msystems mdoc

# mDOC H1 4Gb (512MByte) and 8Gb (1GByte)

## High Capacity Flash Disk with NAND and x2 Technology

Data Sheet, Rev. 1.1

### Highlights

mDOC H1 is one of the industry's most efficient memory solutions for high capacity data and code storage, using 90 nm process NAND flash and x2 technology from msystems.

x2 technology enables NAND flash to achieve highly reliable, high-performance data and code storage with a specially designed error detection and correction mechanism, optimized file management, and proprietary algorithms for enhanced performance.

This technology provides a robust, high-capacity memory solution to answer the needs of new multimedia devices such as PDA, smart phones, music phones, game phones and camera phones.

Further cost benefits derive from the cost effective architecture of mDOC H1, which includes a boot block that can replace expensive NOR flash, and incorporates both the flash memory and an embedded controller in a single package.

mDOC H1 provides:

- Flash disk for both code and data storage
- Low voltage: 1.8V or 3.3 I/O (auto-detect), 3.3V core
- Hardware protection and security-enabling features
- High capacity: 4Gb (512MB) and 8Gb (1GB)
- Device cascade capacity up to two devices: 16Gb (2GB)
- Enhanced Programmable Boot Block enabling eXecute In Place (XIP) functionality using 16-bit interface.
- Unrivaled data integrity with a robust Error Detection Code/Error Correction Code (EDC/ECC) tailored for NAND flash technology
- Maximized flash endurance with TrueFFS®
- Support for major mobile operating systems (OSs), including: Symbian, Windows Mobile, Palm, Nucleus, Linux
- Compatible with major CPUs, including TI OMAP, Intel XScale, FreeScale MX, and Qualcomm MSMxxxx

### Performance

- Access time: 64 nsec
- Sustained read: 8 MB/sec
- Sustained write: 3.8 MB/sec

### Protection & Security-Enabling Features

- 16-byte Unique Identification (UID) number
- 6KByte user-controlled One Time Programmable (OTP) area
- A configurable hardware-protected partition for data and code:
  - Read-only mode
  - Write-only mode
  - Protection key and LOCK# signal
  - Sticky Lock (SLOCK) to lock boot partition
  - Protected Bad Block Table

### Reliability and Data Integrity

- Hardware- and software-driven, on-the-fly EDC and ECC algorithms

- 4-bit Error Detection Code/Error Correction Code (EDC/ECC), based on a patented combination of BCH and Hamming code algorithms, tailored for NAND flash technology
- Guaranteed data integrity after power failure
- Transparent bad-block management
- Dynamic and static wear-leveling

### Boot Capability

- Programmable Boot Block with XIP capability to replace boot ROM – 1KB
- Download Engine (DE) for automatic download of boot code from Programmable Boot Block
- Asynchronous Boot

### Hardware Compatibility

- Configurable interface: simple NOR-like or multiplexed address/data interface
- CPU compatibility, including:
  - Texas Instruments (TI) OMAP, DBB
  - Intel XScale PXAxxx family
  - Infinion xGold family
  - Analog Devices (ADI) AD652x family
  - Freescale MX family
  - Emblaze ER4525
  - Renesas SH mobile
  - Qualcomm MSMxxxx
  - Hitachi SuperH™ SH-x
- Supports 8 and 16 bit architectures.

### TrueFFS® Software

- Full hard-disk read/write emulation for transparent file system management
- Patented TrueFFS
  - Flash file system management
  - Automatic bad block management
  - Data management to maximize the limit of typical flash life expectancy
  - Dynamic virtual mapping
- Dynamic and static wear-leveling
- Programming, duplicating, testing and debugging tools available in source code

### Operating Environment

- Wide OS support, including:
  - Symbian
  - Microsoft Windows Mobile
  - Palm
  - Nucleus
  - Linux
  - OSE
- TrueFFS Software Development Kit (SDK) for quick and easy support for proprietary OSs, or OS-less environment
- TrueFFS Boot Software Development Kit (BDK)

### Power Requirements

- Operating voltage:
  - Core 2.7V – 3.6V
  - I/O 1.65V-1.95V or 2.5-3.6V (auto-detect)
- Current Consumption
  - Active mode: 44 mA
  - Deep Power-Down mode: 25 µA

### Packaging

- 115-ball FBGA package: 12x18x1.4 mm

### Capacity

- 4 Gbit (512MB)
- 8 Gbit (1GB)
- 16 Gbit (2GB), cascading two mDOC H1 8Gb (1GB)

# REVISION HISTORY

| Revision | Date | Description | Reference |
|---|---|---|---|
| 0.1 | December 2004 | Updated system interface description | Sections 1.2.2, 1.3.2 |
| | | Updated device cascading information | Section 7.6 |
| | | Updated order information | Section 9 |
| | | Updated standard interface read cycle timing parameters | Sections 8.3.1, 8.3.2, 8.3.3, 8.3.4 |
| 0.2 | March 2005 | Updated package and ballout information | Sections 1.2, 1.3 |
| 0.3 | June 2005 | Mechanical dimensions layout graphic | Section 8.4 |
| 0.4 | June 2005 | Chip Identification (ID) Register and Chip Identification Confirmation Register corrected | Section 5.4 |
| 0.5 | July 2005 | Accommodations to account for dual source and highlights section updated | Sections 1.1, 8.2 |
| 0.6 | August 2005 | Updated Device Signal Descriptions | Sections 1.2.3, 1.3.3 |
| | | Updated Modes of Operation | Section 4 |
| | | Updated Product Specifications | Sections 8.1, 8.2, 8.3 |
| 1.0 | December 2005 | Timing specification diagrams | Section 8.3 |
| 1.1 | August 2006 | Updated Ambient Temperature (TA) | Section 8.2.4 Table 9 |

# TABLE OF CONTENTS

## Introduction

This data sheet includes the following sections:

**Introduction:** Overview of product and data sheet contents

**Section 1:** Product overview, including a brief product description, ball diagrams and signal descriptions

**Section 2:** Theory of operation for the major building blocks

**Section 3:** Detailed description of hardware protection and security-enabling features

**Section 4:** Detailed description of modes of operation and TrueFFS technology, including power failure management and 8KByte memory window

**Section 5:** mDOC H1 register descriptions

**Section 6:** Overview of how to boot from mDOC H1

**Section 7:** Hardware and software design considerations

**Section 8:** Environmental, electrical, timing and product specifications

**Section 9:** Information on ordering mDOC H1

For additional information on msystems' flash disk products, please contact one of the offices listed on the back page.

# 1. PRODUCT OVERVIEW

## 1.1 Product Description

mDOC H1 is one msystems' latest developments in industry-leading memory solutions for high-capacity data and code storage. mDOC H1, packed in a small FBGA package with 4Gb (512MB) or 8Gb (1GB) capacity, is a device with an embedded thin flash controller and flash memory. It uses 90 nm process NAND-based flash technology, enhanced by msystems' advanced x2 technology.

msystems' proprietary technology overcomes NAND-related error patterns and slow transfer rates by using a robust error detection and correction (EDC/ECC) mechanism. The combination of NAND and x2 technology results in a low-cost, minimal-sized flash disk that achieves unsurpassed reliability levels, enhanced performance and high capacity.

This breakthrough in performance, capacity, size and cost makes mDOC H1 the ideal solution for product manufacturers who require high-capacity, small size, high-performance, and above all, high-reliability storage to enable applications such as Digital TVs (DTVs), rugged handheld terminals, Digital Still Cameras (DSCs), Mobile Point of Sale (POS), telecom equipment, multimedia phones, camera and Video on Demand (VOD) phones, enhanced Multimedia Messaging Service (MMS), gaming, music, video and Personal Information Management (PIM) on mobile handsets, and Personal Digital Assistants (PDAs).

As with msystems' mDOC G3 family, content protection and security-enabling features offer several benefits. A write- and read-protected partition, with both software- and hardware-based protection, can be configured independently for maximum design flexibility. The 16-byte Unique ID (UID) number identifies each flash device, eliminating the need for a separate ID device on the motherboard. The 6KB One Time Programmable (OTP) area, written to once and then locked to prevent data and code from being altered, is ideal for storing customer and product-specific information.

mDOC H1 has a 1KB Programmable Boot Block providing eXecute In Place (XIP) functionality, enabling mDOC H1 to replace the boot device and function as the only non-volatile memory device on-board. Eliminating the need for an additional boot device reduces hardware expenditures, board real estate, programming time, and logistics.

msystems' patented TrueFFS software technology fully emulates a hard disk to manage the files stored on mDOC H1. This transparent file system management enables read/write operations that are identical to a standard, sector-based hard disk. In addition, TrueFFS employs patented methods, such as virtual mapping, dynamic and static wear-leveling, and automatic bad block management to ensure high data reliability and to maximize flash life expectancy.

## 1.2 Standard Interface

### 1.2.1 Ball Diagram

See Figure 1 for the mDOC H1 standard interface ballout. To ensure proper device functionality, balls marked RSRVD are reserved for future use and should either be left floating or connected to the recommended signals.

Note: mDOC H1 is designed as a drop-in replacements for mDOC G3 128MB (1Gb), G3 64MB (512Mb), G3 LP 64MB (512Mb), P3 32MB (256Mb) and P3 LP 32MB (256Mb), assuming the board was designed according to migration guidelines. Refer to the *mDOC G3/P3 to mDOC G3/P3 LP, G4 and H1 migration guide* for further information.

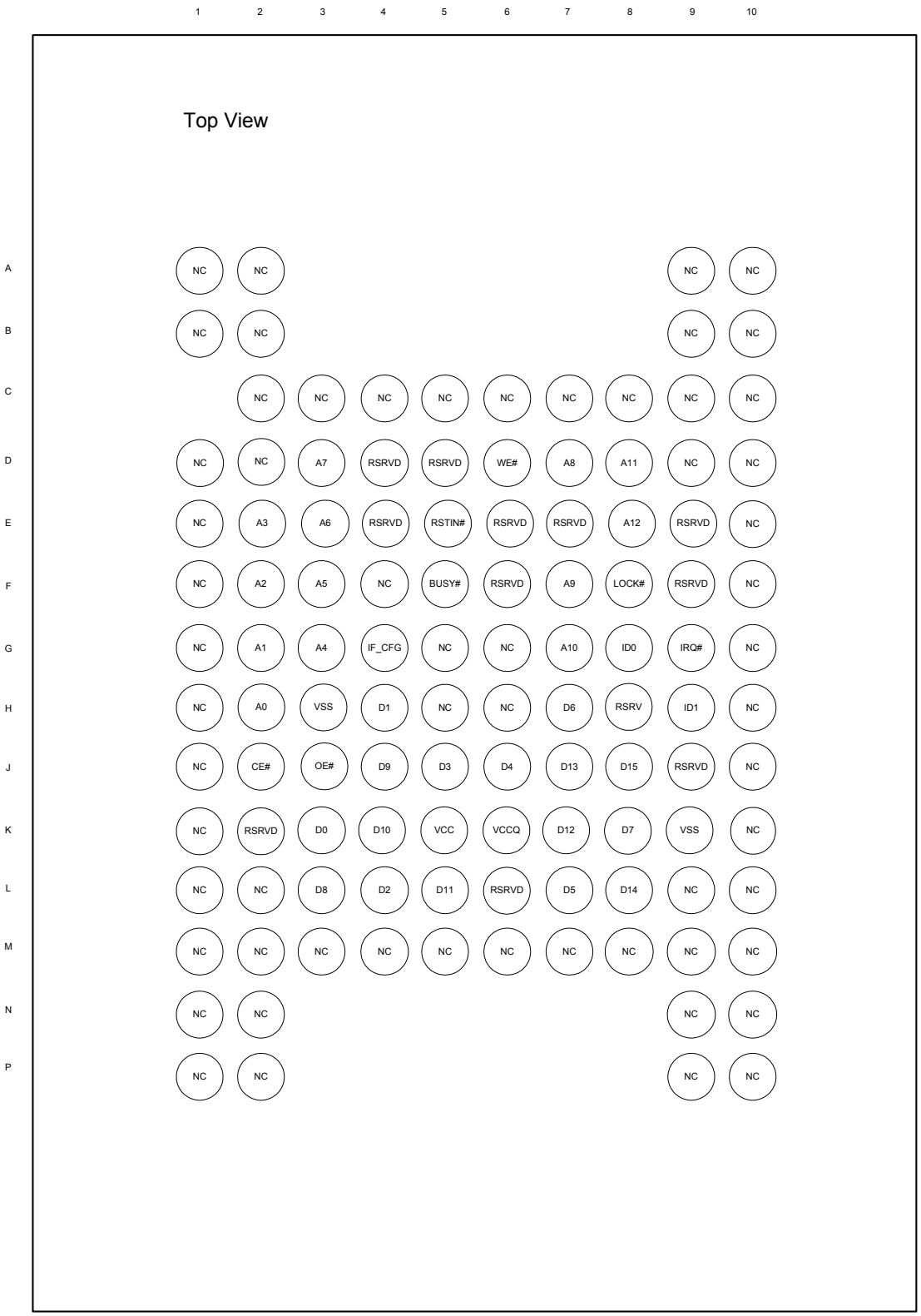



*Figure 1 Standard Interface Ballout for mDOC H1 12x18 FBGA Package*

### 1.2.2 System Interface

See Figure 2 for a simplified I/O diagram for a standard interface of mDOC H1.

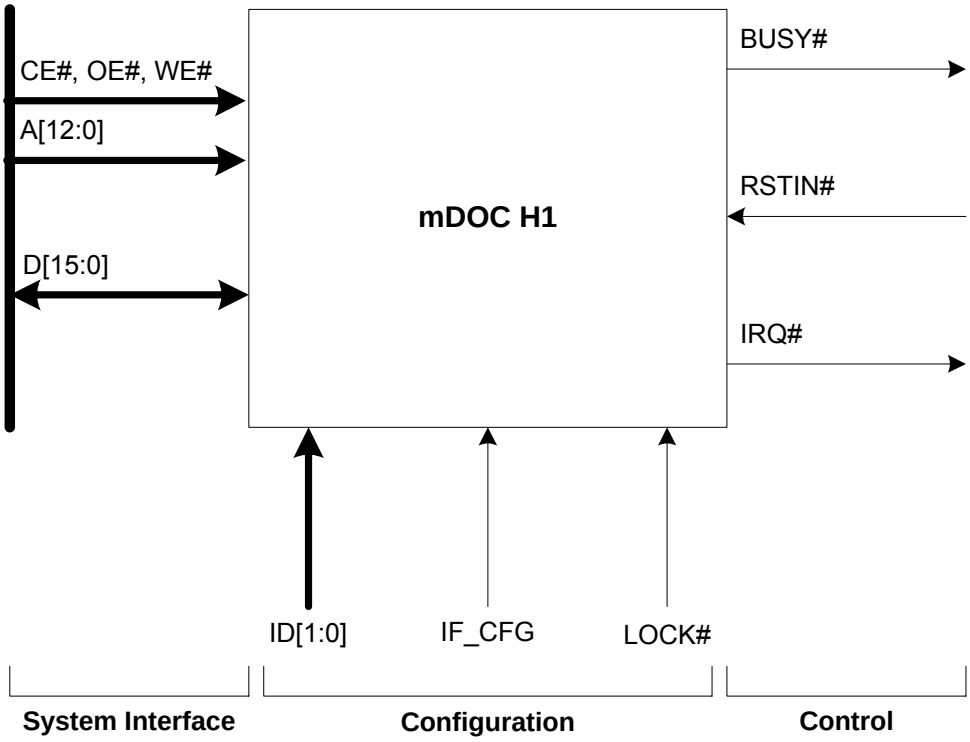



*Figure 2: Standard Interface Simplified I/O Diagram for mDOC H1*

### 1.2.3 Signal Description

*Table 1: Standard Interface Signal Descriptions for mDOC H1 12x18 FBGA Package*

| Signal | Ball No. | Input Type[1,2] | Description | Signal Type |
|---|---|---|---|---|
| **System Interface** | | | | |
| A[12:11]<br>A[10:8]<br>A[7:4]<br>A[3:0] | E8, D8<br>G7, F7, D7<br>D3, E3, F3, G3<br>E2, F2, G2, H2 | PCI | Address bus.<br>When IF_CFG is set to 1 (16-bit mode) it is recommended to connect signal A0 (ball H2) to GND via pull-down resistor to maintain backwards compatibility. | Input |
| D[15:14]<br>D[13:12]<br>D[11:8] | J8, L8<br>J7, K7<br>L5, K4, J4, L3 | PCI, R8 | Data bus, high byte.<br>When IF_CFG is set to 0 (8-bit mode) these balls may be left floating. | Input/ Output |
| D[7:6]<br>D[5:3]<br>D[2:0] | K8, H7<br>L7, J6, J5<br>L4, H4, K3 | PCI | Data bus, low byte. | Input/ Output |
| CE# | J2 | ST | Chip Enable, active low. | Input |
| OE# | J3 | ST | Output Enable, active low. | Input |
| WE# | D6 | ST | Write Enable, active low. | Input |
| **Configuration** | | | | |
| ID[1:0] | H9, G8 | ST | Identification. Configuration control to support up to two chips cascaded in the same memory window.<br>Chip 1 = ID1, ID0 = VSS, VSS (logic 0, logic 0); required for single chip<br>Chip 2 = ID1, ID0 = VSS, VCCQ (logic 0, logic 1) | Input |
| LOCK# | F8 | ST | Lock, active low. When active, provides full hardware data protection of selected partitions. | Input |
| IF_CFG | G4 | ST | Interface Configuration, 1 (VCCQ) for 16-bit interface mode, 0 (VSS) for 8-bit interface mode. | Input |
| **Control** | | | | |
| BUSY# | F5 | OD | Open drain output held asserted (low) until download process has completed.<br>A 10 KΩ pull-up resistor is recommended even if this signal is not used. | Output |
| RSTIN# | E5 | ST | Reset, active low. | Input |
| IRQ# | G9 | OD | Open drain interrupt request output.<br>A 10 KΩ pull-up resistor is recommended even if this signal is not used. | Output |

| Signal | Ball No. | Input Type[1,2] | Description | Signal Type |
|---|---|---|---|---|
| **Power** | | | | |
| VCC | K5 | - | Device core power supply.<br>Requires a 10 nF and 0.1 µF capacitor. | Supply |
| VCCQ | K6 | - | I/O power supply. Sets the logic **1** voltage level range of I/O balls. VCCQ may be either 2.5V to 3.6V or 1.65V to 1.95V. Requires a 10 nF and 0.1 µF capacitor. | Supply |
| VSS | H3, K9 | - | Ground. All VSS balls must be connected. | Supply |
| **Other** | | | | |
| RSRVD | K2, E7 | - | Reserved signal. Not connected internally.<br>Future mDOC FBGA 12x18 devices will use this ball. It is therefore recommended to connect this ball to GND via a 1 µF capacitor to guarantee forward compatibility. | |
| | H8 | | Reserved signal.<br>Not connected internally. For forward compatibility with future products, it is recommended to connect this ball as DMA request (DMARQ#). | |
| | L6 | | Reserved signal.<br>Not connected internally. For forward compatibility with future products, it is recommended to connect this ball as an external system clock input (CLK). | |
| | D4, D5, E4, E6, E9, F6, F9, J9, | | Reserved. Other reserved signals are not connected internally and must be left floating to guarantee forward compatibility with future products. | |
| NC | A1, A2, A9, A10, B1, B2, B9, B10, C2, C3, C4, C5, C6, C7, C8, C9, C10, D1, D2, D9, D10, E1, E10, F1, F4, F10, G1,G5, G6, G10, H1, H5, H6, H10, J1, J10, K1, K10, L1, L2, L9, L10, M1, M2, M3, M4, M5, M6, M7, M8, M9, M10, N1, N2, N9, N10, P1, P2, P9, P10 | | Mechanical. These balls are for mechanical placement, and are not connected internally. | |

Notes: 1. The following abbreviations are used: PCI – PCI 3.3V compatible input, ST - Schmidt Trigger input, OD - Open drain output, R8 - Nominal 22K pull-up resistor, enabled only for 8-bit interface mode (IF_CFG input is 0).

2. For VCCQ≤1.95V, input buffers are defined as LVTTL 1.8V.

## 1.3 Multiplexed Interface

### 1.3.1 Ball Diagram

See Figure 3 for the mDOC H1 ball diagram. To ensure proper device functionality, balls marked RSRVD are reserved for future use and should either be left floating or connected to the recommended signals.

Note: mDOC H1 is designed as a drop-in replacement for mDOC G3 128MB (1Gb), G3 64MB (512Mb), G3 LP 64MB (512Mb), P3 32MB (256Mb), and P3 LP 32MB (256Mb), assuming that the board was designed according to migration guidelines. Refer to the *mDOC G3/P3 to mDOC G3/P3 LP, G4 and H1 migration guide* for further information.

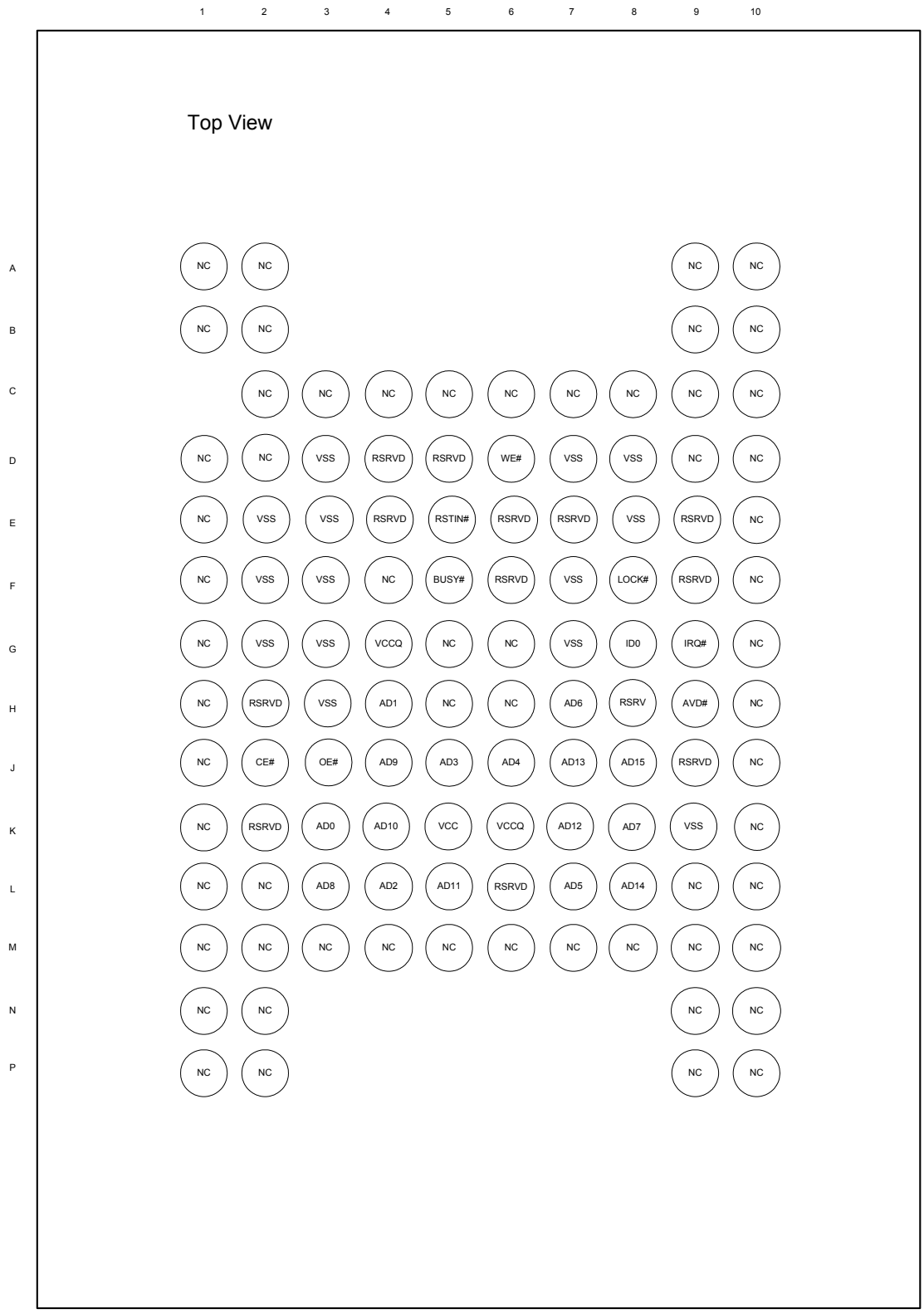



*Figure 3: Multiplexed Interface Ballout for mDOC H1 12x18 FBGA Package*

### 1.3.2 System Interface

See Figure 4 for a simplified I/O diagram of mDOC H1 multiplexed interface.

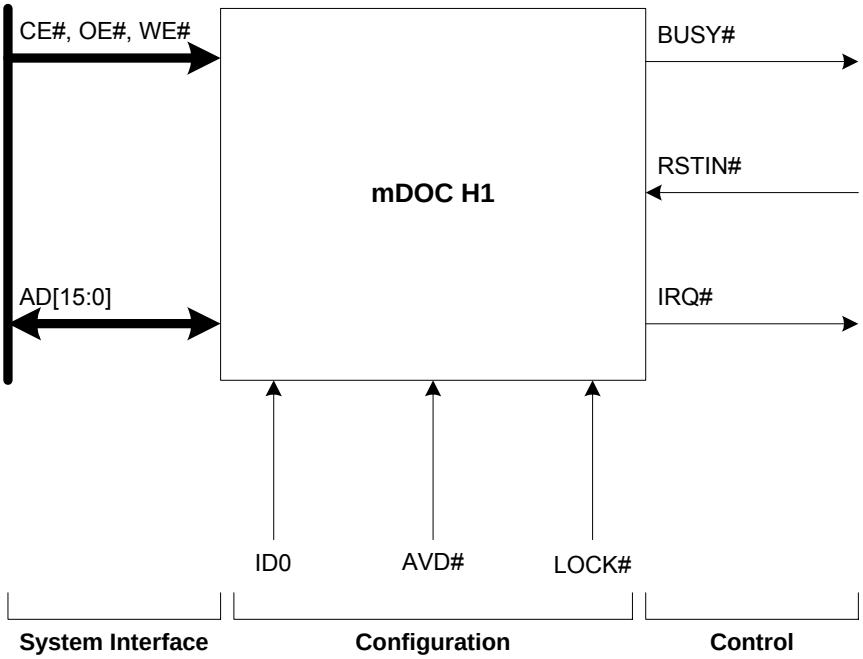



*Figure 4: Multiplexed Interface Simplified I/O Diagram for mDOC H1*

### 1.3.3 Signal Description

*Table 2: Multiplexed Interface Signal Descriptions for mDOC H1 12x18 FBGA Package*

| Signal | Ball No. | Input Type[1,2] | Description | Signal Type |
|---|---|---|---|---|
| **System Interface** | | | | |
| AD[15:12]<br>AD[11:8]<br>AD[7:4]<br>AD[3:0] | J8, L8, J7, K7<br>L5, K4, J4, L3<br>K8, H7, L7, J6,<br>J5, L4, H4, K3 | PCI | Multiplexed bus. Address and data signals. | Input/ Output |
| CE# | J2 | ST | Chip Enable, active low. | Input |
| OE# | J3 | ST | Write Enable, active low. | Input |
| WE# | D6 | ST | Output Enable, active low. | Input |
| **Configuration** | | | | |
| AVD# | H9 | ST | Set multiplexed interface. | Input |
| ID0 | G8 | ST | Identification. Configuration control to support up to two chips cascaded in the same memory window.<br>Chip 1 = ID0 = VSS, must be used for single-chip configuration.<br>Chip 2 = ID0 = VCCQ | Input |
| LOCK# | F8 | ST | Lock, active low. When active, provides full hardware data protection of selected partitions. | Input |
| **Control** | | | | |
| BUSY# | F5 | OD | Open drain output held asserted until download process has completed.<br>A 10 KΩ pull-up resistor is recommended even if this signal is not used. | Output |
| RSTIN# | E5 | ST | Reset, active low. | Input |
| IRQ# | G9 | OD | Open drain interrupt request output.<br>A 10 KΩ pull-up resistor is recommended even if this signal is not used. | Output |
| **Power** | | | | |
| VCC | K5 | - | +3.3V supply for core logic. Requires a 10 nF and 0.1 µF capacitor. | Supply |
| VCCQ | K6, G4 | - | I/O power supply. Sets the logic **1** voltage level range of I/O balls. VCCQ may be either 2.5V to 3.6V or 1.65V to 1.95V. Requires a 10 nF and 0.1 µF capacitor. | Supply |
| VSS | D7, D8, D3, E2,<br>E3, E8, F2, F3,<br>F7,G2, G3, G7,<br>H3, K9 | - | Ground. All VSS balls must be connected. | Supply |

| Signal | Ball No. | Input Type[1,2] | Description | Signal Type |
|---|---|---|---|---|
| **Other** | | | | |
| RSRVD | H8 | - | Reserved signal.<br>Not connected internally. For forward compatibility with future products, it is recommended to connect this ball as DMA request (DMARQ#). | |
| | K2, E7 | - | Reserved signal. Not connected internally.<br>Note: Future mDOC FBGA 12x18 devices will use this signal. It is therefore recommended to connect this ball to GND via a 1 µF capacitor to guarantee forward compatibility. | |
| | L6 | - | Reserved signal.<br>Not connected internally. For forward compatibility with future products, it is recommended to connect this ball as DMA request (DMARQ#). | |
| | H2 | | Reserved. In order to maintain backwards compatibility connect ball H2 to GND via 10KΩ pull-down resistor. | |
| | D4, D5, E4, E6, E9, F6, F9, H8, J8, L6, | - | Reserved. All reserved signals are not connected internally and must be left floating to guarantee forward compatibility with future products. | |
| NC | A1, A2, A9, A10, B1, B2, B9, B10, C2, C3, C4, C5, C6, C7, C8, C9, C10, D1, D2, D9, D10, E1, E10, F1, F4, F10, G1, G5, G6, G10, H1, H5, H6, H10, J1, J10, K1, K10, L1, L2, L9, L10, M1, M2, M3, M4, M5, M6, M7, M8, M9, M10, N1, N2, N9, N10, P1, P2, P9, P10 | | Mechanical. These balls are for mechanical placement, and are not connected internally. | |

Notes: 1. The following abbreviations are used: PCI – PCI 3.3V compatible input, ST - Schmidt Trigger input, OD - Open drain output.

2. For VCCQ≤1.95, input buffers are defined as LVTTL 1.8V.

# 2. THEORY OF OPERATION

## 2.1 Overview

mDOC H1 consists of the following major functional blocks, as shown in Figure 5.

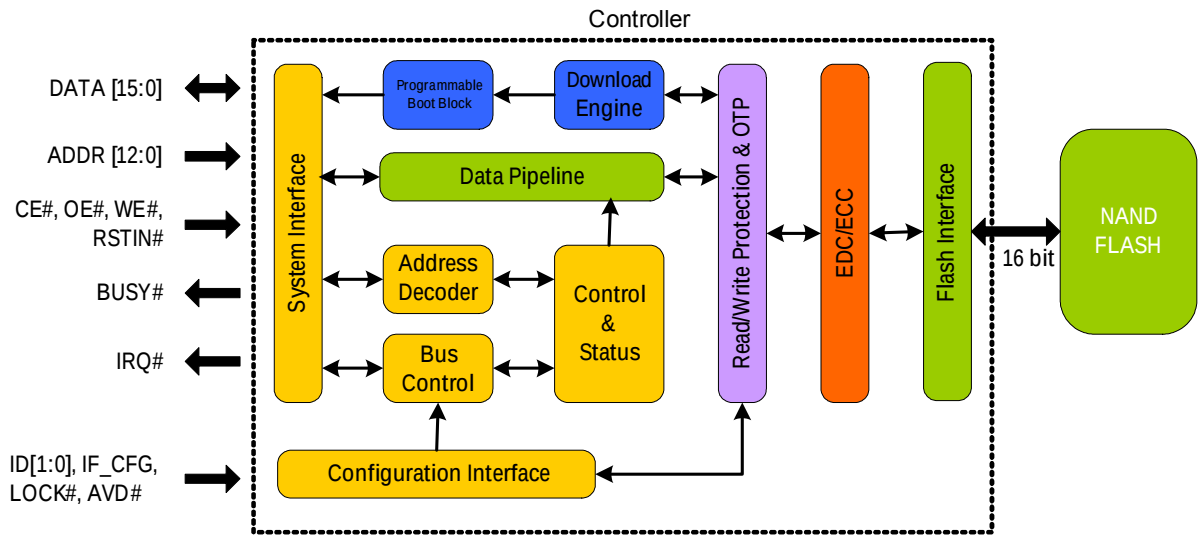



*Figure 5: mDOC H1 Simplified Block Diagram, Standard Interface*

These components are described briefly below, and in more detail in the sections that follow.

- **System Interface** for the host interface.
- **Configuration Interface** for configuring mDOC H1 to operate in 8-bit, 16-bit mode, cascaded configuration and hardware read/write protection.
- **Read/Write Protection & OTP** for advanced data/code security and protection.
- **Programmable Boot Block** with XIP functionality enhanced with a Download Engine (DE) for system initialization capability.
- **Error Detection** and **Error Correction Code** (EDC/ECC) for on-the-fly error handling.
- **Data Pipeline** through which the data flows from the system to the NAND flash arrays.
- **Control & Status** block that contains registers responsible for transferring the address, data and control information between the TrueFFS driver and the flash media.
- **Flash Interface** that interfaces to NAND flash.
- **Bus Control** for translating the host bus address, and data and control signals into valid NAND flash signals.
- **Address Decoder** to enable the relevant unit inside the mDOC controller, according to the address range received from the system interface.

## 2.2 System Interface

### 2.2.1 Standard (NOR-Like) Interface

The system interface block provides an easy-to-integrate NOR-like (also SRAM and EEPROM-like) interface to mDOC H1, enabling it to interface with various CPU interfaces, such as a local bus, NOR interface, SRAM interface, EEPROM interface, or any other compatible interface. In addition, the interface enables direct access to the Programmable Boot Block to permit XIP (Execute-In-Place) functionality during system initialization.

A 13-bit wide address bus enables access to the mDOC H1 8KB memory window (as shown in Section 4.6).

The Chip Enable (CE#), Output Enable (OE#) and Write Enable (WE#) signals trigger read and write cycles. A write cycle occurs while both the CE# and the WE# inputs are asserted. Similarly, a read cycle occurs while both the CE# and OE# inputs are asserted. Note that mDOC H1 does not require a clock signal. It features a unique analog static design, optimized for minimal power consumption. The CE#, WE# and OE# signals trigger the controller (e.g., system interface block, bus control and data pipeline) and flash access.

The Reset In (RSTIN#) and Busy (BUSY#) control signals are used in the reset phase.

The Interrupt Request (IRQ#) signal can be used when long I/O operations, such as Block Erase, delay the CPU. The signal is also asserted when a Data Protection violation has occurred. This signal frees the CPU to run other tasks, continuing read/write operations with mDOC H1 only after the IRQ# signal has been asserted and an interrupt handling routine (implemented in the OS) has been called to return control to the TrueFFS driver.

### 2.2.2 Multiplexed Interface

In this configuration, the address and data signals are multiplexed. The AVD# input is driven by the host AVD# signal, and the D[15:0] balls, used for both address inputs and data, are connected to the host AD[15:0] bus. While AVD# is asserted, the host drives AD[11:0] with bits [12:1] of the address. Host signals AD[15:12] are not significant during this part of the cycle.

This interface is automatically used when a falling edge is detected on AVD#. This edge must occur after RSTIN# is negated and before the first read or write cycle to the controller.

## 2.3 Configuration Interface

The Configuration Interface block enables the designer to configure mDOC H1 to operate in different modes. The ID[1:0] signals are used in a cascaded configuration (refer to Section 7.6) and the IF_CFG signal is used to configure 8/16-bit access (refer to Section 7.4.2).

## 2.4 Protection and Security-Enabling Features

The Protection and Security-Enabling block, consisting of read/write protection, UID, and an OTP area, enables advanced data and code security and content protection. Located on the main route of traffic between the host and the flash, this block monitors and controls all data and code transactions to and from mDOC H1.

### 2.4.1 Read/Write Protection

Data and code protection is implemented through a Protection State Machine (PSM). The user can configure an independently programmable area of the flash memory as read protected, write protected, or read/write protected.

A protected partition may be protected by either/both of these hardware mechanisms:

- 64-bit protection key
- Hard-wired LOCK# signal

If the Lock option is enabled (by means of software) and the LOCK# signal is asserted, the protected partition has an additional hardware lock that prevents read/write access to the partition, even with the use of the correct protection key.

The size and protection attributes of the protected partition are defined during the media-formatting stage.

In the event of an attempt to bypass the protection mechanism, illegally modify the protection key, or in any way sabotage the configuration parameters, the entire mDOC H1 becomes both read and write protected, and is completely inaccessible.

For further information on hardware protection, please refer to the *TrueFFS Software Development Kit (SDK)* developer guide.

### 2.4.2 Unique Identification (UID) Number

Each mDOC H1 is assigned a 16-byte UID number. Burned onto the flash during production, the UID cannot be altered and is unique worldwide. The UID is essential in security-related applications, and can be used to identify end-user products in order to fight fraudulent duplication by imitators.

The UID on mDOC H1 eliminates the need for an additional on-board ID device, such as a dedicated EEPROM.

### 2.4.3 One-Time Programmable (OTP) Area

The 6KB OTP area is user programmable for complete customization. The user can write to this area once, after which it is automatically and permanently locked. After it is locked, the OTP area becomes read only, just like a ROM device.

Typically, the OTP area is used to store customer and product information such as: product ID, software version, production data, customer ID and tracking information.

### 2.4.4 Sticky Lock (SLOCK)

The boot partition can be locked automatically by hardware after the boot phase is completed and the device is in Normal mode. This is done by setting the Sticky Lock (SLOCK) bit in the Output Control register to 1. This has the same effect as asserting the LOCK# signal. Once set, SLOCK can only be cleared by asserting the RSTIN# input. Like the LOCK# input, assertion of this bit prevents the protection key from disabling the protection for a given partition. There is no need to mount the partition before setting this bit.

This feature can be useful when the boot code in the boot partition must be read/write protected. Upon power-up, the boot code must be unprotected so the CPU can boot directly from mDOC. At the end of the boot process, protection can be set until the next power-up or reset.

## 2.5 Programmable Boot Block with eXecute In Place (XIP) Functionality

The Programmable Boot Block with XIP functionality enables mDOC H1 to act as a boot device in addition to performing flash disk data storage functions. This eliminates the need for expensive, legacy NOR flash or any other boot device on the motherboard.

The Programmable Boot Block on mDOC H1 is 1KB in size. The Download Engine (DE), described in the next section, expands the functionality of this block by copying the boot code from the flash into the boot block.

Note: When more than one mDOC H1 device is cascaded, a maximum boot block of 2KB is available. The Programmable Boot Block of each device is mapped to a unique address space.

## 2.6 Download Engine (DE)

Upon power-up after the rising edge of RSTIN#, the DE automatically downloads the Initial Program Loader (IPL) to the Programmable Boot Block. The IPL is responsible for starting the booting process. The download process is quick, and is designed so that when the CPU accesses mDOC H1 for code execution, the IPL code is already located in the Programmable Boot Block. During the download process, mDOC H1 does not respond to read or write accesses. Host systems must therefore observe the requirements described in Section 8.3.6.

In addition, the DE downloads the data protection rules from the flash to the Protection State Machines (PSM), so that mDOC H1 is secure and protected from the first moment it is active.

During the download process, mDOC H1 asserts (low) the BUSY# signal to indicate to the system that it is not yet ready to be accessed. Once BUSY# is negated, the system can access mDOC H1.

A failsafe mechanism prevents improper initialization due to a faulty VCC or invalid assertion of the RSTIN# input. Another failsafe mechanism is designed to overcome possible NAND flash data errors. It prevents internal registers from powering up in a state that bypasses the intended data protection. In addition, any attempt to sabotage the data structures causes the entire mDOC to become both read and write protected, and completely inaccessible.

## 2.7 Error Detection Code/Error Correction Code (EDC/ECC)

Because NAND-based flash is prone to errors, it requires unique error-handling capability. msystems' x2 technology implements 4-bit Error Detection Code/Error Correction Code (EDC/ECC), based on a combination of Bose, Chaudhuri and Hocquenghem (BCH) and Hamming code algorithms. Error Detection Code (EDC) is implemented in hardware to optimize performance, while Error Correction Code (ECC) is performed in software, when required, to save silicon costs.

Every time 512-bytes are written, additional parity bits are calculated and written to the flash. Every time data is read from the flash, the parity bits are read and used to calculate error locations.

# 3. HARDWARE PROTECTION

## 3.1 Method of Operation

mDOC H1 enables the user to define a partition that is protected (in hardware) against any combination of read or write operations. The protected area can be configured as read protected or write protected, and is protected by a protection key (i.e. password) defined by the user.

The size and protection attributes (protection key, read, write, changeable, lock) of the protected partition are defined during the media formatting stage (DFORMAT utility or the format function in the TrueFFS SDK).

In order to enable or disable read/write protection, the protection key (i.e., password) must be used, as follows:

- Insert the protection key to disable read/write protection.
- Remove the protection key to enable read/write protection.

mDOC H1 has an additional hardware safety measure. If the Lock option is enabled (by means of software) and the LOCK# signal is asserted, the protected partition has an additional hardware lock that prevents read/write access to the partition, even with the use of the correct protection key. It is possible to set the Lock protection for one session only; that is, until the next power-up or reset. This Sticky Lock feature can be useful when the boot code in the boot partition must be read/write protected. Upon power-up, the boot code must be unprotected so the CPU can run it directly from mDOC H1. At the end of the boot process, protection can be set until the next power-up or reset.

Setting the Sticky Lock (SLOCK) bit in the Output Control register to 1 has the same effect as asserting the LOCK# signal. Once set, SLOCK can only be cleared by asserting the RSTIN# input. Like the LOCK# input, the assertion of this bit prevents the protection key from disabling the protection for the partition. For more information, see Section 2.4.4

The only way to read from or write to a protected partition is to insert the key (even DFORMAT cannot remove the protection without inserting the key first). This is also true for modifying its attributes (protection key, read, write and lock). Read/write protection is reactivated (the key is automatically removed) in each of the following events:

- Power-down
- Change of any protection attribute
- Write operation to the IPL area
- Removal of the protection key.

For further information on hardware protection, please refer to the *TrueFFS Software Development Kit (SDK)* developer guide.

# 4. MODES OF OPERATION

mDOC H1 operates in one of three basic modes:

- Normal mode
- Standby Mode
- Reset mode
- Deep Power-Down mode

The current mode of the chip can always be determined by reading the mDOC Control register. Mode changes can occur due to any of the following events:

- Assertion of the RSTIN# signal sets the device in Reset mode.
- A valid write sequence to mDOC H1 sets the device in Normal mode. This is done automatically by the TrueFFS driver on power-up (reset sequence end).
- Switching back from Normal mode to Reset mode can be done by a valid write sequence to mDOC H1, or by triggering the boot detector circuitry (via a soft reset).
- Deep Power-Down - A valid write sequence, initiated by software, sets the device from Normal mode to Deep Power-Down mode. Accessing the NOP register and waiting for 3uS set the device back to Normal mode.

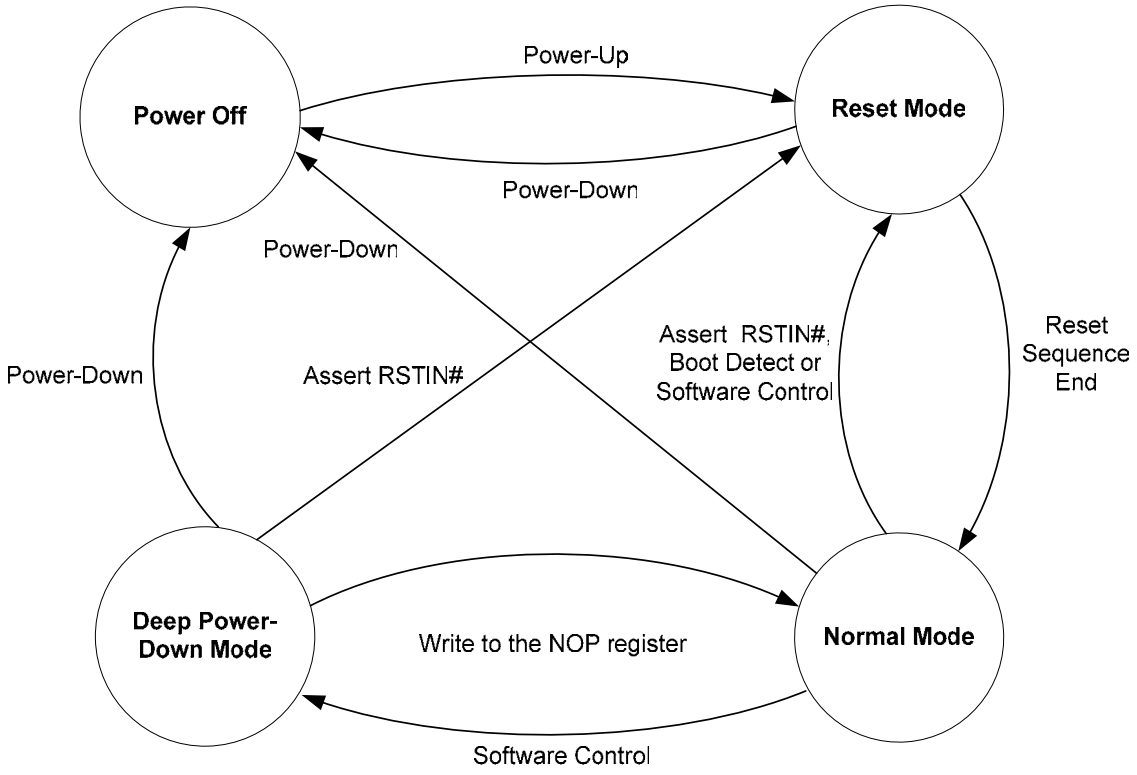



*Figure 6: Operating Modes and Related Events*

## 4.1 Normal Mode

This is the mode in which standard operations involving the flash memory are performed. Normal mode is entered when a valid write sequence is sent to the mDOC Control register and Control Confirmation register. A write cycle occurs when both the CE# and WE# inputs are asserted. Similarly, a read cycle occurs when both the CE# and OE# inputs are asserted. Because the flash controller generates its internal clock from these CPU bus signals and some read operations return volatile data, it is essential that the timing requirements specified in Section 8.3 be met. It is also essential that read and write cycles not be interrupted by glitches or ringing on the CE#, WE#, and OE# inputs.

## 4.2 Reset Mode

In Reset mode, mDOC H1 ignores all write cycles, except for those to the mDOC Control register and Control Confirmation register. All register read cycles return a value of 00H.

Before attempting to perform any operation, the device must be in Normal mode.

## 4.3 Standby Mode

While in Standby mode mDOC H1 power consumption is reduced by disconnecting the internal clock from the core.

mDOC H1 enters Standby mode after not being accessed during 750ns.

mDOC H1 exits Standby mode and returns to Normal operation mode upon the first host access.

## 4.4 Deep Power-Down Mode

While in Deep Power-Down mode, mDOC H1's quiescent power dissipation is reduced by disabling internal high current consumers (e.g. voltage regulators, oscillator, etc.).

To enter Deep Power-Down mode, a proper sequence must be written to the mDOC H1 Control registers and the following signals negated (logical '1'): CE#, WE#, OE#. All other inputs should be VSS or VCCQ.

In Deep Power-Down mode, write cycles have no affect and read cycles return indeterminate data (mDOC H1 does not drive the data bus). Entering Deep Power-Down mode and then returning to the previous mode does not affect the value of any register.

To exit Deep Power-Down mode, write to the NOP register. The device will exit DPD mode and enter to Normal mode after 3uS.

Applications that use mDOC H1 as a boot device must ensure that the device is not in Deep Power-Down mode before reading the Boot vector/instructions. This can be done by pulsing RSTIN# to the asserted state and waiting for the BUSY# output to be negated.

## 4.5 TrueFFS Technology

### 4.5.1 General Description

msystems' patented TrueFFS technology was designed to maximize the benefits of flash memory while overcoming inherent flash limitations that would otherwise reduce its performance, reliability and lifetime. TrueFFS emulates a hard disk, making it completely transparent to the OS. In addition, since it operates under the OS file system layer (see Figure 7), it is completely transparent to the application.

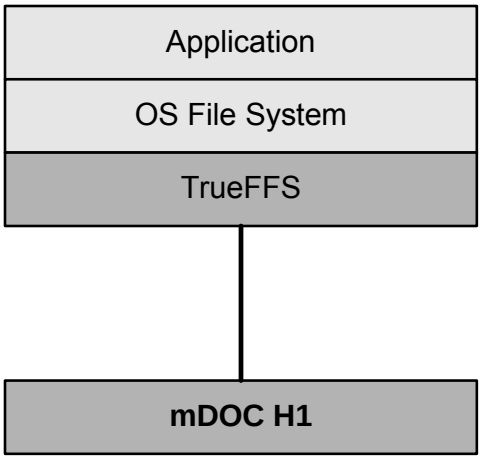



*Figure 7: TrueFFS Location in System Hierarchy*

TrueFFS technology support includes:

- TrueFFS Software Development Kit (TrueFFS SDK)
- Boot Software Development Kit (BDK)
- Support for all major CPUs, including 8, 16 and 32-bit bus architectures.

TrueFFS technology features:

- Block device API
- Flash file system management
- Automatic bad-block management
- Dynamic virtual mapping
- Dynamic and static wear-leveling
- Power failure management
- Implementation of NAND-tailored EDC/ECC
- Performance optimization
- Compatibility with all mDOC products

### 4.5.2 Built-In Operating System Support

The TrueFFS driver is integrated into all major OSs, including Symbian, Palm OS, Windows Mobile 5.0, Windows CE, Linux (various kernels), Nucleus, and others. Please contact your assigned FAE at the regional msystems office for the latest driver version.

### 4.5.3 TrueFFS Software Development Kit (SDK)

The basic *TrueFFS Software Development Kit (SDK)* developer guide provides the source code for the TrueFFS driver. It can be used in an OS-less environment or when special customization of the driver is required for proprietary OSs.

When using mDOC H1 as the boot replacement device, TrueFFS SDK also incorporates in its source code the boot software that is required for this configuration (this package is also available separately). Please refer to the *mDOC Boot Software Development Kit (BDK)* developer guide for further information on using this software package.

Note: mDOC H1 is supported by TrueFFS 6.3 and above.

### 4.5.4 File Management

TrueFFS accesses the flash memory within mDOC H1 through an 8KB window in the CPU memory space. TrueFFS provides block device API by using standard file system calls, identical to those used by a mechanical hard disk, to enable reading from and writing to any sector on mDOC H1. This makes mDOC H1 compatible with any file system and file system utilities, such as diagnostic tools and applications.

### 4.5.5 Bad-Block Management

Since NAND flash is an imperfect storage media, it can contain bad blocks that cannot be used for storage. TrueFFS automatically detects and maps out bad blocks upon system initialization, ensuring that they are not used for storage. This management process is completely transparent to the user, who is unaware of the existence and location of bad blocks, while remaining confident of the integrity of data stored.

### 4.5.6 Wear-Leveling

Flash memory can be erased a limited number of times. This number is called the *erase cycle limit*, or *write endurance limit*, and is defined by the flash array vendor. The erase cycle limit applies to each individual erase block in the flash device.

In a typical application, and especially if a file system is used, specific pages are constantly updated (e.g., the page/s that contain the FAT, registry, etc.). Without any special handling, these pages would wear out more rapidly than other pages, reducing the lifetime of the entire flash.

To overcome this inherent deficiency, TrueFFS uses msystems' patented wear-leveling algorithm. This wear-leveling algorithm ensures that consecutive writes of a specific sector are not written physically to the same page in the flash. This spreads flash media usage evenly across all pages, thereby maximizing flash lifetime.

#### Dynamic Wear-Leveling

TrueFFS uses statistical allocation to perform dynamic wear-leveling on newly written data. This minimizes the number of erase cycles per block. Because a block erase is the most time-consuming

operation, dynamic wear-leveling has a major impact on overall performance. This impact cannot be noticed during the first write to flash (since there is no need to erase blocks beforehand), but it is more and more noticeable as the flash media becomes full.

#### Static Wear-Leveling

Areas on the flash media may contain static files, characterized by blocks of data that remain unchanged for very long periods of time, or even for the whole device lifetime. If wear-leveling were only applied on newly written pages, static areas would never be recycled. This limited application of wear-leveling would lower life expectancy significantly in cases where flash memory contains large static areas. To overcome this problem, TrueFFS forces data transfer in static areas as well as in dynamic areas, thereby applying wear-leveling to the entire media.

### 4.5.7 Power Failure Management

TrueFFS uses algorithms based on “erase after write” instead of "erase before write" to ensure data integrity during normal operation and in the event of a power failure. Used areas are reclaimed for erasing and writing the flash management information into them only *after* an operation is complete. This procedure serves as a check on data integrity.

The “erase after write” algorithm is also used to update and store mapping information on the flash memory. This keeps the mapping information coherent even during power failures. The only mapping information held in RAM is a table pointing to the location of the actual mapping information. This table is reconstructed during power-up or after reset from the information stored in the flash memory.

To prevent data from being lost or corrupted, TrueFFS uses the following mechanisms:

- When writing, copying, or erasing the flash device, the data format remains valid at all intermediate stages. Previous data is never erased until the operation has been completed and the new data has been verified.
- A data sector cannot exist in a partially written state. The operation is either successfully completed, in which case the new sector contents are valid, or the operation has not yet been completed or has failed, in which case the old sector contents remain valid.

### 4.5.8 Error Detection/Correction

TrueFFS implements a unique NAND-tailored Error Correction Code (ECC) algorithm to ensure data reliability. Refer to Section 2.7 for further information on the EDC/ECC mechanism.

### 4.5.9 Special Features Through I/O Control (IOCTL) Mechanism

In addition to standard storage device functionality, the TrueFFS driver provides extended functionality. This functionality goes beyond simple data storage capabilities to include features such as: formatting the media, read/write protection, boot partition(s) access, flash de-fragmentation and other options. This unique functionality is available in all TrueFFS-based drivers through the standard I/O control command of the native file system.

### 4.5.10 Compatibility

mDOC H1 requires TrueFFS driver 6.3.0 or higher. Since this driver does not support some of mDOC products, migrating from other than mDOC H1 to mDOC H1 requires changing the TrueFFS driver.

When using different drivers (e.g. TrueFFS SDK, BootSDK, Windows Mobile, etc.) to access mDOC H1, verify that all software is based on the same code base version. It is also important to use only tools (e.g. DFORMAT, DINFO, DIMAGE, etc.) from the same version as the TrueFFS drivers used in the application. Failure to do so may lead to unexpected results, such as lost or corrupted data. The driver version can be verified by the sign-on messages displayed, or by the version information presented by the driver or tool.

## 4.6 8KB Memory Window

TrueFFS utilizes an 8KB memory window in the CPU address space, consisting of four 2KB sections as depicted in Figure 8. When in Reset mode, read cycles from sections 1 and 2 always return the value 00H to create a fixed and known checksum. When in Normal mode, these two sections are used for the internal registers. The 1KB Programmable Boot Block is in section 0 and section 3, to support systems that search for a checksum at the boot stage both from the top and bottom of memory. The 1KB Programmable Boot Block is aliased two times in both section 0 and section 3 for redundancy reasons. The addresses described here are relative to the absolute starting address of the 8KB memory window.

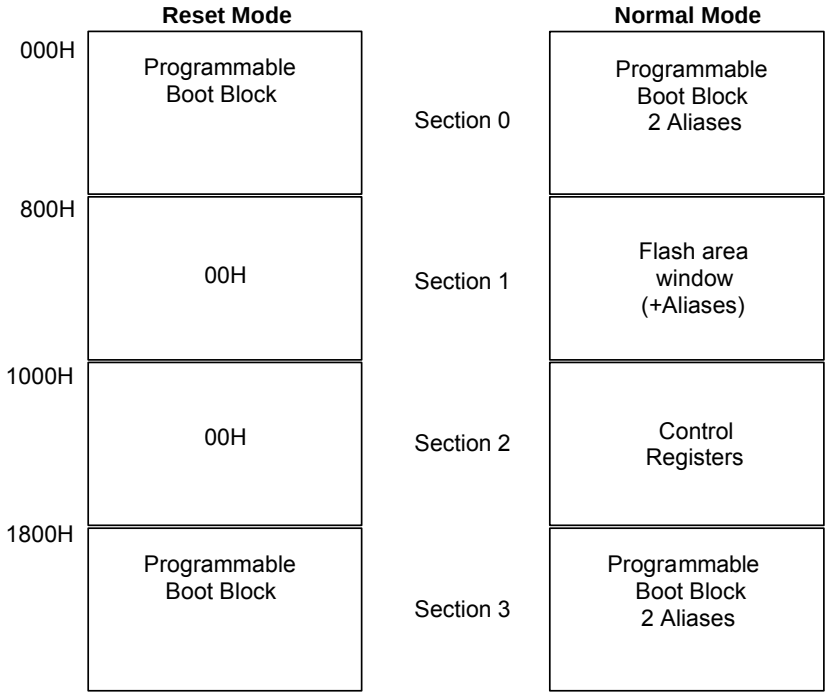



*Figure 8: mDOC H1 Memory Map*

# 5. REGISTER DESCRIPTIONS

This section describes various mDOC H1 registers and their functions, as listed in Table 3. Most mDOC H1 registers are 8-bit, unless otherwise denoted as 16-bit.

*Table 3: mDOC H1 Registers*

| Address (Hex) | Register Name |
|---|---|
| 1002 | No Operation (NOP) |
| 1000/1074 | Chip Identification[0:1] |
| 1008 | Endian Control |
| 100C | mDOC Control |
| 1072 | mDOC Control Confirmation |
| 100A | Device ID Select |
| 100E | Configuration |
| 1010 | Interrupt Control |
| 1014 | Output Control |

## 5.1 Definition of Terms

The following abbreviations and terms are used within this section:

RFU — Reserved for future use. This bit is undefined during a read cycle and "don't care" during a write cycle.

RFU_0 — Reserved for future use; when read, this bit always returns the value 0; when written, software should ensure that this bit is always cleared to 0.

RFU_1 — Reserved for future use; when read, this bit always returns the value 1; when written, software should ensure that this bit is always set to 1.

Reset Value — Refers to the value immediately present after exiting from Reset mode to Normal mode.

## 5.2 Reset Values

All registers return 00H while in Reset mode. The Reset value written in the register description is the register value after exiting Reset mode and entering Normal mode. Some register contents are undefined at that time (N/A).

## 5.3 No Operation (NOP) Register

Description: An access to this 16-bit register results in no change in the device.. To aid in code readability and documentation, software should access this register when performing cycles intended to create a time delay.

Address (hex): 1002H

Type: Write

Reset Value: None

## 5.4 Chip Identification (ID) Register and Chip Identification Confirmation Register [0:1]

Description: These two 8-bit registers are used to identify the mDOC device residing on the host platform.

Chip Identification Confirmation Register always returns the complement value of Chip Identification Register.

Address (hex): 1000H/1074H

Type: Read only

Reset Value: Chip Identification Register[0]: AAH

Chip Identification Confirmation Register[1]: 55H

## 5.5 Endian Control Register

Description: This 16-bit register is used to control the swapping of the low and high data bytes when reading or writing with a 16-bit host. This provides an Endian-independent method of enabling/disabling the byte swap feature.

Note: Hosts that support 8-bit access only do not need to write to this register. Enabling the byte swap is required only if the byte lanes are connected incorrectly between the mDOC and the Big Endian CPU.

Address (hex): 1008H

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| **Read/Write** | R | | | | | | | R/W |
| **Description** | RFU_0 | | | | | | | SWAPL |
| **Reset Value** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| | Bit 15 | Bit 14 | Bit 13 | Bit 12 | Bit 11 | Bit 10 | Bit 9 | Bit 8 |
|---|---|---|---|---|---|---|---|---|
| **Read/Write** | R | | | | | | | R/W |
| **Description** | RFU_0 | | | | | | | SWAPH |
| **Reset Value** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit No. | Description |
|---|---|
| 0 | SWAPL (Swap Low Byte): This bit must be set to enable byte swapping. If the bit is cleared, then byte swapping is disabled. |
| 7-1 | Reserved for future use. |
| 8 | SWAPH (Swap High Byte): This bit must be set to enable byte swapping. If the bit is cleared, then byte swapping is disabled. |
| 15-9 | Reserved for future use. |

## 5.6 mDOC Control Register/Control Confirmation Register

Description: These two registers are identical and contain information about the mDOC H1 operational mode. After writing the required value to the mDOC H1 Control register, the complement of that data byte must also be written to the Control Confirmation register. The two writes cycles must not be separated by any other read or write cycles to the mDOC H1 memory space, except for reads from the Programmable Boot Block space.

Address (hex): 100CH/1072H

| Bit No | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| **Read/Write** | R | R | R/W | R/W | R/W | R/W | R/W | R/W |
| | | | | | | | | |
| **Description** | RFU_0 | | RFU | RST_LAT | BDET | MDWREN | Mode[1:0] | |
| **Reset Value** | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |

Note: The mDOC H1 Control Confirmation register is write only

| Bit No. | Description |
|---|---|
| 1-0 | Mode. These bits select the mode of operation, as follows:<br>00: Reset<br>01: Normal<br>10: Deep Power-Down |
| 2 | MDWREN (Mode Write Enable). The value 1 must be written to this bit when changing the mode of operation. It always returns 0 when read. |
| 3 | BDET (Boot Detect). This bit is set whenever the device has entered Reset mode as a result of the Boot Detector triggering. It is cleared by writing a 1 to this bit. |
| 4 | RST_LAT (Reset Latch). This bit is set whenever the device has entered the Reset mode as a result of the RSTIN# input signal being asserted or the internal voltage detector triggering. It is cleared by writing a 1 to this bit. |
| 7-5 | Reserved for future use. |

## 5.7 Device ID Select Register

Description: In a cascaded configuration, this register controls which device provides the register space. The value of bit ID[0:1] is compared to the value of the ID configuration input balls. The device whose ID input matches the value of bit ID[0:1] responds to read and write cycles.

Address (hex): 100AH

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| **Read/Write** | R | | | | | | R/W | |
| **Description** | RFU_0 | | | | | | ID[1:0] | |
| **Reset Value** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit No. | Description |
|---|---|
| 1-0 | ID[1:0] (Identification). The device whose ID input signal match the value of bits ID[1:0] responds to read and write cycles to register space.<br>Note: Since mDOC H1 cascading supports two devices the possible values for ID[1:0] are 00 and 01. |
| 2-7 | Reserved for future use. |

## 5.8 Configuration Register

Description: This register indicates the current configuration of mDOC H1. Unless otherwise noted, the bits are reset only by a hardware reset, and not upon boot detection or any other entry to Reset mode.

Address (hex): 100EH

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| **Read/Write** | R | | R/W | | R | | | |
| **Description** | IF_CFG | RFU_0 | MAX_ID | | RFU_1 | RFU_0 | RFU_0 | BUS_CONF |
| **Reset Value** | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |

| Bit No. | Description |
|---|---|
| 0 | BUS_CONF (Bus configuration). Indicates the bus type the mDOC H1 is connected to.<br>0: SRAM mode<br>1: Mux mode |
| 1-3, 6 | Reserved for future use. |
| 4-5 | MAX_ID (Maximum Device ID). Indicates the number of currently cascaded mDOC H1 devices when multiple devices are used in a cascaded configuration, using the ID[1:0] input. It can be programmed by the host to the highest ID value that is found by software in order to map all available boot blocks into usable address spaces. |
| 7 | IF_CFG (Interface Configuration). Indicates the state of the IF_CFG input signal.<br>0: Host data bus width is 8bit.<br>1: Host data bus width is 16bit. |

## 5.9 Interrupt Control Register

Description: This register controls how interrupts are generated by mDOC H1, and indicates which of the following three sources has asserted an interrupt:

- Flash array is ready
- Data protection violation
- Reading or writing more flash data than was expected

Address (hex): 1010H

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| **Read/Write** | R/W | | | | | | | |
| **Description** | IRQ_EN | IRQ_FE | IRQ_P | IRQ_F | EDGE | FFER_T | PROT_T | FRDY_T |
| **Reset Value** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

| Bit No. | Description |
|---|---|
| 0 | Flash Ready Trigger. This bit determines if an interrupt will be generated when the flash array of mDOC H1 is ready, as follows:<br>0: Interrupts are disabled – Holds the IRQ# output in the negated state.<br>1: Interrupt when flash array is ready. |
| 1 | Protection Trigger. When set, an interrupt will be generated upon a data protection violation. |
| 2 | FIFO Error Trigger. When set, an interrupt will be generated upon a FIFO error: FIFO overflow or underflow during the data transfer. |
| 3 | EDGE (Edge-sensitive interrupt)<br>0: Specifies level-sensitive interrupts in which the IRQ# output remains asserted until the interrupt is cleared.<br>1: Specifies edge-sensitive interrupts in which the IRQ# output pulses low. |
| 4 | Interrupt Request when flash array is ready. Indicates that the IRQ# output has been asserted due to an indication that the flash array is ready. Writing 1 to this bit clears its value, negates the IRQ# output and permits subsequent interrupts to occur. |
| 5 | Interrupt Request on Protection Violation. Indicates that the IRQ# output has been asserted due to a data protection violation. Writing a 1 to this bit clears its value, negates the IRQ# output and permits subsequent interrupts to occur. |
| 6 | Interrupt Request on FIFO error. Indicates that FIFO error has happened (FIFO overflow or underflow) during the data transfer.<br>Writing a 1 to this bit clears its value, negates the IRQ# output and permits subsequent interrupts to occur. |
| 7 | Global Interrupt Enable<br>0: All interrupt sources are disabled and IRQ# output is negated.<br>1: Interrupt sources enabled and IRQ# is asserted when any enabled interrupt request condition occurs. |

## 5.10 Output Control Register

Description: This register controls the behavior of certain output signals. This register is reset by a hardware reset, not by entering Reset mode.

Note: When multiple devices are cascaded, writing to this register will affect all devices regardless of the value of the ID[1:0] input.

Address (hex): 1014H

| | Bit 7 | Bit 6 | Bit 5 | Bit 4 | Bit 3 | Bit 2 | Bit 1 | Bit 0 |
|---|---|---|---|---|---|---|---|---|
| **Read/Write** | R/W | | | | | | | |
| **Description** | RFU_0 | | | SLOCK | RFU_0 | RFU_0 | | BSY_EN |
| **Reset Value** | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |

| Bit No. | Description |
|---|---|
| 0 | Busy Enable. Controls the assertion of the BUSY# output during a download initiated by a soft reset.<br>1: Enables the assertion of the BUSY# output<br>0: Disables the assertion of the BUSY# output<br>Upon the assertion of the RSTIN# input, this bit will be set automatically and the BUSY# output signal will be asserted until completion of the download process. |
| 4 | Sticky Lock. This bit is the SW version of LOCK# input and logical LOCK signal is OR between LOCK# input and this bit. This bit may be set by the host but cannot be cleared until reset.<br>0: SLOCK negated<br>1: SLOCK asserted |
| 3, 5-7 | Reserved for future use. |

# 6. BOOTING FROM MDOC H1

## 6.1 Introduction

mDOC H1 can function both as a flash disk and as the system boot device.

If mDOC H1 is configured as a flash disk and as the system boot device, it contains the boot loader, an OS image and a file system. In such a configuration, mDOC H1 can serve as the only non-volatile device on board.

## 6.2 Boot Replacement

In common CPU architecture, the boot code is executed from a boot ROM, and the drivers are usually loaded from the storage device.

When using mDOC H1 as the system boot device, the CPU fetches the first instructions from the mDOC H1 Programmable Boot Block, which contains the IPL. Since in most cases this block cannot hold the entire boot loader, the IPL contains minimum initialization code, after which the Secondary Program Loader (SPL) is copied to RAM from flash. The remainder of the boot loader code then runs from RAM.

The SPL is located in a separate (binary) partition on mDOC H1, and can be hardware protected if required.

### 6.2.1 Asynchronous Boot Mode

During platform initialization, certain CPUs wake up in 32-bit mode and issue instruction fetch cycles continuously. An XScale CPU, for example, initiates a 16-bit read cycle, but after the first word is read, it continues to hold CE# and OE# asserted while it increments the address and reads additional data as a burst.

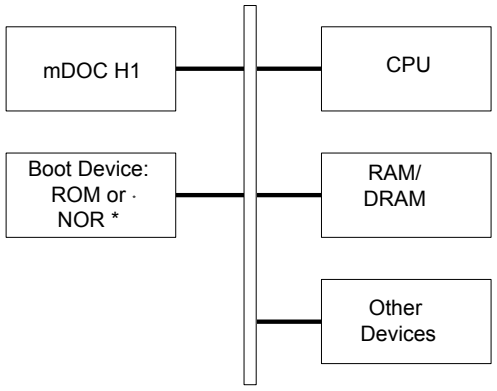
In Asynchronous Boot mode, the CPU can fetch its instruction cycles from the mDOC H1 Programmable Boot Block. After reading from this block and completing boot, mDOC H1 returns to derive its internal clock signal from the CE#, OE#, and WE# inputs. Please refer to Section 8.3 for read timing specifications for Asynchronous Boot mode.

# 7. DESIGN CONSIDERATIONS

## 7.1 General Guidelines

A typical RISC processor memory architecture is shown in Figure 9. It may include the following devices:

- **mDOC H1**: Contains the OS image, applications, registry entries, back-up data, user files and data, etc. It can also be used to perform boot operation, thereby replacing the need for a separate boot device.
- **CPU**: mDOC H1 is compatible with all major CPUs in the mobile phone, Digital TV (DTV) and Digital Still Camera (DSC) markets, including:
  - ARM-based CPUs
  - Texas Instruments OMAP, DBB
  - Intel XScale PXAxxx
  - Emblaze ER4525 application processor
  - SuperH SH
  - FreeScale MX family
  - Analog Devices (ADI) AD652x family.
  - Infineon xGOLD
- **Boot Device**: ROM or NOR flash that contains the boot code required for system initialization, kernel relocation, loading the operating systems and/or other applications and files into the RAM and executing them.
- **RAM/DRAM Memory**: This memory is used for code execution.
- **Other Devices**: A DSP processor, for example, may be used in a RISC architecture for enhanced multimedia support.




* When used as a boot device, mDOC H1 eliminates the need for a dedicated boot ROM/NOR device.

*Figure 9: Typical System Architecture Using mDOC H1*

## 7.2 Standard NOR-Like Interface

mDOC H1 uses a NOR-like interface that can easily be connected to any microprocessor bus. With a standard interface, it requires 13 address lines, 8 data lines and basic memory control signals (CE#, OE#, WE#), as shown in Figure 10 below. Typically, mDOC H1 can be mapped to any free 8KB memory space.

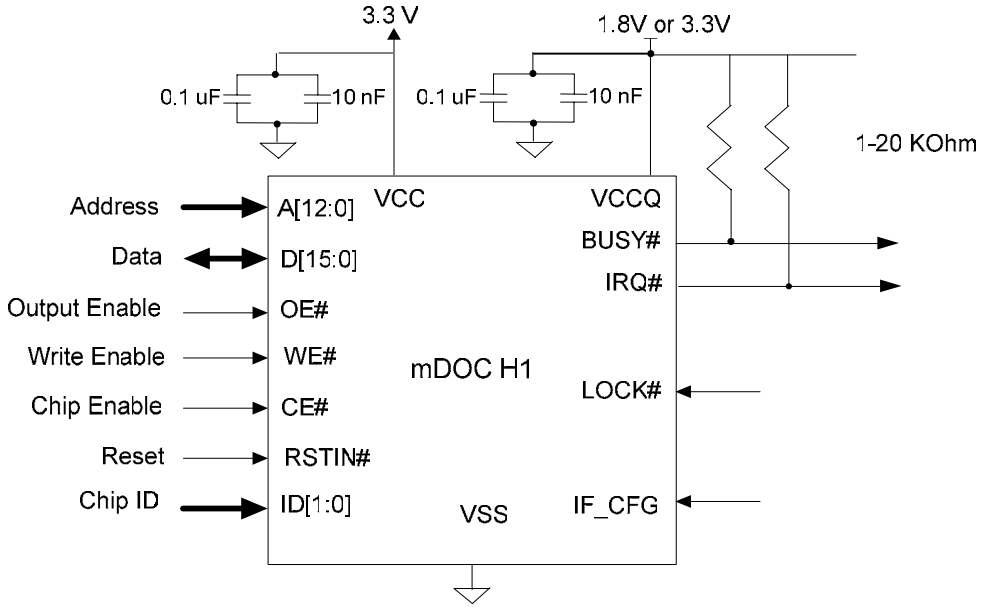



*Figure 10: Standard System Interface*

Note: The 0.1 µF and the 10 nF low-inductance, high-frequency capacitors must be attached to each of the device's VCC and VSS balls. These capacitors must be placed as close as possible to the package leads.

## 7.3 Multiplexed Interface

With a multiplexed interface, mDOC H1 requires the signals shown in Figure 11 below.

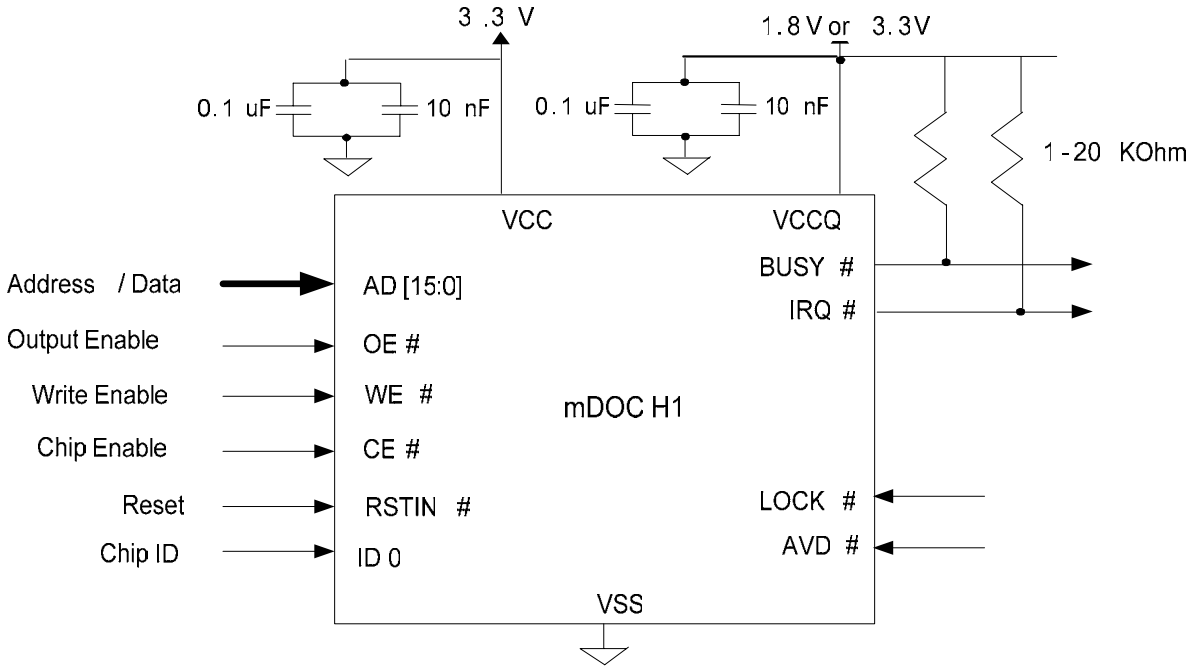



*Figure 11: Multiplexed System Interface*

## 7.4 Connecting Control Signals

### 7.4.1 Standard Interface

When using a standard NOR-like interface, connect the control signals as follows:

- A[12:0] – Connect these signals to the host's address signals (see Section 7.8 for platform-related considerations).
- D[15:0] – Connect these signals to the host's data signals (see Section 7.8 for platform-related considerations).
- Output Enable (OE#) and Write Enable (WE#) – Connect these signals to the host RD# and WR# signals, respectively.
- Chip Enable (CE#) – Connect this signal to the memory address decoder. Most RISC processors include a programmable decoder to generate various Chip Select (CS) outputs for different memory zones. These CS signals can be programmed to support different wait states to accommodate mDOC H1 timing specifications.
- Power-On Reset In (RSTIN#) – Connect this signal to the host's active-low Power-On Reset signal. Note: The reset circuit should be designed to accommodate system specific requirement
- Chip Identification (ID[1:0]) – Connect these signals as shown in. Both signals must be connected to VSS if the host uses only one mDOC H1. If more than one device is being used, refer to Section 7.6 for more information on device cascading.
- Busy (BUSY#) – This signal indicates when the device is ready for first access after reset. It may be connected to an input port of the host, or alternatively it may be used to hold the host in a wait-state condition. The later option is required for hosts that boot from mDOC H1.
- Interrupt Request ( IRQ#) – Connect this signal to the host interrupt input.
- Lock (LOCK#) – Connect to a logical 0 to prevent the usage of the protection key to open a protected partition. Connect to logical 1 in order to enable usage of protection keys. Note: If this feature will not be used then connect LOCK# to VCCQ.
- 8/16 Bit Interface Configuration (IF_CFG) – This signal is required for configuring the device for 8- or 16-bit access mode. When low, the device is configured for 8-bit access mode. When high, 16-bit access mode is selected.

### 7.4.2 Multiplexed Interface

mDOC H1 can use a multiplexed interface to connect to a CPU with multiplexed bus (asynchronous read/write protocol). In this configuration, the AVD# input is driven by the host's AVD# signal, and the D[15:0] balls, used for both address inputs and data, are connected to the host AD[15:0] bus. As with a standard interface, only address bits [12:0] are significant.

This mode is automatically entered when a falling edge is detected on AVD#. This edge must occur after RSTIN# is negated and before OE# and CE# are both asserted; i.e., the first read cycle made to mDOC H1 must observe the multiplexed mode protocol. See Section 8.3 for more information about the related timing requirements. Please refer to Section 1.2 for ballout and signal descriptions.

## 7.5 Implementing the Interrupt Mechanism

### 7.5.1 Hardware Configuration

To configure the hardware for working with the interrupt mechanism, connect the IRQ# ball to the host interrupt input.

Note: A nominal 10 KΩ pull-up resistor should be connected to this ball.

### 7.5.2 Software Configuration

Configuring the software to support the IRQ# interrupt is performed in two stages.

**Stage 1**

Configure the software so that when the system is initialized, the following steps occur:

1. The correct value is written to the Interrupt Control register to configure mDOC H1 for:
   - Interrupt source: Flash ready and /or data protection and/or FIFO error
   - Output type: either edge or level-triggered

   Note: Refer to Section 5.9 for further information on the value to write to this register.
2. The host interrupt is configured to the selected input type, either edge or level-triggered.
3. The handshake mechanism between the interrupt handler and the OS is initialized.
4. The interrupt service routine to the host interrupt is connected and enabled.

**Stage 2**

Configure the software so that for every long flash I/O operation, the following steps occur:

1. The correct value is written to the Interrupt Control register to enable the IRQ# interrupt.

   Note: Refer to Section 5 for further information on the value to write to this register.
2. The flash I/O operation starts.
3. Control is returned to the OS to continue other tasks. When the IRQ# is received, other interrupts are disabled and the OS is flagged.
4. The OS either returns control immediately to the TrueFFS driver, or waits for the appropriate condition to return control to the TrueFFS driver.

## 7.6 Device Cascading

When connecting mDOC H1 using either standard or multiplexed interface, up to two devices can be cascaded with no external decoding circuitry. Figure 12 illustrates the configuration required to cascade two devices on the host bus (only the relevant cascading signals are included in this figure, although all other signals must also be connected). All the balls of the cascaded devices must be wired in common, except for ID0 and ID1. The ID0 input ball is strapped to VCCQ or VSS, according to the location of each mDOC. The ID1 input ball is always strapped to VSS. The ID0 ball value determines the identity of each device. The first device is identified by connecting the ID0 ball to VSS, and the second device by connecting the ID0 ball to VCCQ. Systems that use only one mDOC H1 must connect the ID0 ball VSS.

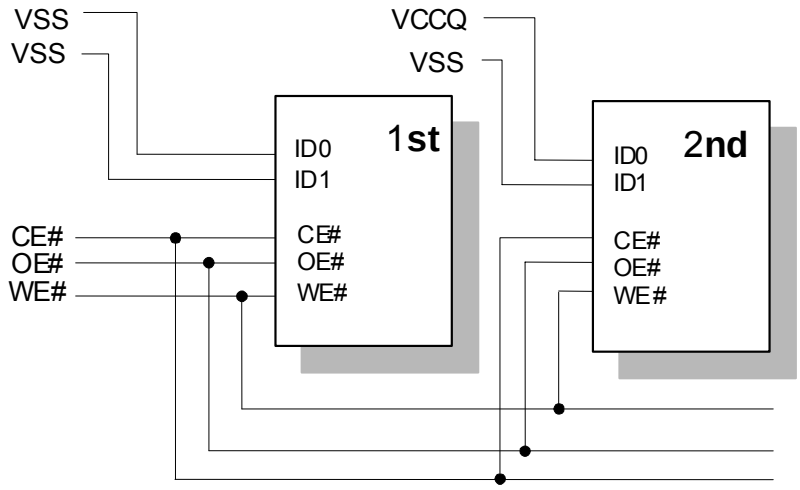



*Figure 12: Standard Interface, Cascaded Configuration*

Note: When more than one mDOC H1 is cascaded, a boot block of up to 2KB is available. The Programmable Boot Block of each device is mapped to a unique address space.

## 7.7 Boot Replacement

A typical RISC architecture uses a boot ROM for system initialization. The boot ROM is also required to access mDOC H1 during the boot sequence in order to load OS images and the device drivers.

msystems' Boot Software Development Kit (BDK) and DOS utilities enable full control of mDOC H1 during the boot sequence. For a complete description of these products, refer to the *mDOC Boot Software Development Kit (BDK)* developer guide and the *mDOC Software Utilities* user manual. These tools enable the following operations:

- Formatting mDOC H1
- Creating multiple partitions for different storage needs (OS images files, registry entry files, backup partitions, and FAT partitions)
- Loading the OS image file

Figure 13 illustrates the system boot flow using mDOC H1 in a RISC architecture

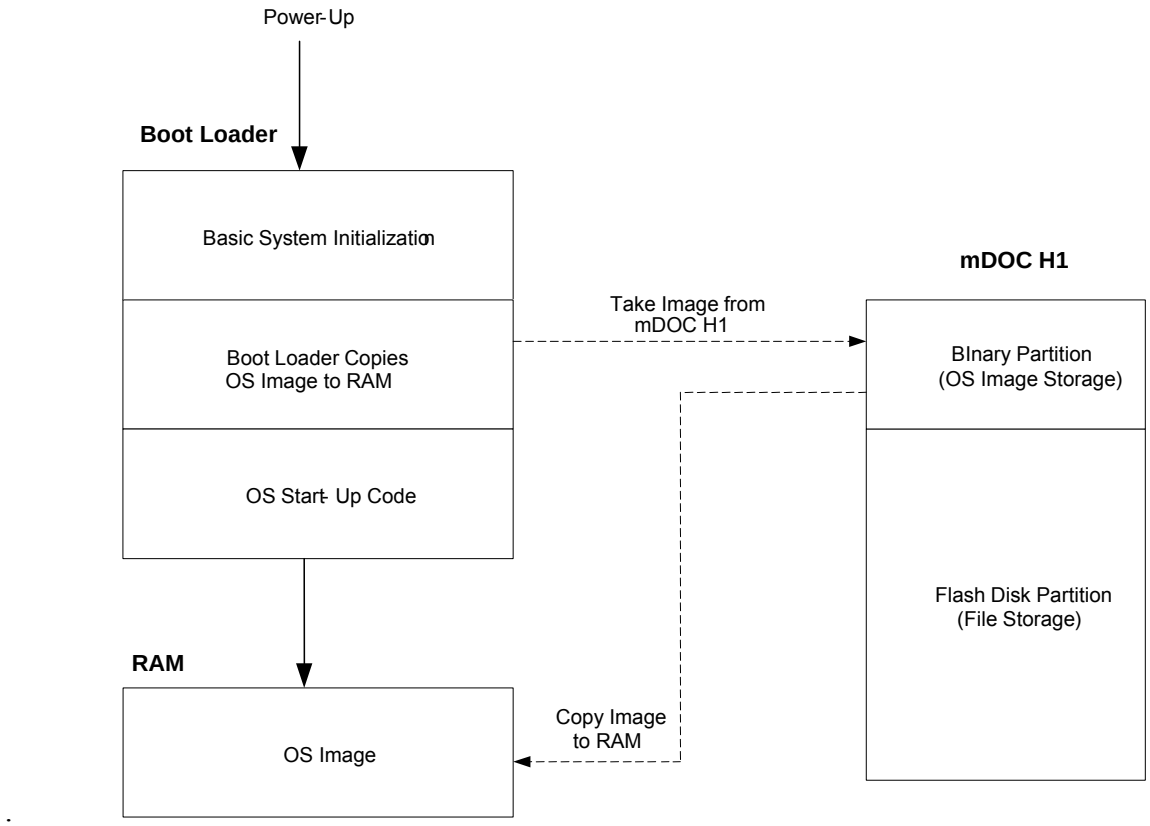



*Figure 13: System Boot Flow with mDOC H1*

## 7.8 Platform-Specific Issues

This section discusses hardware design issues for major embedded RISC processor families.

### 7.8.1 Wait State

Wait states can be implemented only when mDOC H1 is designed in a bus that supports a Wait state insertion, and supplies a WAIT signal.

### 7.8.2 Big and Little Endian Systems

mDOC H1 is a Little Endian device. Therefore, byte lane 0 (D[7:0]) is its Least Significant Byte (LSB) and byte lane 1 (D[15:8]) is its Most Significant Byte (MSB). Within the byte lanes, bit D0 and bit D8 are the least significant bits of their respective byte lanes. mDOC H1 can be connected to a Big Endian device in one of two ways:

1. Make sure to identify byte lane 0 and byte lane 1 of your processor. Then, connect the data bus so that the byte lanes of the CPU match the byte lanes of mDOC H1. Pay special attention to processors that also change the bit ordering within the bytes (for example, PowerPC). Failing to follow these rules results in improper connection of mDOC H1, and prevents the TrueFFS driver from identifying it.
2. Set the bits SWAPH and SWAPL in the Endian Control register. This enables byte swapping when used with 16-bit hosts.

### 7.8.3 Busy Signal

The Busy signal (BUSY#) indicates that mDOC H1 has not yet completed internal initialization. After reset, BUSY# is asserted while the IPL is downloaded into the internal boot block and the Data Protection Structures (DPS) are downloaded to the Protection State Machines. Once the download process is completed, BUSY# is negated. It can be used to delay the first access to mDOC H1 until it is ready to accept valid cycles.

Note: mDOC H1 does NOT use this signal to indicate that the flash is in busy state (e.g. program, read, or erase).

### 7.8.4 Working with 8/16/32-Bit Systems

mDOC H1 uses a 16-bit data bus and supports 16-bit data access by default. However, it can be configured to support 8 or 32-bit data access mode. This section describes the connections required for each mode.

The default of the TrueFFS driver for mDOC H1 is set to work in 16-bit mode. It must be specially configured to support 8 and 32-bit mode. Please see TrueFFS documentation for further details.

Note: The mDOC H1 data bus must be connected to the Least Significant Bits (LSB) of the system. The system engineer must verify whether the matching host signals are SD[7:0], SD[15:8] or D[31:24].

### 8-Bit (Byte) Data Access Mode

When configured for 8-bit operation, ball IF_CFG should be connected to VSS, data lines D[15:8] are internally pulled up and may be left unconnected. The device routes odd and even address accesses to the appropriate byte lane of the flash and RAM.

Host address SA0 must be connected to mDOC H1 A0, SA1 must be connected to A1, etc.

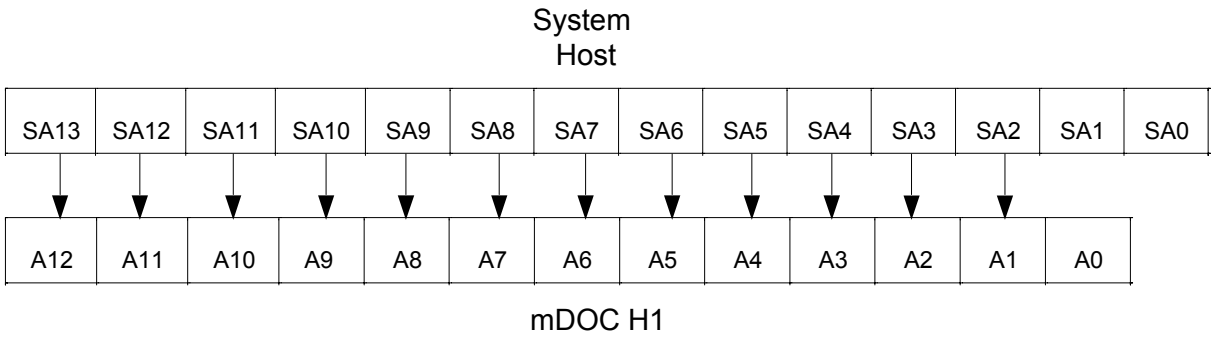
### 16-Bit (Word) Data Access Mode

To set mDOC H1 to work in 16-bit mode, the IF_CFG ball must be connected to VCCQ.

In 16-bit mode, the Programmable Boot Block is accessed as a true 16-bit device. It responds with the appropriate data when the CPU issues either an 8-bit or 16-bit read cycle. The flash area is accessed as a 16-bit device, regardless of the interface bus width. This has no affect on the design of the interface between mDOC H1 and the host. The TrueFFS driver handles all issues regarding moving data in and out of mDOC H1.

### 32-Bit (Double Word) Data Access Mode

In a 32-bit bus system that cannot execute byte- or word-aligned accesses, the system address lines SA0 and SA1 are always 0. Consecutive double words (32-bit words) are differentiated by SA2 toggling. Therefore, in 32-bit systems that support only 32-bit data access cycles, mDOC H1 signal A0 is connected to VSS and A1 is connected to the first system address bit that toggles; i.e., SA2.




Note: The prefix “S” indicates system host address lines

*Figure 14: Address Shift Configuration for 32-Bit Data Access Mode*

## 7.9 Design Environment

mDOC H1 provides a complete design environment consisting of:

- Evaluation boards (EVBs) for enabling software integration and development with mDOC H1, even before the target platform is available.
- Programming solutions:
  - Programming house
  - On-board programming
- TrueFFS Software Development Kit (SDK) and Boot Software Development Kit (BDK)
- DOS utilities:
  - DFORMAT
  - DIMAGE
  - DINFO
- Documentation:
  - Data sheet
  - Application notes
  - Technical notes
  - Articles
  - White papers

Please visit the msystems website (www.m-systems.com) for the most updated documentation, utilities and drivers.

# 8. PRODUCT SPECIFICATIONS

## 8.1 Environmental Specifications

### 8.1.1 Operating Temperature

Extended temperature range: -30°C to +85°C

### 8.1.2 Thermal Characteristics

*Table 4: Thermal Characteristics*

| Thermal Resistance (°C/W) | |
|---|---|
| Junction to Case ($\theta_{JC}$): 35 | Junction to Ambient ($\theta_{JA}$): 85 |

### 8.1.3 Humidity

10% to 90% relative, non-condensing

## 8.2 Electrical Specifications

### 8.2.1 Absolute Maximum Ratings

*Table 5: Absolute Maximum Ratings*

| Symbol | Parameter | Rating[1] | Unit |
|---|---|---|---|
| VCC | DC core supply voltage | -0.3-4 | V |
| VCCQ | DC I/O supply voltage | -0.3-4 | V |
| $T_{1SUPPLY}$ | Maximum duration of applying VCCQ without VCC, or VCC without VCCQ | 500[3] | msec |
| $V_{IN}$ (VCCQ=1.65 to 1.95V) | Input ball voltage[2] | -0.3 to VCCQ+0.3V | V |
| $V_{IN}$ (VCCQ=2.5 to 3.6V) | | -0.3V to 6.0V | |
| $T_{STG}$ | Storage temperature | -55 to 125 | °C |

Notes: 1. Permanent device damage may occur if absolute maximum ratings are exceeded. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

2. The voltage on any ball may undershoot to -1.0V or overshoot to +1.0V for less than 20 ns.

3. When operating mDOC H1 with separate power supplies for VCC and VCCQ, it is recommended to turn both supplies on and off simultaneously. Providing power separately (either at power-on or power-off) can cause excessive power dissipation. Damage to the device may result if this condition persists for more than 500 msec.

### 8.2.2 Capacitance

*Table 6: Capacitance*

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $C_{IN}$ | Input capacitance | $V_{IN}$ = 0V, $T_j$=25C, F=1Mhz | | | 30[1] | pF |
| $C_{IO}$ | Input/Output capacitance | $V_{IN}$ = 0V, $T_j$=25C, F=1Mhz | | | 30[1] | pF |

Notes: 1. The following signals may exceed indicated max capacitance up to a level of 60pF: D5, D6, D7, D14 and IRQ.

### 8.2.3 DC Electrical Characteristics over Operating Range

See Table 7 and Table 8 for DC characteristics for VCCQ ranges 1.65-1.95V and 2.5-3.6V I/O, respectively.

*Table 7: DC Characteristics, VCCQ = 1.65-1.95V I/O*

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| VCC | Core supply voltage | | 2.7 | 3.0 | 3.6 | V |
| VCCQ | Input/Output supply voltage | | 1.65 | 1.8 | 1.95 | V |
| $V_{IH}$ | High-level input voltage | | VCCQ – 0.4 | | | V |
| $V_{IL}$ | Low-level input voltage | | | | 0.4 | V |
| $V_{OH}$ | High-level output voltage | $I_{OH}$ = -100 µA | VCCQ – 0.1 | | | V |
| $V_{OL}$ | Low-level output voltage | $I_{OL}$ = 100 µA | | | 0.1 | V |
| $V_{HYS}$ | Hysteresis | ST signals | 0.2 | | | V |
| $I_{LK}$ | leakage current | $V_{IN}$ = VSS or VCCQ | | | ±10 | µA |
| | | $V_{IN}$ = VSS ~ VCCQ | | | ±20 | µA |
| $I_{CC}$ | Active VCC supply current | Cycle Time = 109 ns | | 43 | 68 | mA |
| $I_{ccq}$ | Active VCCQ supply current | Cycle Time = 109 ns | | 1 | 6 | mA |
| $I_{CCS}$ | Standby supply current VCC balls | Deep Power-Down mode | | 22 | 110 | µA |
| $I_{CCQS}$ | Standby supply current VCCQ balls | Deep Power-Down mode | | 3 | 10 | µA |

*Table 8: DC Characteristics, VCCQ = 2.5V-3.6V*

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| VCC | Core supply voltage | | 2.7 | 3.3 | 3.6 | V |
| VCCQ | Input/Output supply voltage | | 2.5 | 3.3 | 3.6 | V |
| $V_{IH}$ | High-level input voltage | PCI Input | VCCQ/2 | | | V |
| | | ST Input | VCCQ-0.6 | | | |
| $V_{IL}$ | Low-level input voltage | PCI Input | | | VCCQ*0.3 | V |
| | | ST Input | | | 0.35 | V |

| Symbol | Parameter | Conditions | Min | Typ | Max | Unit |
|---|---|---|---|---|---|---|
| $I_{LK}$ | leakage current | $V_{IN}$ = VSS or VCCQ | | | ±10 | µA |
| | | $V_{IN}$ = VSS ~ VCCQ | | | ±20 | µA |
| $V_{OH}$ | High-level output voltage | $I_{OH}$ = -8 mA | VCCQ-0.6 | VCCQ-0.6 | VCCQ-0.6 | V |
| $V_{OL}$ | Low-level output voltage | $I_{OL}$ = 8 mA | 0.4 | 0.4 | 0.4 | V |
| $V_{HYS}$ | Hysteresis | ST signals, VCCQ = 2.5 | 0.26 | | | V |
| | | ST signals, VCCQ = 2.7 | 0.30 | | | V |
| | | ST signals, VCCQ = 3.0 | 0.4 | | | V |
| $I_{CC}$ | Active supply current | Cycle Time = 109 ns | | 43 | 68 | mA |
| $I_{ccq}$ | Active VCCQ supply current | Cycle Time = 109 ns | | 1 | 6 | mA |
| $I_{CCS}$ | Standby supply current VCC balls | Deep Power-Down mode | | 22 | 110 | µA |
| $I_{CCQS}$ | Standby supply current VCCQ balls | Deep Power-Down mode | | 3 | 10 | µA |

### 8.2.4 AC Operating Conditions

Timing specifications are based on the conditions defined below.

*Table 9: AC Characteristics*

| Parameter | VCCQ = 1.65-1.95V | VCCQ = 2.5-3.6V |
|---|---|---|
| Ambient temperature (TA) | -30°C to +85°C | -30°C to +85°C |
| Core supply voltage (VCC) | 2.7V to 3.6V | 2.7V to 3.6V |
| Input pulse levels | 0V to VCCQ | 0V to VCCQ |
| Input rise and fall times | 1V/ns | 1V/ns |
| Input timing levels[1] | VCCQ/2 | VCCQ/2 |
| Output timing levels | VCCQ/2 | VCCQ/2 |
| Output load | 30 pF | 130 pF |

Notes: 1. Test conditions for PCI buffers are in accordance with the PCI Local Bus Specification, revision 2.2.

## 8.3 Timing Specifications

### 8.3.1 Read Cycle Timing Standard Interface

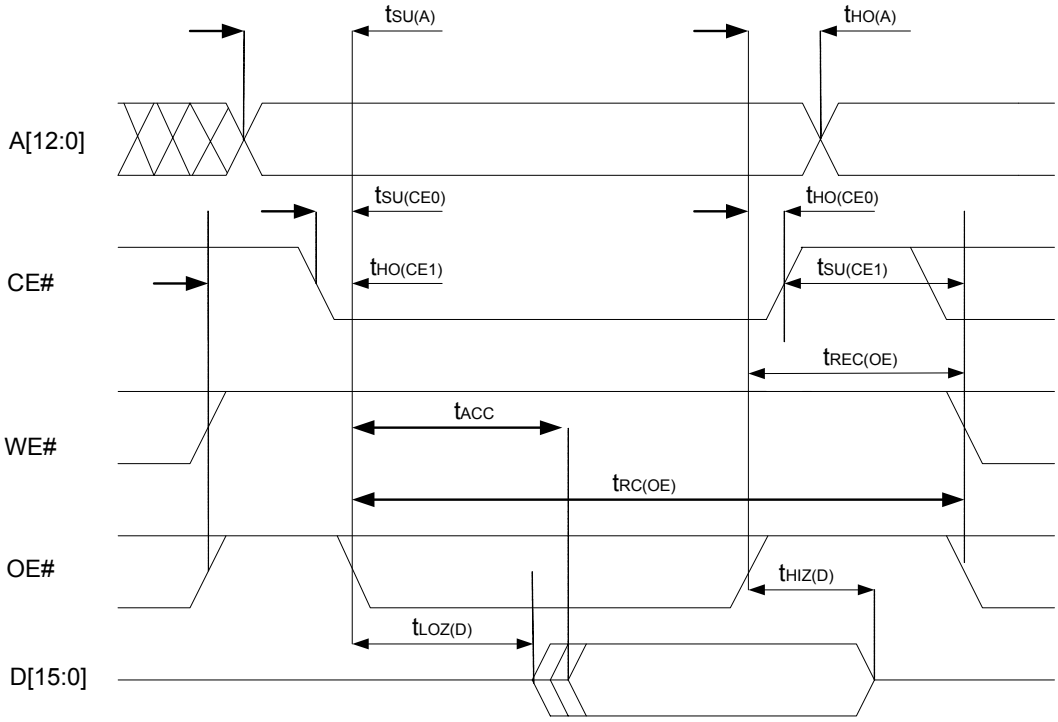



*Figure 15: Standard Interface, Read Cycle Timing*

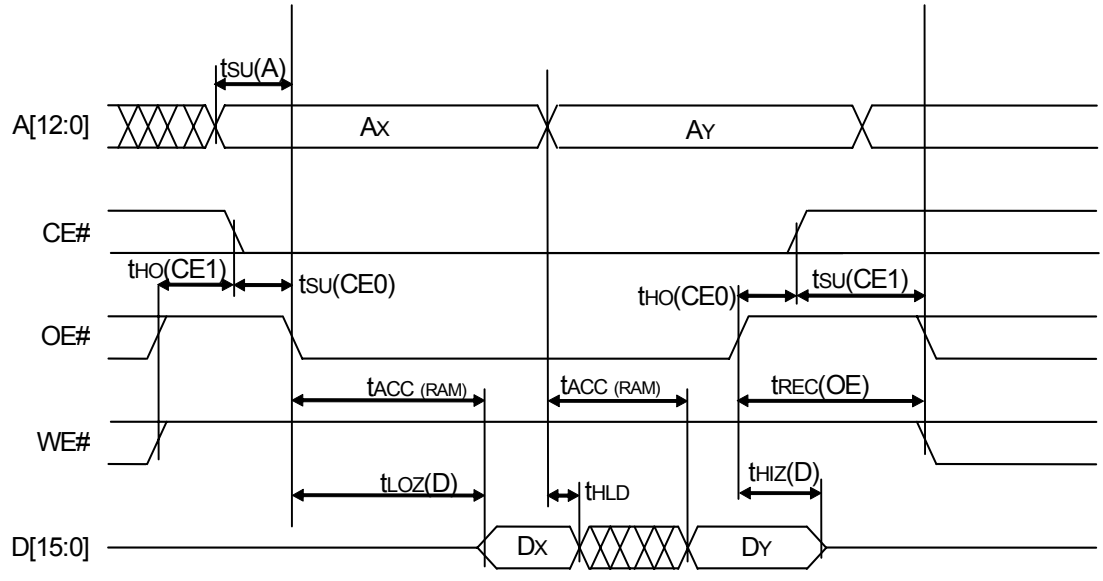



*Figure 16: Standard Interface, Read Cycle Timing – Asynchronous Boot Mode*

*Table 10: Standard Interface Read Cycle Timing Parameters*

| Symbol | Description | VCCQ = 1.65-1.95V | | VCCQ = 2.5-3.6V | | Units |
|---|---|---|---|---|---|---|
| | | Min | Max | Min | Max | |
| $t_{SU(A)}$ | Address to OE# ↓ setup time | 0 | | 0 | | nS |
| $t_{HO(A)}$ | OE# ↑ to Address hold time | 0 | | 0 | | nS |
| $t_{SU(CE0)}$[1] | CE# ↓ to OE# ↓ setup time | - | | - | | nS |
| $t_{HO(CE0)}$[2] | OE# ↑ to CE# ↑ hold time | - | | - | | nS |
| $t_{HO(CE1)}$ | OE# ↑ to CE# ↓ hold time | 11 | | 11 | | nS |
| $t_{SU(CE1)}$ | CE# ↑ to OE# ↓ setup time | 11 | | 11 | | nS |
| $t_{REC(OE)}$ | OE# negated to start of next cycle | 14 | | 14 | | nS |
| $t_{ACC(RAM)}$[4] | Read access time (RAM) | 51 | | 43 | | nS |
| $t_{ACC}$[4] | Read access time (all other addresses) | 64 | | 64 | | nS |
| $t_{RC(OE)}$ | Read Cycle Time | 109 | | 109 | | nS |
| $t_{LOZ(D)}$[3] | OE# ↓ to D driven | 8.4 | | 7.6 | | nS |
| $t_{HIZ(D)}$[5] | OE# ↑ to D Hi-Z delay | 14 | | 14 | | nS |
| $t_{HLD}$[6] | Address to Data hold time | 14 | | 14 | | nS |

Notes:
1. CE# may be asserted any time before or after OE# is asserted. If CE# is asserted after OE#, all timing relative to OE# asserted will be referenced instead to the time of CE# asserted.
2. CE# may be negated any time before or after OE# is negated. If CE# is negated before OE#, all timing relative to OE# negated will be referenced instead to the time of CE# negated.
3. No load (CL = 0 pF).
4. Does not include margin on buffer output enable to eliminate multiple transitions.
5. Does not include buffer turn off time or output enable wire delays (unknown).
6. Applicable to asynchronous boot mode only.

### 8.3.2 Write Cycle Timing Standard Interface

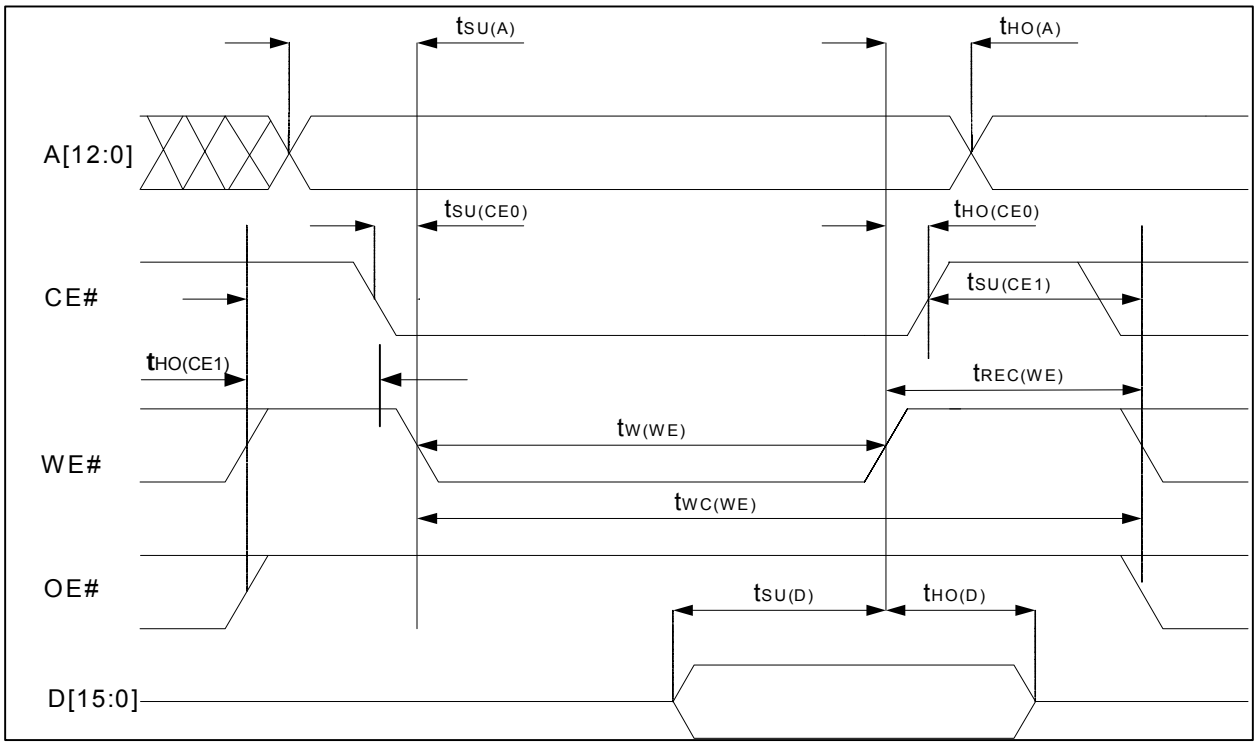



*Figure 17: Standard Interface Write Cycle Timing*

*Table 11: Standard Interface Write Cycle Parameters*

| Symbol | Description | VCCQ = 1.65-1.95V | | VCCQ = 2.5-3.6V | | Units |
|---|---|---|---|---|---|---|
| | | Min | Max | Min | Max | |
| $t_{SU(A)}$ | Address to WE# ↓ setup time | 0 | | 0 | | nS |
| | Address to WE# ↓ setup time (RAM access) | 12 | | 12 | | nS |
| $t_{HO(A)}$ | WE# ↑ to Address hold time | 0 | | 0 | | nS |
| $t_{W(WE)}$ | WE# asserted width (RAM access) | 40 | | 40 | | nS |
| | WE# asserted width (all other addresses) | 64 | | 64 | | |
| $t_{WC(WE)}$ | Write Cycle Time | 109 | | 109 | | |
| $t_{SU(CE0)}$[1] | CE# ↓ to WE# ↓ setup time (RAM access) | 12 | | 12 | | nS |
| | CE# ↓ to WE# ↓ setup time (all other addresses) | - | | - | | nS |
| $t_{HO(CE0)}$[2] | WE# ↑ to CE# ↑ hold time | - | | - | | nS |

| Symbol | Description | VCCQ = 1.65-1.95V | | VCCQ = 2.5-3.6V | | Units |
|---|---|---|---|---|---|---|
| | | Min | Max | Min | Max | |
| $t_{HO(CE1)}$ | WE# ↑ to CE# ↓ hold time | 11 | | 11 | | nS |
| $t_{SU(CE1)}$ | CE# ↑ to WE# ↓ setup time | 11 | | 11 | | nS |
| $t_{REC(WE)}$ | WE# ↑ to start of next cycle | 11 | | 11 | | nS |
| $t_{SU(D)}$ | D to WE# ↑ setup time (RAM access) | 22 | | 22 | | nS |
| | D to WE# ↑ setup time (all other addresses) | 28 | | 28 | | nS |
| $t_{HO(D)}$ | WE# ↑ to D hold time | 2 | | 2 | | nS |

Notes: 1. CE# may be asserted any time before or after OE# is asserted. If CE# is asserted after WE#, all timing relative to WE# asserted will be referenced instead to the time of CE# asserted.

2. CE# may be negated any time before or after WE# is negated. If CE# is negated before WE#, all timing relative to WE# negated will be referenced instead to the time of CE# negated.

### 8.3.3 Read Cycle Timing Multiplexed Interface

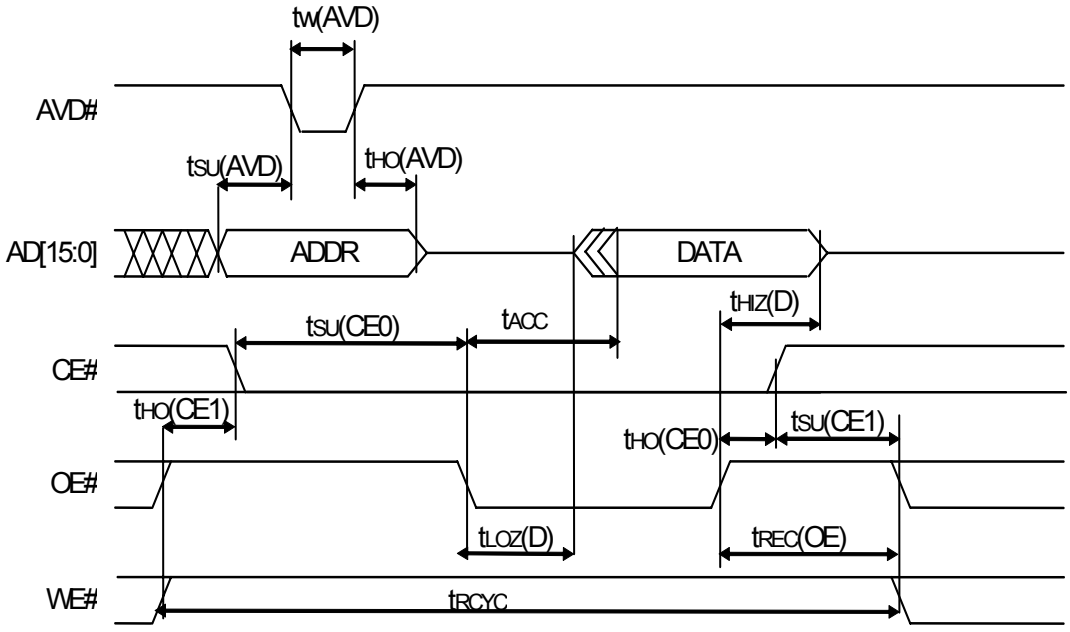



*Figure 18: Multiplexed Interface Read Cycle Timing*

*Table 12: Multiplexed Interface Read Cycle Parameters*

| Symbol | Description | VCCQ = 1.65~1.95V | | VCCQ = 2.5~3.6V | | Units |
|---|---|---|---|---|---|---|
| | | Min | Max | Min | Max | |
| $t_{SU(AVD)}$ | Address to AVD# ↓ setup time | 7 | | 7 | | nS |
| $t_{HO(AVD)}$ | Address to AVD# ↑ hold time | 9 | | 9 | | nS |
| $t_{W(AVD)}$ | AVD# low pulse width | 14 | | 14 | | ns |
| $t_{ACC}$ | Read Access time(RAM) | 51 | | 43 | | nS |
| | Read Access time (all other addresses) | 64 | | 64 | | nS |
| $t_{RCYC}$ | Read Cycle Time | 126 | | 126 | | nS |
| $t_{SU(CE0)}$[1] | CE# ↓ to OE# ↓ setup time | 17 | | 17 | | nS |
| | (RAM write access) | | | | | |
| $t_{HO(CE0)}$[2] | OE# ↑ to CE# ↑ hold time | 0 | | 0 | | nS |
| $t_{HO(CE1)}$ | OE# ↑ to CE# ↓ hold time | 11 | | 11 | | nS |
| $t_{SU(CE1)}$ | CE# ↑ to OE# ↓ setup time | 11 | | 11 | | nS |
| $t_{REC(OE)}$ | OE# ↑ to start of next cycle | 14 | | 14 | | nS |
| $t_{LOZ(D)}$[3] | OE# ↓ to D driven | 8.4 | | 7.6 | | nS |
| $t_{HIZ(D)}$[5] | OE# ↑ to D Hi-Z delay | 14 | | 14 | | nS |

Notes:
1. CE# may be asserted any time before or after OE# is asserted. If CE# is asserted after OE#, all timing relative to OE# asserted will be referenced instead to the time of CE# asserted.
2. CE# may be negated any time before or after OE# is negated. If CE# is negated before OE#, all timing relative to OE# negated will be referenced instead to the time of CE# negated.
3. No load (CL = 0 pF).
4. Does not include margin on buffer output enable to eliminate multiple transitions.
5. Does not include buffer turn off time or output enable wire delays (unknown).

### 8.3.4 Write Cycle Timing Multiplexed Interface

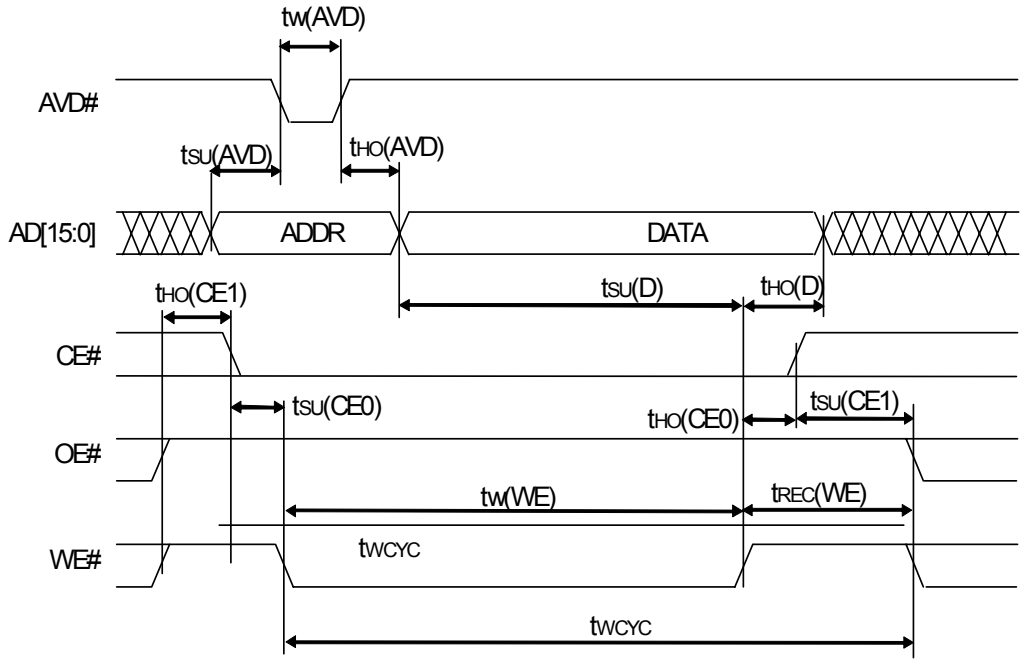



*Figure 19: Multiplexed Interface Write Cycle Timing*

*Table 13: Multiplexed Interface Write Cycle Parameters*

| Symbol | Description | VCCQ = 1.65~1.95V | | VCCQ = 2.5~3.6V | | Units |
|---|---|---|---|---|---|---|
| | | Min | Max | Min | Max | |
| $t_{SU(AVD)}$ | Address to AVD# ↓ setup time | 7 | | 7 | | nS |
| $t_{HO(AVD)}$ | Address to AVD# ↑ hold time | 9 | | 9 | | nS |
| $t_{W(AVD)}$ | AVD# low pulse width | 14 | | 14 | | ns |
| $t_{W(WE)}$ | WE# asserted width (RAM) | 40 | | 40 | | nS |
| | WE# asserted width (all other addresses) | 64 | | 64 | | |
| $t_{WCYC}$ | Write Cycle Time | 139 | | 139 | | |
| $t_{SU(CE0)}$[1] | CE# ↓ to WE# ↓ setup time | 17 | | 17 | | nS |
| | (RAM write access) | | | | | |
| $t_{HO(CE0)}$[2] | WE# ↑ to CE# ↑ hold time | 0 | | 0 | | nS |
| $t_{HO(CE1)}$ | WE# ↑ to CE# ↓ hold time | 11 | | 11 | | nS |
| $t_{SU(CE1)}$ | CE# ↑ to WE# ↓ setup time | 11 | | 11 | | nS |
| $t_{REC(WE)}$ | WE# ↑ to start of next cycle | 11 | | 11 | | nS |
| $t_{SU(D)}$ | D to WE# ↑ setup time (RAM) | 22 | | 22 | | nS |

| Symbol | Description | VCCQ = 1.65~1.95V | | VCCQ = 2.5~3.6V | | Units |
|---|---|---|---|---|---|---|
| | | Min | Max | Min | Max | |
| | D to WE# ↑ setup time (all other addresses) | 28 | | 28 | | nS |
| $t_{HO(D)}$ | WE# ↑ to D hold time | 2 | | 2 | | nS |

Notes: 1. CE# may be asserted any time before or after OE# is asserted. If CE# is asserted after WE#, all timing relative to WE# asserted will be referenced instead to the time of CE# asserted.

2. CE# may be negated any time before or after WE# is negated. If CE# is negated before WE#, all timing relative to WE# negated will be referenced instead to the time of CE# negated.

### 8.3.5 Power Supply Sequence

When operating mDOC H1 with separate power supplies powering the VCCQ and VCC rails, it is desirable to turn both supplies on and off simultaneously. Providing power to one supply rail and not the other (either at power-on or power-off) can cause excessive power dissipation. Damage to the device may result if this condition persists for more than 500 msec.

### 8.3.6 Power-Up Timing

mDOC H1 is reset by assertion of the RSTIN# input. When this signal is negated, mDOC H1 initiates a download procedure from the flash memory into the internal Programmable Boot Block. During this procedure, mDOC H1 does not respond to read or write accesses.

Host systems must therefore observe the requirements described below for first access to mDOC H1. Any of the following methods may be employed to guarantee first-access timing requirements:

1. Use a software loop to wait at least Tp (BUSY1-BUSY1) before accessing the device after the reset signal is negated.
2. Poll the state of the BUSY# output.
3. Use the BUSY# output to hold the host CPU in wait state before completing the first access which will be a RAM read cycle. The data will be valid when BUSY# is negated.

Hosts that use mDOC H1 to boot the system must employ option 3 above or use another method to guarantee the required timing of the first-time access.

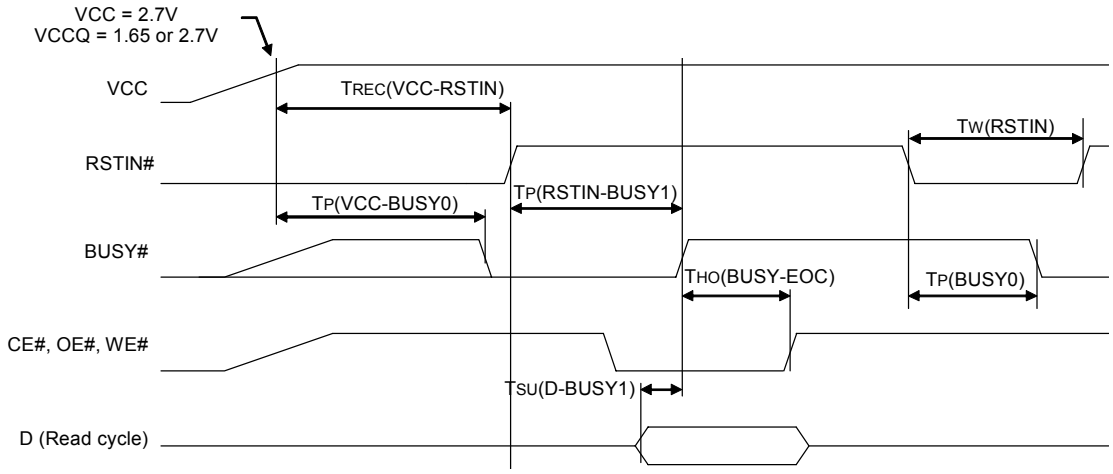



*Figure 20: Reset Timing*

*Table 14: Power-Up and Reset Timing Parameters*

| Symbol | Description | Min | Typical | Max | Units |
|---|---|---|---|---|---|
| $t_{RISE(VCC)}$[1] | VCC rise time | | | 18 | ms |
| $t_{RISE(VCCQ)}$[1] | VCCQ rise time | | | 18 | ms |
| $t_{REC(VCC-RSTIN)}$[4] | VCC > minimum operating voltage to RSTIN# ↑ | 100 | | | nS |
| $t_{W(RSTIN\#)}$ | RSTIN# asserted pulse width | 60 | | | nS |
| $t_{P(BUSY0)}$ | RSTIN# ↓ to BUSY# ↓ | | | 50 | nS |
| $t_{P(RSTIN-BUSY1)}$[3] | RSTIN# ↑ to BUSY# ↑ | | 380 | 1000 | µS |
| $t_{HO(BUSY-EOC)}$[2] | BUSY# ↑ to end of cycle | 0 | | | nS |
| $t_{SU(D-BUSY1)}$[2] | Data valid to BUSY# ↑ | 0 | | | nS |
| $t_{P(VCC-BUSY0)}$ | VCC stable to BUSY# ↓ | | | 50 | nS |

Notes:
1. Specified from the first positive crossing of VCC/VCCQ above 0.5V to the final positive crossing above the minimum VCC/VCCQ operating voltage (typically 2.7V or 3.0V).
2. Relevant parameters refer to the case were host whishes to apply a read/write cycle before BUSY# goes to '1'. (can be done up to Tsu(D-Busy) before de-assertion of BUSY#.
3. If the assertion of RSTIN# occurs during a flash erase cycle, this time could be extended by up to 2ms.
4. VCC indicates both VCC and VCCQ.

### 8.3.7 Interrupt Timing

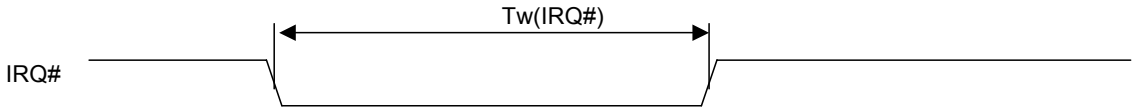



*Figure 21: IRQ# Pulse Width in Edge Mode*

*Table 15: Interrupt Timing*

| Symbol | Description | Min | Max | Unit |
|---|---|---|---|---|
| Tw(IRQ#) | IRQ# asserted pulse width (Edge mode) | 300 | 600 | nS |

## 8.4 Mechanical Dimensions

FBGA dimensions: 12.0 ±0.20 mm x 18.0 ±0.20 mm x 1.3 ±0.1 mm

Ball pitch: 0.8 mm

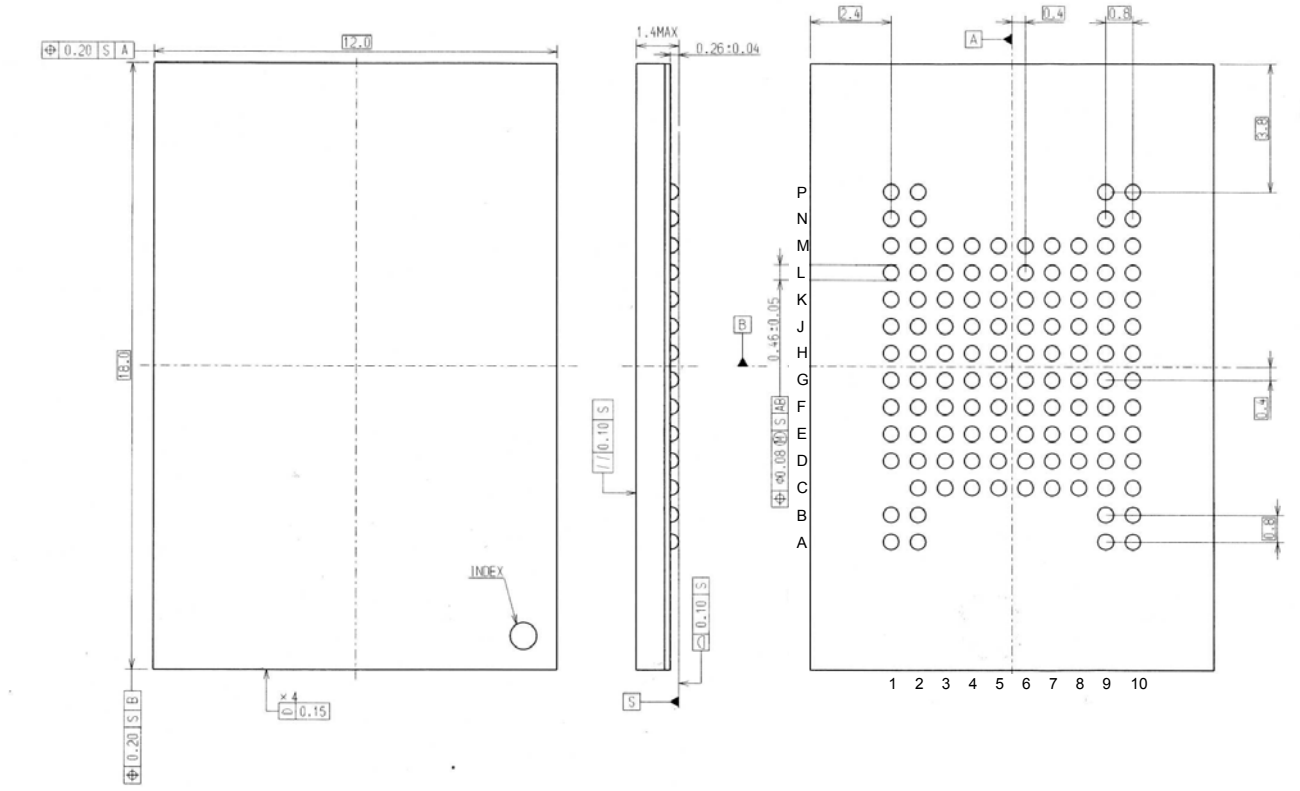



*Figure 22: Mechanical Dimensions 12x18 FBGA Package (Bottom View)*

# 9. ORDERING INFORMATION

Refer to Table 16 for mDOC H1 products currently available and the associated ordering information.

*Table 16: mDOC H1 Available Products*

| Ordering Information | Capacity | | Package | Source Alternative |
|---|---|---|---|---|
| | MB | Gb | | |
| MD2433-d4G-V3Q18-X-P | 512 | 4 | 115-ball 12x18 FBGA Pb-free in Tape & Reel | A/B |
| MD2433-d4G-V3Q18-X-P/Y | 512 | 4 | 115-ball 12x18 FBGA Pb-free in Tray | A/B |
| MD2433-d4G-V3Q18-X-P-H | 512 | 4 | 115-ball 12x18 FBGA Pb-free in Tape & Reel | A |
| MD2433-d4G-V3Q18-X-P-H/Y | 512 | 4 | 115-ball 12x18 FBGA Pb-free in Tray | A |
| MD2433-d4G-V3Q18-X-P-T | 512 | 4 | 115-ball 12x18 FBGA Pb-free in Tape & Reel | B |
| MD2433-d4G-V3Q18-X-P-T/Y | 512 | 4 | 115-ball 12x18 FBGA Pb-free in Tray | B |
| MD2433-d8G-V3Q18-X-P | 1024 | 8 | 115-ball 12x18 FBGA Pb-free in Tape & Reel | N/A |
| MD2433-d8G-V3Q18-X-P/Y | 1024 | 8 | 115-ball 12x18 FBGA Pb-free in Tray | N/A |
| MD2433-d00-DAISY | - | - | Daisy chain | N/A |
| MD2433-d00-DAISY-P | - | - | Pb-free daisy chain | N/A |

# HOW TO CONTACT US

### USA

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

### China

msystems 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

msystems 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

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

### Taiwan

msystems 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