-ATA disk)
#               Writability  =  2 (rewritable)
# Write Protect Switch (WP)  =  Not Active for this Device
# Write Protect Switch (WP)  =  OFF (not protected)
#   Battery Status (if any)  =  2 (good)
#           Card's CIS mode  =  1 (LVL 1 CIS only)
#                  JEDEC ID  =  $DF01
#               Access time  =  150.0 nS
#  Physical device capacity  =  $00000800 / 2048 bytes
#   Data partition capacity  =  $1F020000 / 520224768 bytes
#    Card Manufacturer name  =  "Calluna"
#         Product name/info  =  "CT520RM"
#            Product info 1  =  "20Aug98"
#            Product info 2  =  "'546-10 "
#     Format program vendor  =  "MPL AG"
#        Information string  =  "'546-10 Calluna Technolo"
#          Data organsiation =  $00 (File System)
#     Data organisation name =  ""
```

Please refer to the “MCDISK Technical Reference Manual” for the description of the parameters.

## 3.3 Start offset within input file

The option –s <offs> specifies the byte offset with which the input file will be read. For example:

```
ccard32 –l 0 –s $200 test.bin
```

writes the data from the address $200 in the file to the card. This means, that the data on the address $200 of the file is written to the card at address 0.
Notes:

- The offset can be decimal, hexadecimal, numbers of blocks, kBytes, or MBytes. Refer to chapter 4 Format of numbers for further information.
- The default offset is 0.

## 3.4 Start offset on card

The option –d <offs> specifies the byte offset on the card. This option works an read os well as on writes. For example:

```
ccard32 –l 0 –d $200 test.bin
```

writes the contens of the file test.bin to the card at the address $200. This means, that the data on the address 0 of the file is written to the card at address $200.
Notes:

- The offset can be decimal, hexadecimal, numbers of blocks, kBytes, or MBytes. Refer to chapter 4 Format of numbers for further information.
- The offset must be multiple of 512
- The default offset is 0.

## 3.5 Number of bytes to copy

The option –z <size> specifies the size of the datatransfer in bytes. This option works an read and on writes. For example:

```
ccard32 –l 0 –z 1k test.bin
```

writes 1kByte of the file test.bin to the card.
Notes:

- The size can be decimal, hexadecimal, numbers of blocks, kBytes, or MBytes. Refer to chapter 4 Format of numbers for further information.
- The size must be multiple of 512
- The default size is 32kByte.

## 3.6 SCSI ID of the MCDISK

The option –i <ID> specifies SCSI ID of the MCDISK. Normaly the CCARD scans the SCSI bus for an MCDISK, and uses the first MCDISK which is found. This option can be used if more than 1 MCDISK is connected to the SCSI bus.

## 3.7 SCSI LUN to be used

The option –l <LUN> selects Logical Unit Number to be used. The MCDISK distinguish between different access modes on the same card:

- **Transparent mode:**
  For an application, where some type of memory card is used as mass storage for a computer system, the MCDISK provides transparent access to the card. This means that the SCSI host machine may access the card as if it was a removable hard disk device. In this transparent mode of operation, the MCDISK handles all issues that are specific to PCMCIA/JEIDA memory cards (such as the Card Information Structure, CIS) internally and invisible to the SCSI host system.
- **Direct mode:**
  There is a different type of application for the MCDISK, where the transparent access mode is not suitable: When the physical accesses done to the memory card need to be tightly controlled by the application. This is the case with special I/O or controller cards that perform user specific functions. In transparent mode, the MCDISK makes a lot of extra accesses to the card to analyse the CIS, find out the card size etc. These accesses might interfere with the user's intention. Therefore the MCDISK provides a direct mode of operation.

The LUNs are assigned to the slots and devices as follows:

| LUN | Access |
|---|---|
| 0 | Lower slot transparent access |
| 1 | Lower slot transparent access 2nd Device |
| 2 | Upper slot transparent access 2nd Device |
| 3 | Upper slot transparent access |
| 4 | Upper slot attribute space direct access |
| 5 | Upper slot common space direct access |
| 6 | Lower slot attribute space direct access |
| 7 | Lower slot common space direct access |

**Exampels:**

- To read the first 10 blocks of a ATA-Harddisk inserted in the lower slot:
  `CCARD32 –l 0 –z 10b –r test.bin`
- To read the CIS of a ATA-Harddisk inserted in the lower slot:
  `CCARD32 –l 6 –z 1b –r cis.bin`
- To write a flash card inserted in the upper slot:
  `CCARD32 –l 3 test.bin`
- To read a flash card inserted in the lower slot:
  `CCARD32 –l 7 –r test.bin`

Please refer to the “MCDISK Technical Reference Manual” for further information of the LUN handling.

## 3.8 Host adapter number

The option –n <HAN> selects the host adapter number of the SCSI adapter which the MCDISK is connected. Please note, that on some systems the host adapter number of the first SCSI adapter is 1 because the IDE CD-ROM (ATAPI) uses parts of the SCSI driver and the IDE Interface becomes the SCSI host adapter number 0.

## 3.9 Erase a portion of the card

The option –e erases a portion of a flash card. This works only with a block erase flash card. The block number to be erased is addressed with its byte address with the –d option. The number of blocks to be erased has to be selected with the –z option. If only one block has to be erased, the –z option must contain the size of the block. To erase the entire card, use the –format option.

## 3.10 Translate input file

The option –t selects the translate mode. The input file can be in an Intel Hex or a Motorola S-Record format. To limit the data to be written (some linker output file contains also the initialized RAM contents) use the –z option. If the address has to be shifted, the –d and –s options can be used.
All data which are not specified by the Intel Hex or Motorola S-Record format are filled with 0xff. If another filling pattern is needed, use the option –f.

## 3.11 Read from card to file

The option –r switches in read mode. The data are now transfered from the card to the file. To manipulate the size and the start on the card of the data, use the –z and –d options.

## 3.12 Verify card against file

The option –v selects the verify mode. The data from the file will be compared with the data of the card.

## 3.13 Print hex dump of card to standard output

The option –p read from the card and display the data as hex dump on the screen. Since the default size is 32kByte you like probably to limit the amount of the data with the -z option. Please use the –d option to set the start address of the displayed data on the card. Following example shows the hex dump of the first 512Bytes of an ATA Harddisk:

```
C:\>ccard32 -l 0 -p -z 1b
# Found MPL      MC-DISK-FA 0.3  MEV-10038-010 on SCSI ID# 3.
Addr      00 01 02 03 04 05 06 07  08 09 0A 0B 0C 0D 0E 0F  0123456789ABCDEF
--------  -----------------------------------------------  ----------------
00000000: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000010: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000020: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000030: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000040: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000050: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000060: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000070: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000080: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000090: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000C0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000100: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000110: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000120: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000130: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000140: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000150: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000160: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000170: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000180: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000190: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001A0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001B0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 80 01  ................
000001C0: 01 00 06 00 C1 EF 3F 00  00 00 D1 7C 0F 00 00 00  ......?....|....
000001D0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001E0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001F0: 00 00 00 00 00 00 00 00  00 00 00 00 00 00 55 AA  ..............U.
# Read 512 ($200) bytes, checksum=$04D2
```

## 3.14 Fill area with <byte>

With the option –f the card will be filled with the specified value. Please note that the value is most probably in the hexadecimal format, so use the $ prefix for the value. The –d and –z options can be used to manipulate the start and size of the filling area. The –f option can also be used to specify the filling pattern for Intel hex and Motorola S-Record formats (default $FF).

## 3.15 Quiet operation

The option –q surpress all progress information. This option may be usefull, if using the CCARD utility in a batch file.

## 3.16 Set access speed code

The –u option is used to set the access speed code. The MCDISK uses the correct access speed for all transparent LUNs, by analyzing the CIS. Use –u $24 for 150ns or –u 0 for 250ns (default).

## 3.17 Set access mode to I/O

The –o option selects the access takes place in the I/O area instead of the common memory area.

## 3.18 Override write protect

The –w option overwrites the write protection (if existing) of the card. Use this option with care.

## 3.19 Use non-counting access

The –a option selects the non-counting mode. The adresses are not incremented for every access. This option may be useful for special cards in I/O mode.

## 3.20 Issue REZERO UNIT before any other operation

The –x option issue a REZERO_UNIT SCSI command before any other operation. For PC-ATA cards, this command issues a RECALIBRATE command to the ATA drive, which causes the R/W heads to be restored to some initial position for mechanical drives. For all cards, REZERO UNIT will perform a PCMCIA hardware RESET and causes the card to be re-analysed (CIS reading, size testing, ATA controller initialisation), if it is accessed in transparent mode. In direct mode, the card will just be RESET.

## 3.21 Format/erase card

The –format option formats or erases the card.
For PC-ATA cards a FORMAT_TRACK will be issued (Low-Level Format). This causes some PC-ATA cards to check for bad blockes and move any bad blocks to a certain (hidden) area. Use the –m 1 option if issue this command for a PC-ATA card.
Flash cards will be completely erased (all bytes changed to $FF) with this command.
Please refer to the following chapters for formatting options.

### 3.21.1 Format mode

The –m option specifies the formatting mode. Possible values are:

**-m 0:** Disk-like PCMCIA Level2 format (default). After formatting, a CIS with the device information will be written to the card.

**-m 1:** Raw format, no Level2 CIS in common space. This mode is the most used mode.

This option can only be used in conjunction with the –format option.
Please refer to the “MCDISK Technical Reference Manual” for further information formatting.

### 3.21.2 JEDEC ID for formatting

The –j option selects the JEDEC ID for formatting. This is only useful, if the MCDISK does not find out the JEDEC ID of the card itselfes. Mostly used with flash cards.
Please note that JEDEC ID is normally a hex number, so use $ prefix.
This option can only be used in conjunction with the –format option.

### 3.21.3 Set card size

The –z option sets the size of the formatting. This is only used, if the MCDISK cannot determine the size.

### 3.21.4 Do card test during format

The –t option enables destructive card test during format. This option makes no sense with flash cards, since they are not random writable.

# 4. Format of numbers

Depending on the format, the numbers can be decimal, hexadecimal, numbers of blocks, kBytes, or MBytes:

| Mode | Format | Exsample (size) | |
|---|---|---|---|
| Decimal | number | -z 1024 | 1024Bytes or 1kByte |
| Hexadecimal | **$**number | -z $1000 | 4096Bytes or 4kByte |
| number of blocks | number**b** | -z 10b | 5120Bytes or 5kByte |
| kBytes | number**k** | -z 10k | 10240Bytes or 10kBytes |
| MBytes | number**M** | -z 4M | 4194304Bytes or 4MBytes |

# 5. CCARD examples:

## 5.1 SRAM card

- **SRAM Read**
  Reading 512k starting from 128k (128k..640k) from SRAM card in lower slot:
  ```
  C:\>ccard32 -r test.bin -z 512k -d 128k
  ```
  or:
  ```
  C:\>ccard32 -l 0 -r test.bin -z 512k -d 128k
  ```
- **SRAM Write binary**
  Writting 64k from a binary file starting at address 1024 to a SRAM card starting at address 128k in lower slot. (Data from 1024 to 65k in the binary file will be transfered at address 128k to 192k from SRAM):
  ```
  C:\>ccard32 test.bin -z 64k -d 128k -s 1k
  ```
  or:
  ```
  C:\>ccard32 -l 0 test.bin -z 64k -d 128k -s 1k
  ```
- **SRAM Verify binary**
  Verifying 1MByte binary file with the 1MByte written to the SRAM card:
  ```
  C:\>ccard32 test.bin -z 1M -v
  ```
  or:
  ```
  C:\>ccard32 -l 0 test.bin -z 1M -v
  ```
- **SRAM Write Intel hex or Motorola S-Record**
  Writting from 32k to 128k from a Intel hex/S-Record file to a SRAM card starting at address 128k in lower slot:
  ```
  C:\>ccard32 test.a20 -z 96k -d 128k -s 32k -t
  ```
  or:
  ```
  C:\>ccard32 -l 0 test.a20 -z 96k -d 128k -s 32k -t
  ```
- **SRAM Verify Intel hex or Motorola S-Record**
  Verifying 128kByte Intel hex/S-Record file with the 128kByte written to the SRAM card:
  ```
  C:\>ccard32 test.a20 -z 128k -v -t
  ```
  or:
  ```
  C:\>ccard32 -l 0 test.a20 -z 128k -v -t
  ```

## 5.2 Flash card

- **Flash card erase**
  Erasing the entire Flash card and writes no CIS on it:
  ```
  C:\>ccard32 -l 0 -format -m 1
  ```
- **Flash card block erase**
  Erasing the 2nd and the 3rd block of an Intel Series 2 Flash card (128kByte erase block size):
  ```
  C:\>ccard32 -l 0 -e -d 128k -z 256k
  ```
- **Flash card write binary**
  Write a 64kByte binary file to the flash card:
  ```
  C:\>ccard32 -l 0 test.bin -z 64k
  ```
- **Flash card write Intel hex or Motorola S-Record**
  Write a 4MByte Intel hex/S-Record file to the flash card:
  ```
  C:\>ccard32 -l 0 test.a20 -t -z 4M
  ```

- **Flash card verify**
  Verify a 1MByte binary file with the flash card contents:
  ```
  C:\>ccard32 -l 0 test.bin -z 1M -v
  ```
  Verify a 1MByte Intel hex/S-Record file with the flash card contents:
  ```
  C:\>ccard32 -l 0 test.a20 -t -z 1M
  ```

### 5.3 PC-ATA card

- **PC-ATA card low-level format**
  Low level formatting of a PC-ATA disk:
  ```
  C:\>ccard32 -l 0 -format -m 1
  ```
- **PC-ATA card reading the CIS**
  Reading the CIS of a PC-ATA card and store it in the file CIS.BIN:
  ```
  C:\>ccard32 -l 6 -r cis.bin -z 2b
  ```
- **PC-ATA card reading the 1st block**
  Reading the 1st of a PC-ATA card and store it in the file MBR.BIN:
  ```
  C:\>ccard32 -l 0 -r mbr.bin -z 1b
  ```

## 6. Notes on direct LUN support

Unfortunaly the direct LUN support on the Microsoft operating systems are not the same on every operating system:

| OS | Support |
|---|---|
| DOS | All LUNs are accessible, but only LUN0 will be mounted as removable drive. PC-ATA disks or High-Level formatted SRAM cards are only accessible in the lower slot. |
| Windows9x | Only LUN0 is accessible. PC-ATA disk or High-Level formatted SRAM cards are only accessible in the lower slot. |
| WindowsNT4.0 | All LUNs are accessible, all transparent LUNs will be mounted as removable drive. PC-ATA disks or High-Level formatted SRAM cards are accessible in the lower or upper slot. |

Note: WindowsNT4.0 with SP3 installed, does not allow to access the direct LUNs.

This page is left blank intentionally

## Copyright and Revision History



This manual reflects the Version 2.0 of the CCARD program.

Publication Date : Feb 2001

## Disclaimer and Licence Agreement



## Support

Support can be contacted by sending an e-mail to support@mpl.ch