M-Systems
Flash Disk Pioneers

DiskOnChip Software Utilities for TrueFFS 6.3

User Manual, Rev. 1.3

December 2005
02-UM-0505-40

TABLE OF CONTENTS

1. INTRODUCTION

This manual describes the following DiskOnChip utilities for Windows 2000/XP and DOS that may be used with M-Systems DiskOnChip products:

- DFORMAT
- DINFO
- DIMAGE
- DFS
- SPLITIMAGE

Note: TrueFFS 6.3.X supports only SAFTL-formatted DiskOnChip devices. TrueFFS 6.3.X utilities (DINFO, DFORMAT, DFS, and DIMAGE) support only SAFTL-formatted DiskOnChip devices.

A description of the utility flags is provided, including specific examples and basic instructions to assist in easy and quick installation of DiskOnChip over the target platform.

This manual is intended for system integrators who are familiar with the PC environment and the operating system in use. It is also recommended to read the relevant DiskOnChip data sheets and installation instructions for your specific operating system.

The latest versions of the DiskOnChip utilities can be downloaded from M-Systems' website at www.m-systems.com.

1.1. DiskOnChip G3/P3, G3/P3 LP, G4 and H1

M-Systems' DiskOnChip products are high-performance, single-chip flash disks that provide full hard-drive emulation for all major operating systems (OSs) and platforms. This unique product line offers a complete boot and data storage solution for applications, such as embedded systems, feature phones, smartphones and PDAs. DiskOnChip provides superior performance through a true 16-bit bus interface, X2 technology and offers security-enabling and data-protection features.

A TrueFFS driver is required to work with all DiskOnChip products. TrueFFS is natively supported by every major OS, such as Windows Mobile, Symbian, Palm, and Linux. For other environments (including OS-less) the TrueFFS Software Development Kit (SDK) can be obtained. When using DiskOnChip as the boot device, M-Systems' Boot Software Development Kit (BDK) package is required. Contact M-Systems regarding availability for both packages.

1.2. Terms and Abbreviations

Table 1: Glossary of Terms and Abbreviations

Interface Term	Definition
Socket	The physical location where a DiskOnChip device can reside. The maximum number of sockets is defined by the **SOCKETS** definition in **FLCUSTOM.H**.
Physical Drive/Physical Device	DiskOnChip device inserted in a socket. The number of physical drives depends on the number of sockets registered by the socket components in the TrueFFS application. Physical drives are numbered from zero, and serve as the four least significant bits (LSB 0-3) of the drive handle (see below). See also *TrueFFS Sockets and Drives*.
Partition/Volume	A partition is part of a physical drive handled as an independent unit. A partition can be either a disk partition (logical drive partition) or a binary partition (see page 5). A physical device can contain up to four partitions of any type, provided at least one of them is a disk partition.
Disk Partition (Logical Drive/BDTL Partition/ BDTL Volume) **Note**: The terms disk partition/BDTL volume/BDTL partition are used in parallel for the same independent area on the flash. Binary partition/ BDK partition/binary volume are also interchangeable.	A disk or a BDTL (Block Device Translation Layers) partition is a partition formatted and supported by a TrueFFS Translation Layer (such as SAFTL, described below), making it capable of supporting a block device driver and file system. Most operations on the disk partition require a mount operation on the partition, using one of the API functions provided for that purpose. This enables the TrueFFS driver to acknowledge the presence of the partition, check the validity of its BDTL and/or FAT data formats, and initialize its supporting data structures in memory. The disk partitions are numbered starting from zero.
EP (Enhanced Performance Partition)	Disk partition which is formatted to be an enhanced performance partition. An enhanced performance disk partition comprises two types of space – Fast and Normal. Fast space always found at the beginning of the disk partition. The fast area usually takes twice the space of the normal area.
Floor	In a cascaded DiskOnChip configuration, each DiskOnChip device is called a floor
Binary Partition/Binary Volume	A binary partition is not supported by the TrueFFS block device translation layer (BDTL), but can be used as a direct-access storage area. The binary partitions are numbered starting from zero, independently of the disk partitions co-existing on the same physical drive.

Interface Term	Definition
Binary Sub-Partition	All of the binary partition blocks are marked with a unique signature. A dedicated routine enables changing this signature for a contiguous subset of the partition's blocks, creating several separated areas within the binary partition. Each area is called a binary sub-partition. When first formatted, the binary partition contains a single sub-partition.
SPL Partition	A simplified binary partition that does not use the matching algorithm, making it ideal for storing the SPL code. The simpler logic simplifies the IPL code required for reading the SPL, and allows it to fit on the DiskOnChip XIP block.
SAFTL Formatted Device	Sequential Access Flash Translation Layer (SAFTL) is the flash management algorithm used by TrueFFS to manage DiskOnChip G3/P3, DiskOnChip G3/P3 LP, DiskOnChip G4 and DiskOnChip H1.

1.3. General Rules for all Utilities

Maximum number of each command line parameter type that the utilities may accept is 99. No more parameters can be accepted.

2. DOCSHELL UTILITY

This utility provides the shell for all other utilities described in this document.

The **DOCSHELL** utility has two operation modes – Menu mode and command line mode.

2.1. Command Line Mode

This mode allows running a single utility, included in **DOCSHELL**. In this case the **DOCSHELL** must be run with the first flag, defined as utility name.

For example:

1. To show DiskOnChip binary, disk partition and IPL information:

   ```
   DOCSHELL /dinfo /bdk /bdtl /ipl
   ```

2. To read content of the binary partition number 1 into file bdk1.bin:

   ```
   DOCSHELL /dformat /win:d0000 /y /read /bdkf1:bdk1.bin
   ```

Table 2: DOCSHELL Flags

DFORMAT Option	Description
/DFORMAT	Run DFORMAT utility
/DINFO	Run DINFO utility
/DIMAGE	Run DIMAGE utility
/DFS	Run DFS utility
/SplitImage	Run SplitImage utility

2.2. Menu Mode

To use the **DOCSHELL** in menu mode, the user must run the executable without parameters. The menu in Figure 1 will appear on the screen.

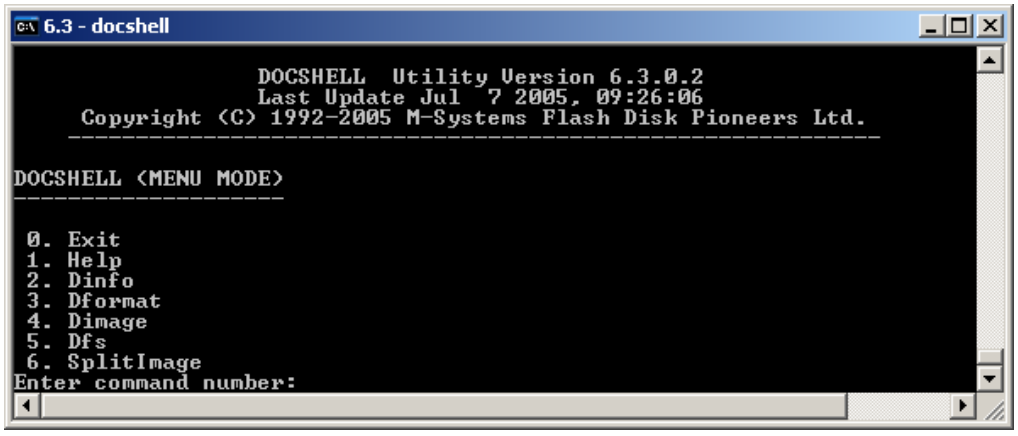

Figure 1: DOCSHELL Menu

The number that appears to the left side of the command (utility) name should be typed followed by the enter key. This will run the requested utility and prompt the user for relevant parameters.

After the specified command (utility) is completed, the menu is displayed again and the user may choose another command to execute or exit the utility.

The Exit and the Help commands are not utilities. The Exit command will close **DOCSHELL**, and the Help command will display relevant help content, as shown in Figure 2.

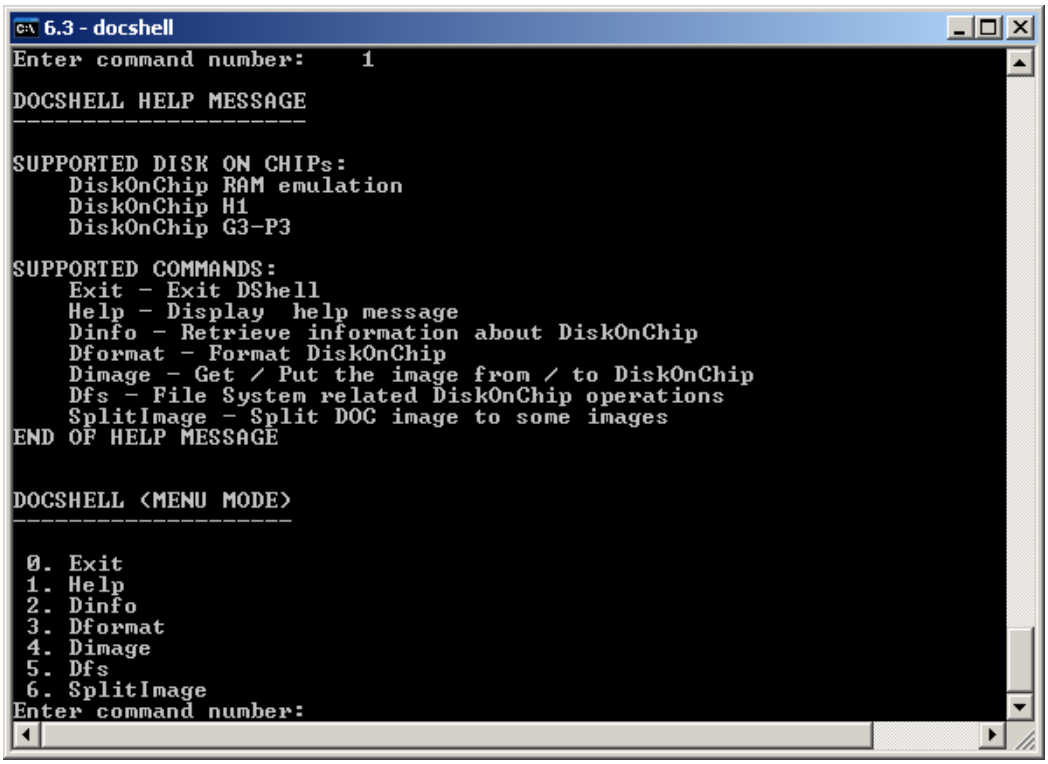

Figure 2: DOCSHELL Help

Note: menu mode accepts no more than 255 characters command lines.

2.3. How to Extend DOCSHELL

Systems with a source code package, may utilize **DOCSHELL** to run command, by extension **DOCSHELL**.

DOCSHELL utility may be extended to be able to run another commands (utilities).

2.3.1. To Extend DOCSHELL

- In a separate file, define the function that should be from type
  `typedef ExStatus EXAPI (*pCommandFunction)(void);`
- Add the declaration of this function to docshell.c file
- Add to function RegisterTools() from docshell.c the line as following:

```
CheckStatus( RegTool("<command name>", "<short help message>",
<function name, defined in 2.3.1>, <parameters types>));
```

Where <parameters types> may be as following:

- `DSHELL_STOP_AFTER_CMD` - exit DOCSSHELL after this command,
- `DSHELL_PARAMETERS` - request parameters from user and parse them,
- `0` (zero) - means do not request parameters and do no exit DOCSHELL.

3. DFORMAT UTILITY

Before the TrueFFS driver can access the DiskOnChip device, the device must be formatted (similar to a floppy disk). Formatting the device initializes the flash media and creates the binary and disk partitions.

DiskOnChip can be formatted more than once. However, all stored data on the device is erased during the formatting process.

Note: When DiskOnChip is reformatted, the binary partitions are retained by default.

Appropriate versions of the DFORMAT utility are required for formatting. If the versions do not match, the formatting procedure stops and DFORMAT returns an error message. The following sections describe how to use the DFORMAT utility and provide a description of its flags.

DFORMAT functionality has changed compare to previous DFROMAT releases:

- There is no file system formatting done by DFORMAT. This functionality has been moved to the DFS utility. Since this change was made, some corresponding flags have been removed.
- Read operation is now supported by DFORMAT. Since this changes some flags and enhances functionality, the **/READ** flag was added.

3.1. DFORMAT Syntax

3.1.1. Operation Modes and Command Line Flags

In TrueFFS 6.3.X, the DFORMAT utility may be used for the following purposes:

- Format the entire DiskOnChip device and program it. This is default operation mode for DFORMAT.
- Format (update) single Binary or Disk partition. In this case DFORMAT should be used with the **/NOFORMAT** flag. The size of the partition MUST remain the same.
- Read the IPL data or partition data located on one or more Disk or Binary partitions. In this case **/READ** flag MUST be used.
- Return DiskOnChip to virgin state using the **/UNFORMAT** flag.

Note: Many flags have been changed and/or enhanced as a result of extending DFORMAT operation modes.

3.1.2. Common Syntax

DFORMAT syntax is:

```
DFORMAT <[/WIN:address][/WINL:low_address /WINL:high_address]>
[/Flag[n]:parameter[suffix]]
```

Where:

`WIN (WINL WINH)`	Memory address (or address window) where DiskOnChip resides
`Flag`	See full flag list and corresponding descriptions below
`N:`	Partition number (n = 0 - 3). Default value is 0.
`Parameter`	See full flag list and descriptions below
`Suffix`	Size units. Can be either `M` for *0x100000, or `K` for *0x400

Example:

```
DFORMAT /win:D0000 /BDKL0:1M
```

Formats the DiskOnChip device located in memory address D0000, with a 1MB binary partition. The rest of the memory is formatted, by default, as a disk partition.

3.2. Using DFORMAT Flags

The following sections define the various flags used with the DFORMAT utility.

3.2.1. Common Flags

Table 3: Most Commonly used DFORMAT Flags

DFORMAT Option	Description
/WIN:address	Memory address at which DiskOnChip is located. The address should be specified in hexadecimal format (for example, **/WIN**:D0000).
/NOFORMAT	Use this flag to update the DiskOnChip with the IPL, binary or Disk partition(s) without reformatting the entire device. When **/READ** flag used **/NOFORMAT** not required.
/Y	Instructs the system not to pause for confirmation before beginning to format.
/? & /H	Shows the full Help screen.
/READ	Causes DFORMAT to perform a read operation. Conjunction with NOFORMAT is not required.
/CLFILE:<filename>	Get command line parameters from file. This flags must be a single flag, get from command line.

3.2.2. Binary Partition Flags

Table 4: Binary Partition Flags

DFORMAT Option	Description
/BDKF[n]: Boot Image File	Places the boot image file on the binary partition [n]. Up to three binary partitions can be defined for SAFTL-formatted DiskOnChip devices. Note: When the length of the file is smaller than the unit size, the remainder of the unit is filled with zeros. When used in conjunction with the **/READ** flag, **/BDKF** specifies the name of the file and holds the data that will be read from binary partition number 'n'. The length of the file will be aligned to the unit size. The extension will be filled with zeros. In case of an empty binary partition the file created will be of size zero.
/BDKN[n]: BDK Partition Signature [Default value = BIPO]	Used in FORMAT mode and specifies the binary partition [n] signature (4 characters). Used in READ and NOFORMAT (update) mode and specifies the binary sub partition signature that will be read into file or written into file, specified by /BDKF.

DFORMAT Option	Description
/BDKL[n]: BDK Partition Size	Defines the size of the binary partition [n]. Useful for reserving space in the binary partition (over the size of the binary program placed using the **/BDKF** flag) for later upgrades. Up to 3 binary partitions can be defined on DiskOnChip. Note: When the specified partition size is less than 1 unit, DFORMAT rounds it up to a full unit.
/LEAVE:k	Leaves the content of the first [k] binary partitions. Cannot be used in conjunction with /NOFORMAT, /UNFORMAT and /READ flags. /LEAVE:0 – Used in format mode will delete all binary partitions from the DiskOnChip device. By default, binary partitions are not deleted.
/BDKSPL[n]	Defines binary partition number [n] as an SPL partition.
/BDKL[n]:Size;[Sign0Size0[;...;[Sign6Size6]]]	When a binary sub-partition is defined using the **/BDKL** flag, the sub-partition size should be specified after the sub-partition sign: This option creates a sub-partition with signature Sign# and size Size#. For example, **/BDKL0:5m;sig12m;sig23m** creates a binary partition with 2 sub-partitions (sig1,sig2) whose sizes are 2m and 3m respectively. Up to 7 binary sub partitions may be defined by BDKL command. When some space reminds at Binary partitions after all required by user sub partitions created, this space will set to another sub partition with default signature. Note: '+' character may be used instead of ';' . (Linux for example)
/OTWBDK[n]	Places OTW protection on BDK partition 'n'. OTW protection may be placed only on the partitions, which have read_protected or protectable changeable protection flag ON. The OTW feature is not implemented on H1 devices.

Note: When the binary partition number is not specified, the default partition number is 0.

3.2.3. Disk Partition Flags

Table 5: DFORMAT Flag Options for Disk Partitions

DFORMAT Option	Description
`/BDTLL[n]:Partition Size`	Sets the size of the disk partition [n]. The size of the last disk partition does not have to be defined. For example, `/BDTLL0:1MB` creates two partitions, the first 1MB (n = 0) in size, and the second (n = 1) the remaining size of the flash disk. Up to 6 partitions (including binary partitions) can be defined on DiskOnChip. Note: When the specified size is less than 1 unit, DFORMAT fails with a wrong parameter error. By default (EP flag not specified for current disk partition) it will be formatted as 99% Normal, 1%Fast)
`/BDTLF[n]:<[Disk Partition Image file]|[!]>`	In update or format modes - Places a disk partition image file on the disk partition [n]. When used in conjunction with the **`/READ`** flag, **`/BDTLF`** specifies the name of the file and holds the data that will be read from disk partition number 'n'. The output file holds all disk partition data, including empty sectors, which will appear as zeros in the file. When '!' is specified – Deletes all the data from corresponding partition. Other partitions will stay in tact.
`/OTWBDTL[n]`	Places OTW protection on BDK partition 'n'. OTW protection may be placed only on the partitions, which have read_protected or protectable changeable protection flag ON. The OTW feature is not implemented on H1 devices.

Note: When the partition number is not specified, the default number is 0.

3.2.4. Protection Flags

Table 6: DFORMAT Protection Flags

`/BDKP[n]:RWCL:Password` [Default value = no protection]	Sets the protection type (**Read**/**Write**/**Change**/**Lock**) and the protection key (password) of the binary partition [n] (n = 0-2): **R**ead: Read-only mode **W**rite: Write-only mode **C**hange: Enable changing the protection type (**R**/**W** or both). **L**ock: Defines whether the LOCK# signal overrides the password.
`/BDTLP[n]:RWCL: Password` [Default value = no protection]	Sets the protection type (**Read**/**Write**/**Change**/**Lock**) and the protection key (password) of the disk partition [n] (n = 0-5).
`/BDKZ[n]: Password`	Inserts the key of the binary partition number [n] (n = 0-2, default is 0). Useful when formatting devices with previously defined protected partitions.
`/BDTLZ[n]: Password`	Inserts the key of the disk partition number [n] (n=0-5, default is 0) Useful when formatting devices with previously defined protected

	partitions.

Notes: 1. Note1: When the partition number is not specified, the default number is 0.

2. Note2: Password length is 8 bytes exactly.

3.2.5. Advanced Operation Flags

Table 7: Advanced Operation Flag Options

DFORMAT Option	Description
/IPL:File | !	Useful in systems where DiskOnChip is used as the boot device. **file** writes a custom IPL (up to 2K bytes in DiskOnChip G3/P3, G3/P3 LP, G4). When used in conjunction with the **/READ** flag, **/IPL** specifies the name of the file and holds the IPL data that will be read from the DiskOnChip device. Reading IPL from an unformatted device will create an empty file. **!** erases the IPL. IPL file size should not exceed IPL size. Up to 1K bytes in DiskOnChip H1 512MB and 1GB, 2K bytes in DiskOnChip H1 2GB). When booting DiskOnChip G3/P3 LP and G4 in Paged Ram, up to 15 Kbytes are available in G3/P3 LP and up to 246 Kbytes in G4.
/OTP:<file name>	Write/Read OTP to/from the file. G3/P3, G3/P3 LP and H1 devices – 6 Kbytes per page.
/UNFORMAT	Removes any existing DiskOnChip formatting and restores DiskOnChip to its virgin state. This flag is used, for example, when there are two cascaded DiskOnChip G3 devices and another device must be added. Regular DFORMAT on the extra device will not be effective in building the shared BBT. The **/UNFORMAT** flag must be run to integrate the three devices as one unit.
/WINL:Address	Determines the lower memory limit of the window in which the DiskOnChip device will be searched. If the **/WIN** option was used, this option is ignored.
/WINH:Address	Determines the higher memory limit of the window in which the DiskOnChip device will be searched. If the **/WIN** option was used, this option is ignored.
/FAST	Causes TrueFFS to use the multiple byte/word read and write routines in the access layer.
/SHIFT	Defines the address shift
/WIDTH	Defines the address width
/IPLMODE:<p|v>[:at|ab]	Write IPL in the customized mode (supported only by

DFORMAT Option	Description
	DiskOnChip G3/P3 LP and DiskOnChip G4 products): P – for paged RAM boot mode; V – for virtual RAM boot mode AT – for alternate TOP; AB – for alternate bottom The default mode is paged RAM boot mode, alternate top mode.
/RAMDEVFILE, /RAMDEVTYPE	For more information on **RAM MTD** usage please refer to section 8.

3.2.6. Removed Flags

The flags listed in Table 8 were removed from the DFORMAT utility. The functionality of the flags, listed here and related to the file system, is now handled by the DFS utility.

Table 8: Removed Flag Options

DFORMAT Option	Description
/LABEL[n]: Label [Default value = NULL]	The string used as the DOS volume label of the formatted partition [n]. Up to 11 bytes long.
/NODOS[n]	Use this flag to prevent creating a DOS FAT file system while formatting partition [n]. Only low-level formatting is performed. Useful for systems with file systems other then FAT. Up to 4 bytes long.
/DOSID[n]: ID	FAT partition [n] identification number (ID).
/FAT[n]: Number	Number of FAT copies on partition [n], where n has a value of 1 or 2. The default value is 2.
/OLD_FORMAT[n]	Format disk partition [n] with one sector per cluster.
/BDTLBINFS	Formats the disk partition with BinFS.
/LOG:file	Copies the Bad-Block Table (BBT), stored on DiskOnChip, to a file. When this flag is used, DiskOnChip is also reformatted. If you want to preserve data on DiskOnChip and only read the BBT, use the **/NOFORMAT** flag. Note: The BBT returned by this flag also contains several blocks that are for TrueFFS internal use. Good blocks never appear in the file. Dynamic Bad Blocks also do not appear in the file.
/S: ! | * [Default value = *]	*****: Erase the disk partitions, but leave the content of the binary partitions intact. **!**: Erase the contents of the firmware binary sub partition from DiskOnChip.

3.3. DFORMAT Usage Examples

In the following examples, the DiskOnChip device is located at address 0xd0000.

Example 1

```
DFORMAT /WIN:D0000 /IPL:IPL.bin /BDKSPL0 /BDKF0:BootLoader.bin
/BDTLF0:OSimage.bin
```

This formats DiskOnChip with IPL, one SPL partition and two disk partitions. One binary partition is used for storing the boot loader. Two disk partitions are used, one for OS image, and the other as a user partition that occupies the reminder of the media.

Example 2

```
DFORMAT /WIN:D0000 /BDKL0:1M /BDKSPL0 /BDKL1:2M
```

This formats DiskOnChip with two binary partitions and one disk partition. One binary partition, 1MB in size, is an SPL partition for the SPL code, and the other is a normal binary partition 2MB in size. The disk partition occupies the reminder of the media.

Example 3

```
DFORMAT /WIN:D0000 /NOFORMAT /BDTLF:disk0.img
```

This updates disk partition 0 with data from file disk0.img. The reminder of DiskOnChip stay in tact after this command.

Example 4

```
DFORMAT /WIN:D0000 /BDKLZ2:xxxxxxxx
```

This reformats DiskOnChip with a protected disk partition. If the password provided in the command line is not correct, the format operation fails.

Example 5

```
DFORMAT /WIN:D0000 /BDTLL0:12M /BDTLP1:WC:xxxxxxxx
```

This formats DiskOnChip with two disk partitions. The second partition can be write protected and changed (the protection can be switched later, to either read protection or read/write protection).

Example 6

```
DFORMAT /WIN:D0000 /READ /BDTLF:part0.bin
```

This reads the DiskOnChip disk partition 0 to file part0.bin.

4. DINFO UTILITY

The DINFO utility displays DiskOnChip information, such as physical size and partition number. Available options include printing/displaying specific information, not just general information. If no options are selected, DINFO displays general information.

4.1. DINFO Syntax

The DINFO syntax is:

```
Dinfo [/WIN:Address] [/FLAG:parameter]
```

Where:

`/WIN`: Memory window address (if no input is specified, DINFO searches for all connected DiskOnChip devices and displays their information).

`/FLAG`: See flag list in Table 9 for details.

4.2. Using DINFO Flags

Table 9 defines the various flags used with the DINFO utility.

Table 9: Typical DINFO Flags

DINFO Option	Description
`/WIN:Address`	Memory address where DiskOnChip is located. The address should be specified in hexadecimal format (for example, **`/WIN:D0000`**). If the `/WIN` option is not specified, the utility searches for DiskOnChip devices between default addresses. For example, when running the utilities with M-Systems EVBs for x86-based platforms, the default addresses are 0xC8000 – 0xE0000. All devices that are located will be related.
`/WINL:Address`	Determines the lower memory limit of the window where DINFO will search for the DiskOnChip device. If the **`/WIN`** option was used, this option is ignored.
`/WINH:Address`	Determines the upper memory limit of the window, where DINFO will search for the DiskOnChip device. If the **`/WIN`** option was used, this option is ignored.
`/BBT`	Displays Bad Block Table (BBT) information: Number of bad units, list of bad blocks, and the percentage of bad blocks. When working with non-formatted DiskOnChip devices – Prints the simple matched units for the entire DiskOnChip device; otherwise prints units as matched for current DiskOnChip device configuration.
`/OTP`	Displays OTP and device ID information: • OTP Size field: Refers to the maximum capacity available • OTP Used Size: Actual size of the customer OTP • Lock Status: Indicates whether the customer OTP is locked

DINFO Option	Description
	• Unique ID.
/BDTL	Displays the disk partition information for the selected DiskOnChip device, including the number of disk partitions on the media, capacity data, format type and protection status for each individual partition. Quick Mount status will be displayed as well. Starting with version 6.3, file system will not be displayed using this flag. Use the LOGD option to display the file system related information.
/BDK	Displays the binary partition information for the selected DiskOnChip device, including the number of partitions on the media, start unit, end unit, capacity data, signature and protection for each individual partition. Information regarding sub-partitions is also provided.
/CLFILE:<filename>	Get command line parameters from file. This flag must be a single flag, get from command line.
/LOGD[bdtl][:drive]	Total number of logical partitions per DiskOnChip device, per BDTL. Prints information about logical partition number drives found at BDTL number BDTL. When BDTL is not specified, first disk partition information will be printed. When a drive is not specified – All logical partition on BDTL number BDTL information printed. When both BDTL and drive not specified – all DiskOnChip device logical partition information is printed. FS type will be printed for primary partitions only. Only the following FS types will be converted by DINFO to strings: 1, 2, 4, 5, 6, 7, 0x21, 0x25, 0x82 and 0x83. Other types will be printed as DINFO unknown. Number of FS will be printed in hexadecimal notation as well. Information regarding read protected partitions will not be printed.
/FAST	Causes TrueFFS to use the multiple byte/word read and write routines in the access layer.
/IPL	Provides information about the DiskOnChip IPL: • Type: RAM or ROM • Size: in bytes
/H	Displays the usage screen.
/RAMDEVFILE	See Section 7.

4.3. DINFO Usage Examples

Example: Getting DiskOnChip Configuration Information

```
DINFO    /WIN:d0000
```

Finds DiskOnChip at address 0xd0000 and displays its configuration information. The output is shown in Figure 3.

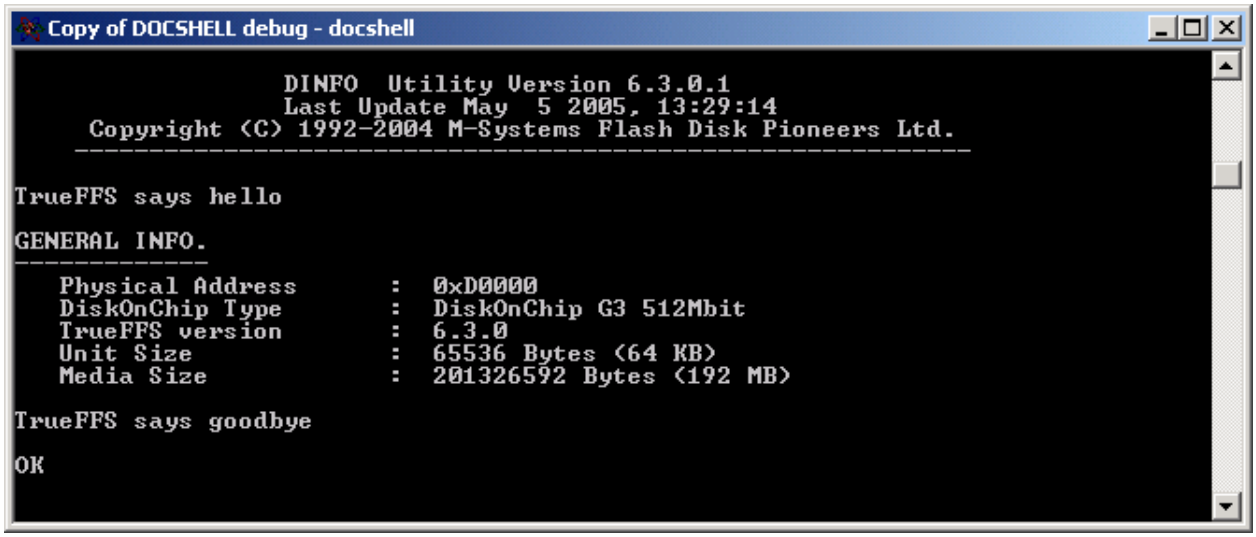

Figure 3: DiskOnChip Configuration Information Displayed Using the DINFO Utility

5. DFS UTILITY

This utility supplies file system based services to work with DiskOnChip. The utility will allow logical partitions format, read and update. MBR will also be updated by this utility

5.1. DFS syntax

The DFS syntax is:

```
Dfs </WIN:Address> </FLAG:parameter>[… /FLAG:parameter]
```

Where Address is the DiskOnChip address, /FLAG – Valid flag name with or without parameters.

5.2. Using DFS Flags

Table 10: DFS Flags

DFS Option	Description
/WIN:Address	Memory address where DiskOnChip is located. The address should be specified in hexadecimal format (for example, **/WIN:D0000**). If the **/WIN** option is not specified, the utility searches for DiskOnChip devices between default addresses. For example, when running the utilities with M-Systems EVBs for x86-based platforms, the default addresses are 0xC8000 – 0xE0000. All devices that are located will be related.
/FAST	Causes TrueFFS to use the multiple byte/word read and write routines in the access layer.
/SHIFT	Defines the address shift
/WIDTH	Defines the address width
/LOGD[k] :bdtl<n>	Creates/formats logical partition number 'k'; default is 0. On disk (bdtl) partition number 'n' – Must be specified. Up to 11 logical partitions (including the extended one) may be created using this flag. This means /log10 will be accepted, while /log11 will not be accepted.
[:t<type>[b]]	Set the partition type, which can be one of the following: 0 – primary, 1 – secondary, 2 – extended. When b is specified – A bootable partition is created. Default for single logical partition on BDTL is bootable when FS type is not specified, otherwise primary non-bootable partition. The type will be specified by number. Help message will display strings/numbers and type conversions for known utility types.
[:fs<file system>]	Place in MBR file system indicator <file system>, which can be one of the following: fat16, fat12, binfs. Default is DOS4. Specified by number in hexadecimal format. Help message will display strings/numbers and type conversions for known utility types. This type should not be specified for logical partition, in the event

DFS Option	Description
	that /FAT is used in the same command line for same logical partition. For example: /LOGD0:bdtl0:fs4 /FAT0:bdl0 – forbidden
[sz:<size>[m|k]]	Size of the partition will be notified as decimal number, where m - MB, k – KB Default – entire Last and Extended partition size MUST not be specified
/LOGF[k]:bdtl<n>:filename	Specifies the filename for logical partition. This file may be source files, when using in format or update mode and may be the target files, when used in read mode. Extended partition cannot be read/updated and not counted as logical partition for /LOGF flag. For example, when 3 primary, 1 extended and 2 secondary found on BDTL, /LOGF3:bdtl0:<filename> corresponds to first secondary partition.
/READ	Defines read mode operation for DFS
/KEY:<8bytes key>	In case of related BDTL protected – Specifies the key, should be inserted before starting the operation
/NOFORMAT	Specifies update mode operation for DFS
/FAT[log]:<BDTL[n]> [:roote<root_entries> [:clust_sz<cluster_size>] [:fats<num_fat_tab>] [:ID<volume_id>] [:lab<label>]	Places M-Systems' FAT format on logical partition number log, found on disk partition number BDTL. When log not specified – partition 0 will be formatted. FAT cannot be placed on extended partition. Extended partition not counted as logical partition for /FAT flag. For example, when 3 primary, 1 extended and 2 secondary found on BDTL0, /FAT3:bdtl0 corresponds to first secondary partition.
	Number of root entries per partition (1-65520), if not specified 65520
	Cluster size, when not specified – SureFS will choose the best option.
	Number of FAT tables, when not specified – 1 FAT table is placed.
	Partition ID, 4 bytes in length.
	Label for partition, up to 8 bytes

5.3. DFS Usage Examples

Example 1

```
Docshell /dfs /win:d0000 /logd0:bdtl0 /fat0:bdtl0
```

Creates a logical partition 0 at the disk partition 0 and places FAT on the first disk partition.

Example 2

```
docshell /dfs /win:d0000
/fat0:bdtl0:lab45678:roote20:clust_sz4:fats2:ID1234
/logd0:bdtl0:t0:fs4:sz2m
```

Creates a 2MB primary logical partition on the first disk partition and marks DOS as OS in MBR, then performs a FAT format on the first logical partition. During FAT format, the following parameters are set: label:'45678', ID: 1234, 2 FAT tables placed, number of root entries: 20, and the cluster size is 4 sectors.

Example 3

```
docshell /dfs /win:d0000 /fat0:bdtl0
```

This performs a FAT format on disk partition 0, without placing the MBR.

Example 4

```
docshell /dfs /win:d0000 /logd0:bdtl0 /logf0:bdtl0:1MB.img
/key0:12345678
```

Opens the HW protected disk partition 0 with key 12345678, creates a logical partition 0 on the entire disk partition 0 and writes the contents of the file 1MB.img on the created logical partition 0.

Example 5

```
docshell /dfs /win:d0000 /logf0:bdtl0:1MB.img /key0:12345678
/NOFORMAT
```

Opens the HW protected disk partition 0 with key 12345678, and updates the logical partition 0 with the contents of the file 1MB.img. The reminder of DiskOnChip is not changed as a result of this command.

Example 6

```
docshell /dfs /win:d0000 /logf0:bdtl0:1MB.img /key0:12345678 /READ
```

Opens the HW protected disk partition 0 with key 12345678, and reads the logical partition 0 contents to the file 1MB.img.

6. DIMAGE UTILITY

The DIMAGE utility is used to perform the following:

- Read the master DiskOnChip image to a file.
- Duplicate DiskOnChip, writing an image on up to 8 target DiskOnChip devices in parallel.
- Verify that the contents of the target devices match the master image file.

Warning: All target DiskOnChip devices must have the identical part number and capacity as the source DiskOnChip. For example, if the source DiskOnChip is DiskOnChip G3 64MB, then the target DiskOnChip must also be DiskOnChip G3 64MB.

The duplication process includes the following stages:

- Preparing the master DiskOnChip.
- Reading the contents of the master DiskOnChip to the image file.
- Burning the target DiskOnChip using the given image file.

Notes: 1. To mass-duplicate DiskOnChip efficiently, it is recommended to use one of the programmer solutions recommended by M-Systems. The availability list of all programmer solutions that support DiskOnChip is available on the M-Systems website www.m-systems.com.

2. Some references in the following section contain information for duplicating a master DiskOnChip to several target DiskOnChip devices in parallel. This information is relevant mainly for programmer solutions.

6.1. DIMAGE Syntax

6.1.1. Serial DIMAGE Syntax

- `DIMAGE /WINSRC:address /FILETRG:file PROTECTION-FLAGS`
- `DIMAGE /FILESRC:file /WINTRG:address PROTECTION-FLAGS`

Where `PROTECTION-FLAGS` can be a subset of the following:

```
[/SRCBDK#:password] [/SRCBDTL#:password] [/TRGBDK#:password]
[/TRGBDTL#:password] [/CHBDTL#:password][/CHBDK#:password]
```

6.1.2. Parallel DIMAGE Syntax

1. DIMAGE /TRGSOCK:sockets-bit-map /FILESRC:file PROTECTION-FLAGS
2. `DIMAGE /READSOCK:socket-number /FILETRG:file [/BASEFLOOR:n /FLOORS:m ] PROTECTION-FLAGS`
3. `DIMAGE /TRGSOCK:sockets-bit-map /VERIFYIMG:file PROTECTION-FLAGS`

4. `DIMAGE { /WINSRC:address or /FILESRC:file } { /WINTRG:address or /FILETRG:file} PROTECTION-FLAGS`
5. `DIMAGE /FILESRC:file /WINTRG:address PROTECTION-FLAGS`

   Where `PROTECTION-FLAGS` can be a subset of the following:

   `[/SRCBDK#:password] [/SRCBDTL#:password] [/TRGBDK#:password] [/TRGBDTL#:password] [/CHBDTL#:password][/CHBDK#:password]`

6.2. DIMAGE Flags

Table 11 describes the DIMAGE flag options:

Table 11: DIMAGE Flag Options (Regular and Parallel)

DIMAGE Option	Description
`/FILESRC:file`	Name of the source image.
`/FILETRG:file`	Name of the target image.
`/WINSRC:address`	Memory address where the source DiskOnChip is located.
`/WINTRG:address`	Memory address where the target DiskOnChip is located.
`/SRCBDK#:password`	Protection key of the protected binary partition on the source device (the default is 0). Inserting this flag disables the read/write protection of the specific binary partition.
`/SRCBDTL#:password`	Protection key of the protected disk partition on the source device (the default is 0). Inserting this flag disables the read/write protection of the specific disk partition.
`/TRGBDK#:password`	Sets the protection of the binary partition on the target DiskOnChip device, using the specified password.
`/TRGBDTL#:password`	Sets the protection of the disk partition on the target DiskOnChip device, using the specified password.
`/RAMDEVFILE:<filename>`	Specifies the name of the file. Will be mapped as Virtual DiskOnChip. When file does not exist – An error will be displayed.
`/SINGLE:floor`	Copies data from one floor in the image file to the target DiskOnChip device. Source image file MUST not include more floors than target DiskOnChip device.
`/FAST`	Causes TrueFFS to use the multiple byte/word read and write routines in the access layer.
`/VERIFY`	Verifies the programming operation.
`/BASEFLOOR`	Specifies the start floor in cases when an image is needed from DiskOnChip, but only the data that exists on specific floors should be read. This flag is used in conjunction with the **`/FLOORS`** flag.
`/FLOORS`	Specifies the number of floors to read in cases when an image is needed from DiskOnChip, but only the data that exists on specific floors should be read. This flag is used in conjunction with the **`/BASEFLOORS`** flag.
`/CLFILE`	Get the command line parameters from file.

DIMAGE Option	Description

Table 12: DIMAGE Flag Options (Parallel Only)

DIMAGE Option	Description
/VERIFYIMG	Performs parallel verify of the image file with the device contents. This option verifies that the devices contain the contents of the image file, however this option does not verify the reverse (e.g. the image file contains the contents of the device).
/READSOCK:n	Source socket number, starting from 0.
/TRGSOCK:nn	When performing parallel programming or parallel verify, this flag designates the sockets to be activated. nn is a bit map in hexadecimal format. All bits that represent an active socket should be set to 1. For example, to activate the first 5 sockets, the number 1F is used as the parameter. A maximum of 8 sockets may be used.

6.3. Creating the Master DiskOnChip

The preparation process for creating a master DiskOnChip image comprises the following steps:

- Formatting DiskOnChip, using the DFORMAT utility, on the target platform.
- Copying all target application files to DiskOnChip.

After the source DiskOnChip device has been properly prepared, follow the guidelines described in the following sections to duplicate it (as many times as required).

6.4. Copying the Source DiskOnChip to an Image File

At this stage, the master DiskOnChip includes all target application files, and it is ready to be duplicated. Use the DIMAGE utility to copy the master DiskOnChip contents to an image file.

To copy the source DiskOnChip to an image file:

1. Power off the system.
2. Insert the source DiskOnChip into the appropriate socket.
3. Power on the system.
4. Run the following command line:

```
DIMAGE  /WINSRC:Address   /FILETRG:image_file_name.IMG.
```

Note: When duplicating a DiskOnChip with active hardware protection, you must use DIMAGE with the **/SRCBDK#** or **/SRCBDTL#** flags in order to disable the protection. The password will be included in the virtual image for later use by DIMAGE.

Example

```
DIMAGE /WINSRC:D0000 /FILETRG:MYDOC.IMG /SRCBDK0:12345678
```

Copies the master DiskOnChip contents with a protected disk partition to the `MYDOC.IMG` file.

6.5. Copying the Image File to Target DiskOnChip Devices

At this stage, the contents of the source DiskOnChip is stored in an image file. Copying this image file to the target DiskOnChip devices results in identical DiskOnChip target devices. The DIMAGE utility is used for this purpose.

To copy the image file to the target DiskOnChip:

1. Power off the system.
2. Insert a target DiskOnChip device, with the same part number and capacity as the source DiskOnChip device, into the appropriate socket.
3. Power on the system.
4. Run the following command line:

   ```
   DIMAGE /FILESRC:image_file_name  /WINTRG:Address
   ```

   For programmer solutions only, use the following command line when writing the image to several devices in parallel:

   ```
   DIMAGE /FILESRC:image_file_name  /TRGSOCK:socket-bit-map
   ```

   When the duplication process is complete, the target DiskOnChip devices have the identical contents and functionality as the master DiskOnChip device. If the master DiskOnChip device is protected, then the target DiskOnChip devices are also protected with the same attributes and using the same passwords.
5. Repeat steps 1 through 4 as many times as required to duplicate additional target DiskOnChip devices.

Example 1

```
DIMAGE /FILESRC:MYDOC.IMG /WINTRG:D0000
```

Copy the contents of the `MYDOC.IMG` file to the target DiskOnChip.

Example 2

```
DIMAGE /FILESRC:MYDOC.IMG /WINTRG:D0000 /TRGBDK1=newpassw
```

The target image has the second binary partition protected with a password. The password for the second binary partition, **newpassw**, is inserted**.** The image of the source DiskOnChip is then copied to a target DiskOnChip.

Example 3

```
DIMAGE /FILESRC:MYDOC.IMG /TRGSOCK: 03
```

Copy the contents of the `MYDOC.IMG` file to the first 2 target DiskOnChip devices.

6.6. DIMAGE Error Messages

Table 13 lists possible DIMAGE error messages:

Table 13: DIMAGE Error Messages

Error Message	Description
Not enough memory	There is not enough system memory to complete the operation.
Can't open file	The system cannot open the image file. This may occur during write or read cycles (for example, if DIMAGE is trying to write the image to DiskOnChip but cannot locate the image file).
Adapter not found	The system cannot find a DiskOnChip adapter.
Can't write to file	The system cannot write to the image file. This may occur in DIMAGE when the media to which the system is writing suddenly becomes unavailable (Disk Full).
Source file format error	DIMAGE detected that the image file being written to the target device is not a valid image file.
Hardware protection	The system is trying to write to a target device with protected partitions. You must first remove the protection (DFORMAT using the `BDTLZ[n]` or the `BDKZ[n]` flag).
Too many bad blocks	The number of bad blocks on the target device is outside the acceptable range. Not enough space is available for writing the image. Note: This type of error does not normally occur unless DiskOnChip was misused.
Wrong parameters passed	There was an error in the parameters given in the command line.
Socket does not exist	The socket chosen by user does not exist
Source not compatible with target	The source DiskOnChip type is different from the target DiskOnChip type.

6.7. DIMAGE Socket Status Messages

Table 14 describes the DIMAGE socket status messages:

Table 14: DIMAGE Socket Status Messages

Socket Status Message	Description
Passed	Operation passed on this socket
Socket failed	Operation failed on this socket
Write failure	Write operation failed on this socket
Erase failure	Erase operation failed on this socket
Read BBT failure	Read Bad Block Table failed on this socket
Device in socket not compatible with image	Media type of the target device is different from that in the image file
Verify failure	Verify operation failed on this socket
Socket not found	Device was not found in this socket
Write fault	Program operation failed on this socket
HW protection	The device is protected (password is needed) or there is a protection error.
Time out	Timeout occurred while waiting for ready
Format failed	Device formatting failed.
Verify OTP failed	The OTP verification process failed
Verify IPL failed	The IPL verification process IPL failed
Complete write failure	Write operation failed on this socket
Write OTP failure	OTP write operation failed on this socket
Write IPL failure	IPL write operation failed on this socket
Activation of socket failed	Activation of this socket failed
Download operation failed	Download failed on this socket
Read OTP failed	Read OTP from device failed
Read IPL failed	Read IPL from device failed
Read physical info failed	Read physical info from device failed
Get extended info failed	Get extended info from device failed. Media may be unformatted.

6.8. Compatibility with Previous Versions

- 6.3.x based DIMAGE MUST not be used with 5.x formatted DiskOnChips, because of different TL algorithms.
- 6.3.x based DIMAGE will fail, when using to get image from a device which is NOT formatted with TrueFFS 6.3.x.
- 6.3.x based DIMAGE will fail when used to put image get by non 6.3.x based DIMAGE.

7. SPLITIMAGE UTILITY

SPLITIMAGE can be used when the source DIMAGE file is taken from a master DiskOnChip device with more than one floor. SPLITIMAGE extracts the master image file taken by the DIMAGE, to separate image files each representing an image of one or more floors.

The SPLITIMAGE output files can be used later by the DiskOnChip programmers.

7.1. SPLITIMAGE Syntax

SPLITIMAGE /f:InputFile /basefloor:n /floors:n /fileout:OutputFile

7.2. SPLITIMAGE Flags

SPLITIMAGE Option	Description
/basefloor:n	First floor (starting from 0)
/floors	Number of floors to extract
/f:filename	Input image file name (the product of DIMAGE get operation)
/fileout:filename	Output image file name (the input for DIMAGE put operation)
/? /H	Display help

7.3. SPLITIMAGE Examples

4floorsG3.img master file is produced by the DIMAGE utility on two cascaded DiskOnChip G3 1Gb devices. Use SPLITIMAGE to extract the 4floorsG3.img master file to two image files. The first file represents the first two floors (first DiskOnChip G3 1Gb in cascaded configuration). The second file represents the other two floors (second DiskOnchip G3 1Gb in cascaded configuration) as follows:

- `SPLITIMAGE /f:4floorsG3.img /basefloor:0 /floors:2 /fileout:01_4floorsG3.img`
- `SPLITIMAGE /f:4floorsG3.img /basefloor:2 /floors:2 /fileout:23_4floorsG3.img`

After executing the two commands, two files will be created: 01_4floorsG3.img – consisting of data read from floor 0 and floor 1 of source device and 23_4floorsG3.img - consisting of data read from floors 2 and floor 3 of the source devices.

8. UTILITIES RAM MTD UTILIZATION

RAM MTD simulates DiskOnChip. The DiskOnChip data is stored in a file built in TLV format (Type-Length-Value elements), in order to allow it to be easily distinguished and found. Type and Length are 4 bytes each field, the length field consists of the value length. TLV elements may encapsulate other TLV elements in the value field. File search seeks for the first occurrence of the Type in file. For this reason each Type should appear only once in the file.

8.1. RAM MTD Flags

Table 15: RAM MTD Flags

Flag name	Description
RAMDEVTYPE:<type>	Specifies the NEW RAM device type simulation. This flag should be specified only when a new virtual DiskOnChip is to be created. Required only when a new virtual DiskOnChip is to be created. This file is accepted by DFORMAT only. If a targeted file exists in the working directory, a new file will NOT be created.
RAMDEVFILE:<filename>	Specifies existing RAM MTD file name, when it should be used; or defines the RAM MTD file name, when file should be created by type or from dump file

When RAMDEVFILE is not specified and a new file is created by specifying the RAMDEVTYPE, the name of the file will consist of the DiskOnChip type (see default file names).

8.2. RAM MTD Supported Devices

Table 16: RAM MTD Supported Devices

DiskOnChip Device	RAMDEVTYPE	Default File Name
G3 512Mb	1	"G3_512MB_BGA_TSO.IMG"
2 cascaded G3 512Mb	2	"G3_1GB_BGA.IMG"
4 cascaded G3 512Mb	3	"G3_1GB_BGA_CASCADE_2.IMG"
P3 256Mb	4	"P3_256MB_BGA_TSO.IMG"
H1T 4Gb	5	"H1_4GB_BGA.IMG"
H1T 8Gb	6	"H1_8GB_BGA.IMG"
2 cascaded H1T 8Gb	8	"H1_8GB_BGA_CASCADE_2.IMG"
G3 LP 512Mb	9	"G3_512MB_BGA_TSO_LP.IMG"
P3 LP 256Mb	10	"P3_256MB_BGA_TSO_LP.IMG"
G4 1Gb	12	"G4_1Gb_BGA.IMG"
2 cascaded G4 1Gb	13	"G4_2Gb_BGA.IMG"
H1H 4Gb	15	"H1_4Gb_HYN_BGA.IMG"
H1H 2Gb	16	"H1_2Gb_HYN_BGA.IMG"
G4 1Gbit, 4 floors	17	"G4_4Gb_BGA.IMG"
G3 512Mb	1	"G3_512MB_BGA_TSO.IMG"
2 cascaded G3 512Mb	2	"G3_1GB_BGA.IMG"

9. UTILITY PACKAGE FOR WINDOWS 2000 AND WINDOWS XP

The **DOCSHELL** joined utilities (DINFO, DFORMAT, DFS, DIMAGE etc) can be obtained in different packages for running on different OSs. For example, there are separate utility packages for DOS and Windows 2000/XP. All general information described in the previous sections, including the utility command line syntax, applies for all OS utility packages. This section specifically discusses the Windows 2000/XP package, describing the special installation requirements and customization features.

9.1. The Utility DLL Package

9.1.1. Purpose

The DLL provides flexibility to the user, enabling the adjustment of the core utility to the target platform without re-compiling the utility itself. You can install your own implementation of low-level read/write routines, or customize the progress bar and print functions used by the utility.

The JTAG is an example of a user implementation of the low-level read and write routine. Installing JTAG read and write routines enables formatting, identifying and even programming any DiskOnChip device on any type of platform/CPU with a JTAG connection, without changing the utility.

The print routine may require customization when the utility is integrated with an existing tool. The utility inputs and outputs can be redirected to the existing tool user interface.

9.1.2. Package Description

The utility DLL package is comprised of the following folders:

- Install Shield
- Binary
- Customization

9.1.2.1 Install Shield

Applications that run on Windows at the user level cannot access physical memory directly. Instead, they can access virtual memory pointing to that physical memory location. M-Systems supplies a dedicated driver called Mapmemory.dll, which allows the utility to map the DiskOnChip memory range into virtual memory. The driver can be easily installed using a dedicated Install Shield installation wizard.

This folder contains the files required to run the wizard that installs Mapmemory.dll (virtual-to-physical memory driver). You must install the Mapmemory.dll before running the utilities included in the Binary folder.

To install Mapmemory.dll:

1. Launch the installation wizard using the setup.exe program.
2. A Welcome screen is displayed. Click **Next**.
3. The wizard displays a list of the files that are going to be installed. When the subsequent screen is displayed, click **Next**.
4. The driver files are installed in the specified location, and the wizard informs the user that the installation process is complete.
5. When the appropriate screen is displayed, click **Finish**.
6. The wizard prompts you to restart your computer before using the M-Systems utility.

9.1.2.2 Binary

DINSHELL utility is a DLL file based on the TrueFFS SDK. The file is referred to as Utility_dll.dll,. The Binary folder includes three Utility_dll.dll files: Dinfo_dll.dll, Dformat_dll.dll and Dimage_dll.dll.

In addition, the folder contains a set of Utility.exe files (one for each DLL) that is used for loading and running the DLLs.

To run a utility:

1. Install Mapmemory.dll using the installation wizard.
2. Open a command prompt window.
3. Execute the required utility from the Binary folder.

If you use the standard Windows 2000/XP OS with an M-Systems EVB, the utilities do not require customization, and there is no need to access the Customization folder.

9.1.2.3 Customization

This folder contains the necessary tools for adapting the utilities to meet specific customer requirements. It contains the following folders:

- **Output\release**: This folder consists of docsehll.dll file . All projects in the customization folder will automatically copy its output to this folder.
- **Src\Access_wrapper**: This folder contains a Visual Studios 6 project that produces a second DLL containing the user implementation for some or all of the DiskOnChip access routines. This file may contain functions such as write-byte, read-byte and write-block functions (See Table 17 for a complete list of user-defined access routines.) The file created is called user_access.dll, and is automatically copied to the Binary release folder.
- **Src\Os_wrapper**: This folder contains a Visual Studio 6 project that produces a third DLL containing the user implementation for system-dependent services, such as the print function. (See Table 19 for a complete list of user-defined OS routines.) The file created is called user_os.dll, and is automatically copied to the Binary release folder.

- **Src\Docshell**: This folder contains a sample application for the DOCSHELL utility. The example is given as a Visual Studio 6 project that:
    - Loads the docshell.dll file.
    - Sets the names of the user_access dll and user_os dll files that will be used.
    - Calls the docsehll.dll main entry point with the given arguments.

  The .EXE file is automatically copied to the Output\release folder.

9.1.2.4 Functionality

Docshell components described above (**docshell.dll, docshell_dll.exe, user_access.dll** and **user_os.dll**) must be in the same folder.

When **Docshell_dll.exe** is launched, the following occurs:

1. **Docshell_dll.exe** designates user_access.dll and os_access.dll as the default user-implemented DLLs.
2. Utility.exe loads **Docshell_dll.dl**l and calls the main entry point (`ExMainDll`) with the user arguments.
3. **Docshell_dll.dll** loads **user_access.dll** and **user_os.dll** and looks for the customizable routines. If a specific routine is not found, or if the DLL itself is not present, **Utility_dll.dll** uses its own default implementation.
4. **Docshell_dll.exe** executes the docshell engine using the installed resources.

9.1.3. Customization Routine API

9.1.3.1 User-Defined Access Routines

The routines in this section should be implemented in the user **Access Wrapper** project, which produces user_access.dll as output.

To insert your implementation, uncomment the relevant function and add your code. It is not necessary to customize all of the routines in the access.dll. Routines that are not customized simply use the default implementation.

If you have customized the block access routines, you must use the /FAST flag with all utilities.

Table 17: Functions That May Be Customized in user_access.dll

Function	Description
`void EXAPI Write_Byte(FLByte val,volatile void* address);`	Writes a byte to the address location.
`void EXAPI Write_Word(FLWord val,volatile void* address);`	Writes a word to the address location.
`void EXAPI Write_Dword(FLDword val,volatile void* address);`	Writes a double word to the address location.
`FLByte EXAPI Read_Byte(volatile void* address);`	Reads a byte from the address location.
`FLWord EXAPI Read_Word(volatile void* address);`	Reads a word from the address location.
`FLDword EXAPI Read_Dword(volatile void* address);`	Reads a double word from the address location.
`void memcpy_from_io_8bit(EXBYTE EXFAR *dest, volatile EXBYTE EXFAR * src, EXWORD count)`	Copies a block of bytes from the flash to a destination address.
`void memcpy_to_io_8bit( volatile EXBYTE EXFAR *dest,EXBYTE EXFAR *src, EXWORD count)`	Copies a block of bytes from a memory address to the flash address.
`void memset_to_io_8bit( volatile EXBYTE EXFAR * dest,EXBYTE val, EXWORD count)`	Sets a block of memory to a specified value.
`void memcpy_from_io_16bit(EXBYTE EXFAR *dest, volatile EXBYTE EXFAR * src, EXWORD count)`	Copies a block of words from the flash to a destination address.
`void memcpy_to_io_16bit( volatile EXBYTE EXFAR *dest,EXBYTE EXFAR *src, EXWORD count)`	Copies a block of words from a memory address to the flash address.
`void memset_to_io_16bit( volatile EXBYTE EXFAR * dest,EXBYTE val, EXWORD count)`	Sets a block of memory to a specific value.
`void Get_Search_Range(EXDWORD*low_range,EXDWORD *high_range)`	Gets the search range for DiskOnChip.
`void *Map_Memory(unsigned long dwAddress,unsigned long dwLen)`	Maps a block of physical memory to virtual memory.

Table 18: Parallel Access Functions that May Be Customized in user_access.dll (for Programmers Only)

Function	Description
`void PARA_Write8BitRoutine (volatile FLByte FAR1 *win,FLWord wOffset,FLByte bVal)`	Writes the same data (byte) to the same address in parallel to all activated sockets.
`void PARA_Write16BitRoutine (volatile FLByte FAR1 *win,FLWord wOffset,FLWord wVal)`	Writes the same data (word) to the same address in parallel to all activated sockets .
`void PARA_Write8BitDifferentDataRoutine`	Writes different data (byte) to the

(volatile FLByte FAR1 *win, FLWord wOffset,FLByte *bDataArr)	same address in parallel to all activated sockets.
void PARA_WriteBufferRoutine (volatile FLByte FAR1 *win, FLWord wOffset, FLByte FAR1* BufferArr, FLWord wBufferLength)	Writes a buffer to the same address in parallel to all activated sockets.
FLBoolean PARA_Verify8BitRoutine (volatile FLByte FAR1 *win,FLWord wOffset, FLByte bCompareByte, FLByte bByteMask,FLBoolean *fCompareArr)	Reads a byte from all sockets and compares it to a predefined value.
FLBoolean PARA_VerifyBufferRoutine (volatile FLByte FAR1*win,FLWord wOffset,FLByte FAR1* bCompareBufferArr, FLWord wBufferLength, FLBoolean *fCompareArr)	Reads a buffer from all sockets and compares it to a predefined value.
FLStatus PARA_ActivateSocketsRoutine (FLBoolean *fSocketStateArr)	Enables/Disables sockets.
FLStatus PARA_GetSocketStatusRoutine (FLBoolean *fSocketStateArr)	Returns the status of the sockets.
FLStatus PARA_AccessLayerInitRoutine (FLFlash *flashPtrArr[])	Initializes the access layer.
EXBOOL EXAPI ExIsDigit(EXCHAR ch)	Determines whether the given character is digit or not

9.1.3.2 User-Defined OS Routines

The routines in this section should be implemented in the user_os.dll project. To insert your implementation, uncomment the relevant function and add your code. It is not necessary to customize all of the routines in the file. Routines that are not customized simply use the default implementation.

Table 19: Functions That May Be Customized in user_os.dll

Function	Description
ExStatus EXAPI ExOsOpenCmdLineDevice(void** data);	Opens a command line device.
ExStatus EXAPI ExOsCloseCmdLineDevice(void* data);	Closes a command line device.
ExStatus EXAPI ExOsReadCmdLineDevice(void*data, EXWORD argc, void** retCommand);	Reads a command from a command line device.
ExStatus EXAPI ExOsOpenMsgDevice(void** data);	Opens a message device.
ExStatus EXAPI ExOsCloseMsgDevice(void* data);	Closes a message line device.
ExStatus EXAPI ExOsWriteMsgDevice(void* data, EXCHAR* pStr);	Writes a string to a message device.
ExStatus EXAPI ExOsOpenErrorMsgDevice(void** data);	Opens an error message device.
ExStatus EXAPI ExOsCloseErrorMsgDevice(void* data);	Closes an error message line device.

Function	Description
`ExStatus EXAPI ExOsWriteErrorMsgDevice( void* data, EXCHAR* pStr );`	Writes string to error message device
`ExStatus EXAPI ExOsOpenPrgBarDevice( void** data, EXCHAR* pPrgBarName, EXDWORD dwPrgBarSize );`	Opens a Progress Bar device.
`ExStatus EXAPI ExOsClosePrgBarDevice( void* data );`	Closes the Progress Bar line device.
`ExStatus EXAPI ExOsWritePrgBarDevice( void* data, EXDWORD dwDoneSoFar );`	Writes a string to the Progress Bar device.
`ExStatus ExOsOpenFile( void** data, EXCHAR* name, EXWORD wMode );`	Opens a file device.
`ExStatus EXAPI ExOsCloseFile( void* data, EXWORD wMode );`	Closes a file.
`ExStatus EXAPI ExOsWriteToFile( void* data, EXDWORD size, void EXFAR* buff );`	Writes a buffer to a file.
`ExStatus EXAPI ExOsReadFromFile( void* data, EXDWORD size, void EXFAR* buff, EXDWORD* byteRead );`	Reads a buffer from a file.
`ExStatus EXAPI ExOsGetLengthOfSerialFileDevice( void* data, EXDWORD* retLen );`	Gets the length of a file.
`ExStatus EXAPI ExOsOpenUserInputDevice( void** data );`	Opens a user input device
`ExStatus EXAPI ExOsCloseUserInputDevice( void* data );`	Closes a user input device
`ExStatus EXAPI ExOsWaitForCharUserInputDevice( void*data, ExWaitForCharChoices* waitForCharChoices );`	Prints user choices and waits for char from the user input device.
`void* EXAPI ExMemAlloc( EXDWORD size );`	Allocates a memory block.
`void EXAPI ExMemFree( void* memBlock );`	Frees a memory block.
`void* EXAPI ExMemSet( void* dest, EXWORD ch, EXDWORD size );`	Sets a buffer to a specified character.
`EXWORD EXAPI ExMemCmp( void* buf1, void* buf2, EXDWORD count );`	Compares characters between two buffers.
`void* EXAPI ExMemCpy( void* dest, void* src, EXDWORD count );`	Copies a character between buffers.
`EXSWORD EXAPI ExRand( void );`	Gets a random number.
`void EXAPI ExSrand( EXWORD seed );`	Sets the starting point for generating a series of pseudorandom integers.
`EXDWORD EXAPI ExTime( EXSDWORD* timer );`	Gets the system time.
`EXCHAR* EXAPI ExStrTime( EXCHAR* timestr );`	Copies the system time to a buffer.
`EXCHAR* EXAPI ExStrDate( EXCHAR* datestr );`	Copies the date to a buffer, in the format mm/dd/yy.
`EXSDWORD EXAPI ExStrtol( const EXCHAR* nptr, EXCHAR** endptr, EXWORD radix );`	Converts strings to a long-integer value.
`EXDWORD EXAPI ExStrtoul( const EXCHAR* nptr,`	Converts strings to an unsigned long-

Function	Description
`EXCHAR** endptr, EXWORD radix );`	integer value.
`EXSWORD EXAPI ExAtoi( const EXCHAR* string );`	Converts strings to an integer value.
`EXCHAR* EXAPI ExItoa( EXSWORD value, EXCHAR* string, EXSWORD radix );`	Converts an integer to a string.
`void EXAPI ExClearScreen( void );`	Clears the screen.
`EXWORD EXAPI ExStrLen( const EXCHAR* string );`	Gets the length of a string.
`EXCHAR* EXAPI ExStrCat( EXCHAR* strDestination, const EXCHAR* strSource );`	Appends a string.
`EXWORD EXAPI ExCreateDirectory( const EXCHAR* dirName );`	Creates a new directory.
`EXWORD EXAPI ExGetDrive( void );`	Gets the current disk drive.
`EXWORD EXAPI ExChangeDrive( EXWORD drive );`	Changes the current working drive.
`EXWORD EXAPI ExGetDiskFree( EXWORD drive, ExDiskInfoRecord* diskInfo );`	Gets information regarding the number of available sectors, their size, number of clusters, and sector per cluster.
`EXCHAR EXAPI ExToupper( EXCHAR ch );`	Converts lower case to upper case.

HOW TO CONTACT US

USA

M-Systems, Inc.
555 North Mathilda Avenue, Suite 220
Sunnyvale, CA 94085
Phone: +1-408-470-4440
Fax: +1-408-470-4470

China

M-Systems China Ltd.
Room 121-122
Bldg. 2, International Commerce & Exhibition Ctr.
Hong Hua Rd.
Futian Free Trade Zone
Shenzhen, China
Phone: +86-755-8348-5218
Fax: +86-755-8348-5418

Japan

M-Systems Japan Inc.
Asahi Seimei Gotanda Bldg., 3F
5-25-16 Higashi-Gotanda
Shinagawa-ku Tokyo, 141-0022
Phone: +81-3-5423-8101
Fax: +81-3-5423-8102

Europe

M-Systems Ltd.
7 Atir Yeda St.
Kfar Saba 44425, Israel
Tel: +972-9-764-5000
Fax: +972-3-548-8666

Taiwan

M-Systems Asia Ltd.
14 F, No. 6, Sec. 3
Minquan East Road
Taipei, Taiwan, 104
Tel: +886-2-2515-2522
Fax: +886-2-2515-2295

Internet

http://www.m-systems.com/mobile

General Information

info@m-systems.com

Sales and Technical Information

techsupport@m-systems.com