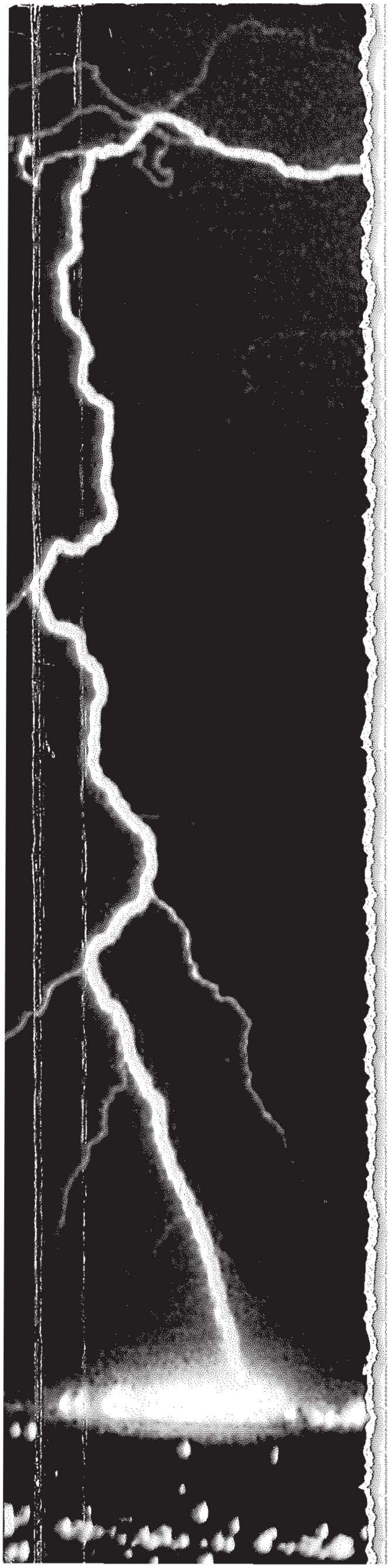


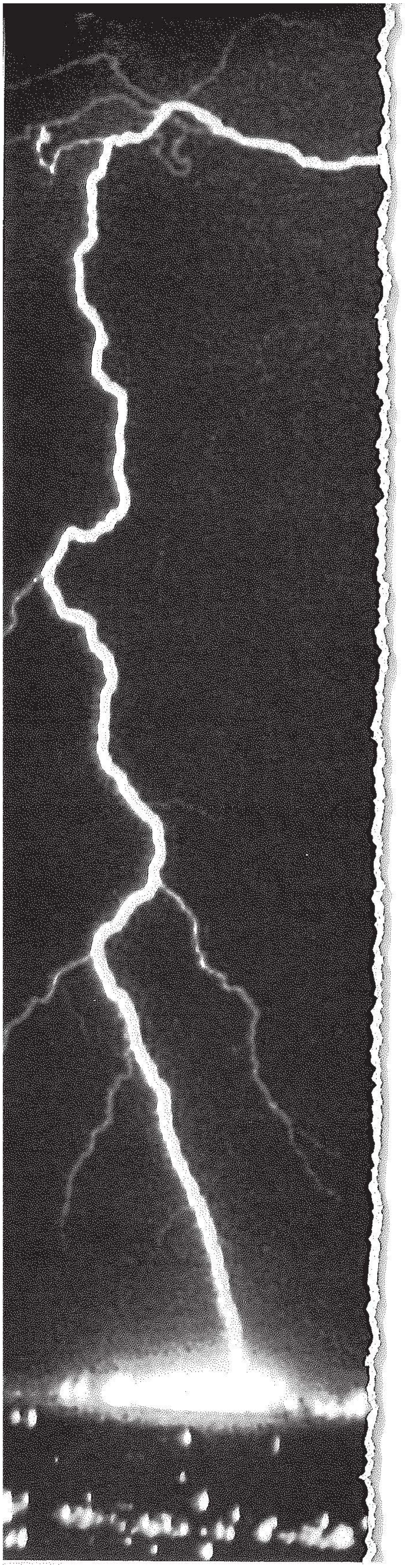
Designing with FLASH MEMORY

Brian Dipert and Markus Levy

The definitive guide to designing flash memory hardware and software for components and PCMCIA cards.

Designing with Flash Memory

The definitive guide to designing flash memory hardware and software for components and PCMCIA cards



Designing with Flash Memory

Brian Dipert & Markus Levy

The definitive guide to designing flash memory hardware and software for components and PCMCIA cards

Annabooks
San Diego

Designing with Flash Memory

BY

BRIAN DIPERT & MARKUS LEVY

PUBLISHED BY

Annabooks
11848 Bernardo Plaza Ct., Suite 110
San Diego, CA 92128
USA

619-673-0870

Copyright © Annabooks 1993, 1994

All rights reserved. No part of the contents of this book may be reproduced or transmitted in any form or by any means without the prior written permission of the publisher, except for the inclusion of brief quotations in a review.

Printed in the United States of America

ISBN 0-929392-17-5
Second Printing April 1994

Contents

Chapter One: Introduction 1
  - Flash Memory Compared to Other Memories 3
    - ROM (Read-Only-Memory) 4
    - RAM (Random-Access-Memory) 4
    - EEPROM (Electrically-Eraseable-Programmable-Read-Only-Memory) 5
    - Magnetic Mass Storage 5
    - An Emerging Alternative: Flash Memory 5
  - A Preview of Chapters to Follow 7

Chapter Two: Flash Memory Applications 11
  - Data Accumulation 12
    - Medical Instrumentation 12
    - Flight Recorders 13
    - More Data Accumulation Examples 13
    - Why Flash Memory for Data Acquisition? 14
  - Data/Lookup Table Storage 14
    - PBX Switcher 14
    - Laser Printers 15
    - Why Flash Memory For Data/Lookup Table Storage? 16
  - Embedded Code Storage 16
    - PC BIOS 16
    - Digital Cellular Phones 18
    - More Embedded Code Storage Applications 18
    - Why Flash Memory for Embedded Code Storage? 18
  - File Storage 19
    - Flash Memory Promotes Longer Battery Life 19
    - HDD Densities with FDD Interchange 20
  - Summary 21

Chapter Three: Flash Memory Technologies 23
  - NOR Flash Memory 25
    - Program 27
    - Erase 28
    - Negative Gate Erase 29
    - Overerase 29
    - NOR Flash Memory Specifications 30

- Flash EEPROM 32
  - Erase 33
  - Programming 34
  - Flash EEPROM Memory Specifications 35
- NAND Flash Memory 36
  - Program and Erase 36
  - NAND Flash Memory Specifications 36
- What's All This *Cycling* Stuff, Anyway? 38
  - Failure Analysis 40
  - Oxide Breakdown 40
  - Electron Trapup 41
  - Mean Time Before Failure 41
  - Extended Cycling-The Vendor's Options 42
  - Extended Cycling-What Can You Do? 43
- Summary 44

Chapter Four: Packaging Options and Update Alternatives 45

- Packaging Options 46
  - DIP (Dual In-Line Package) 46
  - LCC (Leaded/Leadless Chip Carrier) 49
  - SOJ (Small-Outline J-Lead) 52
  - SOP (Small Outline Package) 54
  - TSOP (Thin Small Outline Package) 54
  - SIMM (Single In-Line Leadless Memory Module) 59
  - PCMCIA Flash Memory Cards 62
  - Flash Drives 66
- Update Options 68
  - Off-Board PROM Programming 68
  - On-Board Update 69
  - In-System Write 71
- Summary 72

Chapter Five: Hardware Interfacing to Flash Memory Components 73

- Hardware Interfacing Fundamentals 73
  - Chip Enable 75
  - Addresses 75
  - Data In/Out 76
  - Output Enable 76
  - Write Enable 77

$\overline{\text{WE}}$-Less Flash Memories 78
The V_{PP} Program/Erase Voltage 79
Switching V_{PP} 80
V_{PP} Feedback 81
Advanced Hardware Interfacing 82
The $\overline{\text{PWD}}$ Input 82
RY/$\overline{\text{BY}}$ Output 84
Interpreting Datasheet AC Parameters 85
General Observations 90
Naming Conventions 91
Capacitive Loading and Effects 92
AC Read Characteristics 93
Read Specification Clarifications 94
AC Write Characteristics 95
Write Specification Clarifications 96
Performance Enhancements 98
Caching 98
Shadowing 98
Hardware Interleaving 99
Summary 104
Chapter Six: Power Requirements and Design Techniques 105
The V_{CC} Operating Voltage 106
Read Mode (I_{CCR}) 108
Standby Mode (I_{CCS}) 109
Deep Powerdown Mode (I_{CCD}) 110
Program Mode (I_{CCW}/I_{CCP}) 111
Erase Mode (I_{CCE}) 112
The V_{PP} Program/Erase Voltage 113
Read/Standby Mode (I_{PPR} and I_{PPS}) 114
Deep Powerdown Mode (I_{PPD}) 114
Program Mode (I_{PPW}/I_{PPP}) 115
Erase Mode (I_{PPE}) 115
V_{PP} Generation Techniques 117
Directly from a 12V Regulated Supply 117
Converting from 12V Unregulated 118
Converting from a Lower Voltage 118
Converting from a Higher Voltage 120

    - General Characteristics of Voltage Converters 121
    - Totally Modular Solutions 122
- Bypass and Decoupling Capacitive Filtering 122
    - Decoupling Capacitors-V_{CC} 123
    - Bypass Capacitors-V_{CC} 124
    - Decoupling Capacitors-V_{PP} 124
- Mixed-Voltage System Design 125
    - 3.3 Volt to 5 Volt Interfaces 125
    - 5V to 3.3V Interfaces 127
    - Bidirectional Bus Interface 130
- Power Management Techniques 130
- Summary 132

Chapter Seven: Software Interfacing to Flash Memory 133
- Why Is Flash Memory Controlled By System Software? 134
    - EPROM Programming Algorithm 134
    - Flash Memory Programming 136
- The NOR Bulk-Erase Flash Memory Algorithms 139
    - The Program Algorithm 140
    - The Chip Erase Algorithm 143
    - Summary of First-Generation Programming/Erase Characteristics 148
- The NOR Fully-Automated Flash Memory Algorithms 149
    - Intel Automated Program Algorithm 151
    - Intel Automated Block Erase Algorithm 156
    - Intel Automated Erase Suspend/Resume Algorithm .. 160
    - Alternative Automated Algorithms 162
    - General Automated Algorithm Techniques-Multiple Block Erase 168
    - General Automated Algorithm Techniques-Page Programming 168
    - General Automated Algorithm Techniques-Aborting Internal Automation 168
    - General Automated Algorithm Techniques-The RY/$\overline{\text{BY}}$ Output 169
    - Software Polling or Hardware Interrupt: Which Should You Use? 169
- Update Routines 170

        Flexible Design Techniques 172
    System Boot Code Contents 173
    Software Interface to Flash Cards, SIMMs and Multi-Component Arrays 174
        Parallel Program of Non-Automated Flash Memories 176
        Parallel Erase of Non-Automated Flash Memories 179
        Parallel Program/Erase of Automated Flash Memories 183
    Summary 184
Chapter Eight: Hardware Interfacing Considerations for Flash Cards 187
    A Flash Memory Array Within a Card 187
    PCMCIA Flash Memory Cards 188
        PCMCIA 1.0 188
        PCMCIA 2.0 189
        PCMCIA Signal Definitions 193
        Host System Implementations 211
    Implementing PCMCIA 2.0 Hardware 220
        Proprietary or Commercial Interface Controllers 220
        Supporting Hardware for PCMCIA-Interface Controllers 220
        Accessing Flash Memory Cards with PCMCIA-Interface Controllers 222
        More On Buffering 222
    Summary 225
Chapter Nine: Flash Memory File Systems 227
    Introduction 227
        Flash Memory Solid-State Drive Form Factors 227
        Flash Memory Solid-State Drives Require Special Drivers 228
    Disk-Drive Basics 229
        DOS Data Structures 231
        Device Drivers 233
    Flash File System Designs 238
        Measuring Drive Usage 239
        The Disk-Drive Emulators 239
        Flash Optimized File Systems 240

The Disk-Drive Emulators 241
    Primitive Flash File Systems 242
    Full-Featured Disk-Drive Emulators 247
Flash Optimized FSSD's 252
    Accessing the Flash-Optimized FSSD 253
    Microsoft's Flash File System Design Criteria 253
    Functional Description 256
Flash File System Evaluation 265
    Performance - File Transfer Rate 265
    Performance - Clean-Up Efficiency 267
    Performance - Hot and Cold File Management 268
    Reliability - Cycle Leveling 268
    Reliability - Failure Recovery Modes 269
    System Level Issues - File System Overhead 270
    System Level Issues - Ease of Use 270
Summary 271
Chapter Ten: PCMCIA Software 273
Introduction 273
The Areas of Software Compatibility 274
The PCMCIA-ExCA Relationship 277
Flash File System Models 279
    The Original Flash File System Model 279
    What's Really Necessary? 282
Socket Services 282
    Defining the Adapter Hardware 283
    Accessing Socket Services 284
    Installing Socket Services 286
The Socket Services Functions 287
    Non-Specific Functions 289
    Adapter Functions 292
    Window Functions 304
    Fields In The I/O Window Characteristics Table 317
    Window Size 318
    Detecting Card Insertion 347
    Error Detection and Correction Functions 347
Socket Service Design Considerations and Benefits 348
The Card Information Structure 349

Accessing the Card Information Structure................ 350
The Device Information Tuple.................................. 356
Card Services.. 363
What is Card Services? 364
Do You Need Card Services?...................................... 366
Flash Card Memory Technology Drivers 366
Why Support New Cards?... 367
Flash Card Driver Functions 367
Interfacing to the Flash Card Driver 368
Installing the Flash Card Drivers 368
Summary 371
Appendix A: Flash Memory Component Vendors 373
Appendix B: Flash Memory Card/Drive Vendors 375
Appendix C: Flash Memory Component and Card Programmers....... 379
Appendix D: Component and Card Socket and Adapter Vendors....... 383
Appendix E: 12V Converters 387
Appendix F: Flash Memory Card Readers and Writers...................... 391
Appendix G: Flash File Systems.. 395
Appendix H: PCMCIA and Software Vendors 397
Appendix I: PCMCIA Compliance Testing Facilities 399
Appendix J: PCMCIA Card Types.. 401
Appendix K: PCMCIA Controller Register Functions and Vendors... 403
Appendix L: INT 21H Standard Disk-Related Functions.................... 409
Appendix M: Sample Flash File System Benchmarking Code............ 411

Figures

1.1: The Exploding Flash Memory Market 1
1.2: Flash Memory Cell Simplicity Enables Cost-Effective Manufacturing 2
1.3: Average Selling Price for 1 Mbyte of Flash Memory Storage 3
1.4: Flash Memory Satisfies Many Ideal Memory Attributes 7
2.1: BIOS Glues Common Software to Unique Hardware 17
2.2: Energy Consumed During Various Acitivties 20
2.3: Elan Flash Memory Card Reader/Writer 21
3.1: Dataquest 1992 Flash Memory Market Share (by company) 24
3.2: ETOX™ Flash Memory Cell Similarities Leverage EPROM Learning Curve 25
3.3: ETOX™ Flash Memory Cell Being Read 26
3.4: NOR Flash Memory Array Interconnect 26
3.5: ETOX™ Flash Memory Cell Being Programmed 27
3.6: EPROM Cell Being UV Erased 28
3.7: ETOX™ Flash Memory Cell Being Erased 28
3.8: Negative Gate Erase 30
3.9: Iterative Basic Flash Memory Erase Algorithm 31
3.10: EEPROM-Based Flash Memory Cell 33
3.11: EEPROM-Based Flash Memory Cell Being Erased 34
3.12: EEPROM-Based Flash Memory Cell Being Programmed 35
3.13: NAND Flash Memory Cell Being Read 36
3.14: NAND Flash Memory Array Interconnect 37
3.15: NAND Flash Memory Cell Being Programmed 37
3.16: NAND Flash Memory Cell Being Erased 38
4.1: DIP (Dual In-Line) Package Dimensions 47
4.2: DIP / TSOP Package Comparison (Actual Size) 48
4.3: LCC (Leaded Chip Carrier) Package Dimensions 50
4.4: Trace Layout Comparison: PSOP vs. PLCC 51
4.5: Small Outline J-Lead (SOJ) Package Dimensions 53
4.6: Small Outline Package (SOP) Dimensions 55
4.7: TSOP (Thin Small Outline Package) Dimensions 56
4.8: Standard and Reverse TSOP Packages 57
4.9: TSOP Serpentine Package Layout 58
4.10: SIMM Package Dimensions 60

4.11: SCM Microsystems Flash Memory SIMM Pinout 61
4.12: PCMCIA / JEIDA Type 1 PC Card Package Dimensions........... 64
4.13: PCMCIA / JEIDA Type 2 PC Card Package Dimensions........... 64
4.14: Mass Storage Architecture 66
4.15: Flash Drive Architecture 66
4.16: Design Considerations During On-Board Update........................ 69
4.17: Key Elements of In-System Update 70
5.1: Processor/Flash Memory Interface (separate address and data buses, distinct read and write, one flash memory) 74
5.2: Processor / Flash Memory Interface (multiplexed address/data lines, multiplexed read/write, two x8 flash memories) 74
5.3: V_{PP} Switch Circuit 80
5.4: Maxim MAX705, Used for V_{CC} and V_{PP} Monitoring 82
5.5: Intel 28F001BX Boot Block Flash Memory Map.......................... 83
5.6: Wired-OR RY/$\overline{\text{BY}}$ Implementation 85
5.7: Flash Memory Read Access Time Partitioning 86
5.8: AC Input/Output Reference Waveform 86
5.9: AC Testing Load Circuit 87
5.10: High Speed Input / Output Reference Waveform 87
5.11: High Speed AC Testing Load Circuit 87
5.12: AC Waveforms for Read Operations 88
5.13: AC Waveforms for Write Operation.. 89
5.14: Example Ordering Information Table... 90
5.15: Hardware Interleaving - Utilizes Common $\overline{\text{CE}}$, Unique $\overline{\text{OE}}$ and $\overline{\text{WE}}$ 100
5.16: Hardware Interleaving - State Transition Diagram.................... 103
6.1: V_{CC} Current (Typical) - Read Mode 109
6.2: V_{CC} Current (Typical) - Program Mode...................................... 111
6.3: V_{CC} Current (Typical) - Erase Mode... 112
6.4: V_{PP} Current (Typical) - Program Mode 114
6.5: V_{PP} Current (Typical) - Erase Mode 115
6.6: V_{PP} Current (Typical) - Beginning of an Erase Pulse 116
6.7: Linear Technology LT1110 5V to 12V Converter 118
6.8: Motorola MC34063A 5V to 12V Converter............................... 119
6.9: Maxim MAX732 3V to 12V Converter 119
6.10: Maxim MAX667 12V Linear Voltage Regulator 120
6.11: Linear Technology LT1111 Voltage Step Down Switcher........ 120

6.12: Interfacing a 3.3V Device to a 5V Device (TTL Inputs) 126
6.13: Interfacing a 3.3V Device to a 5V Device (CMOS Inputs) 127
6.14: Interfacing a 5V Device to a 3.3V Device 128
6.15: 5V to 3.3V Direct Interface. Overbiasing the ESD Input Diode 128
6.16: Interfacing a 5V Device to a 3.3V Device - Series Resistor Voltage Drop 129
6.17: Interfacing a 5V Device to a 3.3V Device.- 129
7.1: EPROM Programming Algorithm (Simplified Form)................. 135
7.2: Intel First Generation Flash Memory Non-Automated Programming Algorithm... 137
7.3: Intel First Generation Flash Memory Non-Automated Erase Algorithm.. 144
7.4: Intel Automated Flash Memory Program Algorithm.................. 152
7.5: Intel Automated Flash Memory Status Register 155
7.6: Intel Automated Flash Memory Block Erase Algorithm............. 157
7.7: Intel Automated Erase Suspend / Resume Algorithm................. 161
7.8: AMD 5V-Only Automated Program Algorithm.......................... 165
7.9: AMD 5V-Only Automated Erase Algorithm 166
7.10: AMD 5V-Only Automated Data Polling and Toggle Bit Algorithm... 167
7.11: Parallel Programming of Non-Automated Flash Memories....... 177
7.12: Parallel Erase of Non-Automated Flash Memories................... 180
7.13: Parallel Program / Erase of Automated Flash Memories.......... 184
8.1: PCMCIA 1.0 Flash Memory Card... 190
8.2: Intel Series 2 Flash Memory Card... 191
8.3: PCMCIA Electrical Interface Categories.................................. 193
8.4: PCMCIA Read Timing Waveform.. 195
8.5: Aliasing Caused by Inadequate Address Line Decoding 195
8.6: Internal Component Arrangement Dictated by Flash Memory Architecture 197
8.7: Byte-Wide Access Mode Circuitry for 8-Bit Systems 198
8.8: RDY/$\overline{\text{BSY}}$ Background Sequence... 201
8.9: Use of RDY/$\overline{\text{BSY}}$ in Multiple Device Operations 202
8.10: Standard PCMCIA RDY/$\overline{\text{BSY}}$ Waveform 203
8.11: High-Performance RDY/$\overline{\text{BSY}}$ Waveform for Multiple Device Operations... 204

8.12: PCMCIA Pin Lengths Allow Proper Sequencing of Card Signals 205
8.13: Example Card Detection Circuitry 206
8.14: PCMCIA Controller Chip Controls Voltage Switching 209
8.15: Mapping Memory Through an I/O Port 212
8.16: The Data Bus Generates the Flash Memory Addresses 213
8.17: Counters Enhance I/O-Mapped Read Access 213
8.18: Linearly-Mapped Memory Addressing 215
8.19: DOS Memory Map 216
8.20: Memory Paging Circuitry 217
8.21: Implementing a PCMCIA 1.0 Interface in an Embedded Application 219
8.22: The Intel 82365SL PCMCIA Interface Controller Requires a Minimal Amount of Support Circuitry 221
9.1: Flash Memory Manager and Operating System Interface 230
9.2: Disk Drive Tracks and Sectors 230
9.3: File Directory and FAT Modification 232
9.4: New Device Drivers Supersede Default Drivers 234
9.5: Using the Disk Service Interrupts to Access Disk Sectors 236
9.6: Accessing Devices Using File Handles, Not at the Sector Level 238
9.7: Flash Memory Solid-State Drive System Layers 240
9.8: Using an Interrupt Filter 242
9.9: Creating a Disk Image in Flash Memory 243
9.10: A Flash Memory Array Pre-Formatted with a Blank FAT and Root Directory 245
9.11: Sector-Level Modification Requires Considerable Overhead ... 246
9.12: One-to-One Correspondence Between FAT Entries and Sectors 248
9.13: Three-Step Cleanup Operation: Copy, Erase, and Block Renumbering 250
9.14: Defragmentation Utility Concatenates 252
9.15: Flash Memory Solid-State Drive Accessing Methods 254
9.16: Microsoft's First FFS Functioned Like a WORM Drive 255
9.17: Files are Always Written to the Next Available Free Space 257
9.18: Linked List Pointers Locate Nest File in the Chain 258
9.19: Each Subdirectory has Its Own Linked List 259
9.20: MS-Flash Performs a Three-Step Clean-up Operation 260

9.21: Worst-Case Foreground 263
9.22: File Clean-Up 264
10.1: Glue Logic Holds Together the Major System Pieces 274
10.2: Many Types of PCMCIA-Compatible Cards can Operate in the Same Socket 276
10.3: PCMCIA Provides a General, Three-Dimensional Specification Covering Processors, System Architectures, and Operating Systems 278
10.4: ExCA Provides a Specific Implementation of PCMCIA 278
10.5: The Non-Modular Flash File System Lacked Flexibility 280
10.6: Complete and Flexible Implementation of a Flash File System Consists of Five Modules 281
10.7: Use an Index and Data Register Combination to Access the PCIC's Internal Registers 284
10.8: The Functions of Socket Services Act Like a Black-Box Where Parameters Go In and Out 285
10.9: An Application Uses Interrupt 1AH to Access Socket Services or the PC's Real-Time Clock 286
10.10: Multiple Socket Services can be Chained Together and Accessed Through the Common Entry Point of INT 1AH 287
10.11: Reading a Signature from the Adapter Board to Identify Its Presence 290
10.12: Socket Services Isolates the Differences between a Removable Memory Card and Permanently Resident Flash Array 291
10.13: Five Memory-Mapped Windows for Flash Memory Card Access in an ExCA System 293
10.14: InquireAdapter Returns Information Describing the Adapter's Capabilities, such as the Power Characteristics 296
10.15: The Global Control Register Powers Down the PCIC 299
10.16: Distinguishing Between an Adapter Interrupt and a Socket Interrupt 300
10.17: Mask Status Change Interrupts at the Socket Level, Enabling Them at the Adapter Level 302
10.18: Writing a One to Bits 0-3 Enables the Corresponding Status Change to Generate an Interrupt 303
10.19: Determine the Interface Type from the Identification and Revision Register in PCIC-Compatible Controllers 306

10.20: As the Memory Window Chacteristics Table Indicates, the Base Address and Size may be Programmable 308
10.21: Contrasting Windows Divided into Pages and Multiple Windows..312
10.22: Minimum and Maximum Memory Window Address...............313
10.23: Example Showing Potential Base Address for a 4 Kbyte Window that Must Reside on a Multiple of the Window's Size..................315
10.24: Use this Register to Enable and Disable Memory and I/O Windows..320
10.25: These PCIC Registers Control the Access Speed and Determine the Stop Address of the Corresponding Memory Window...........322
10.26: These PCIC Registers Set Up the Base Address of the System Memory Window 324
10.27: These PCIC Registers Set Up the Flash Memory Card's Offset, Enables Write Protection, and Selects the Memory Plane............327
10.28: Translating System Addresses to Access Various Regions within the Flash Memory Card..329
10.29: This PCIC Register, Power Control and RESETDRV, Controls a Socket's Voltage Levels 335
10.30: The Card Status Change Register Reports on the Source of the Status Change...336
10.31: Use an I/O Port to Control the Socket's Indicators 337
10.32: Read the Interface Status Register with the GetStatus Function to Determine the Presence of a Card 339
10.33: Writing to this PCIC Register Activates the PC Card's Reset State...341
10.34: Flash Memory Card Partitioning Examples.............................351
10.35: Select between a Flash Memory Card's Common and Attribute Memory Planes using the REG Signal...351
10.36: A Tuple is the Basic Data Structure in the CIS 353
10.37: Card Services Interfaces between Clients (Applications, Device Drivers) and PC Cards, Sockets, and System Resources..............365
10.38: Installing a New Flash Card Driver 370

Tables

1.1: Flash Memory Evolution and Innovation Broaden the Application Base ... 3
1.2: Flash Memory Versatility Answers the Needs of Many Applications ... 6
3.1: NOR Flash Memory Characteristics ... 32
3.2: EEPROM Flash Memory Characteristics ... 35
3.3: NAND Flash Memory Characteristics ... 38
4.1: The Key Differences between a Flash Drive and a PCMCIA Flash Memory Card ... 66
5.1: Flash Memory Bus Interface ... 74
5.2. $\overline{\text{WE}}$-Less Flash Memory Bus Interface ... 79
5.3: JEDEC Signal/State Naming Conventions ... 91
5.4: AC Characteristics, Read Operations ... 93
5.5: AC Characteristics, Write Operations ... 97
5.6: Input/Output Capacitance ... 97
6.1. DC Characteristics ... 107
7.1: Intel Bulk-Erase Flash Memory Command Definitions ... 139
7.2: Intel Block-Erase Flash Memory Command Definitions ... 150
7.3: AMD 5V-Only Automated Algorithm Command Definitions ... 163
8.1: Signal Definition of the PCMCIA Interface ... 192
8.2: Common Memory Accesses ... 197
8.3: System Power Requirements ... 208
8.4: Voltage Sense Pin Configurations ... 211
8.5: PC Card Interface Conditions During Insertion and Removal ... 223
9.1: FAT Values for 12 and 16 Bit Entries ... 231
10.1: PCMCIA and ExCA Relationship ... 279
10.2: Socket Services Functions ... 288
10.3: Adapter Characteristics and Power Management ... 295
10.4: Example Adapter Characteristics and Power Management ... 297
10.5: Card Status Change Interrupt Steering ... 301
10.6: Interface Identification for PCIC-Compatible Controllers ... 305
10.7: Memory Window Characteristics ... 307
10.8: Example Memory Window Characteristics ... 316
10.9: I/O Window Characteristics ... 317

10.10: Controlling V_{PP} Enable Signals with the PCIC's Power Control Register....335
10.11: Tuple Format....352
10.12: Minimum Tuple Requirements....355
10.13: Sample Device Information Tuple....356
10.14: Sample Device ID Byte....357
10.15: Sample Device Size Byte....358
10.16: Sample Device Geometry Tuple....360
10.17: Sample JEDEC Identifier Table....361
10.18: Size of Field Byte....361
10.19: Sample Configuration Table....362
10.20: Sample End-of-List Tuple....363
10.21: System Memory Architectures....369

Chapter Three: Flash Memory Technologies

From a very high-level perspective, Chapter 1 answered the question, "What is Flash Memory?" As a review, flash memory has the following primary characteristics:

- Nonvolatility (retains data stored to it when powered off), and
- In-System Updateability (stored data can be erased and replaced under system processor control)

As you can see, this is a pretty broad definition! Various semiconductor vendors have chosen unique and quite dissimilar silicon technology approaches to answer the above application requirements. Some flash memory approaches are *evolutionary*, based on existing memory types that are already nonvolatile and updateable. Other technologies choose a more *revolutionary* path.

This chapter will discuss in detail three flash memory technologies: NOR, EEPROM, and NAND. All three approaches meet the basic criteria for flash memory (nonvolatility and updateability). Where they differ, however, is in their secondary characteristics, some of which are listed below[3]:

- Read Performance
- Program/Erase Performance

---

[3] Chapter 2 discussed specific flash memory applications and indicated the highest priority features in each case.

- Number of Program/Erase Cycles Through Device Lifetime
- Power Supply Voltage Requirements
- Current Draw in Device Operating Modes
- Erase Block Size

When evaluating flash memory alternatives, do not overlook the manufacturing process complexity, and the size of the flash memory cell and periphery logic. Both factors translate into component cost, and ultimately to the price you pay for the component or flash-based subsystem from the manufacturer or distributor. Keep this in mind as you read about the "latest and greatest" flash memory technology unveilings. Creating something in the laboratory is one thing; consistently recreating it in high volume and with low cost in a manufacturing facility is entirely another matter!

As a framework for the following discussion, Figure 3.1 shows the 1992 relative market share for several flash memory semiconductor vendors. The anticipated demand for flash memory in the very near future is evident, and many semiconductor companies are gearing up to supply this market.

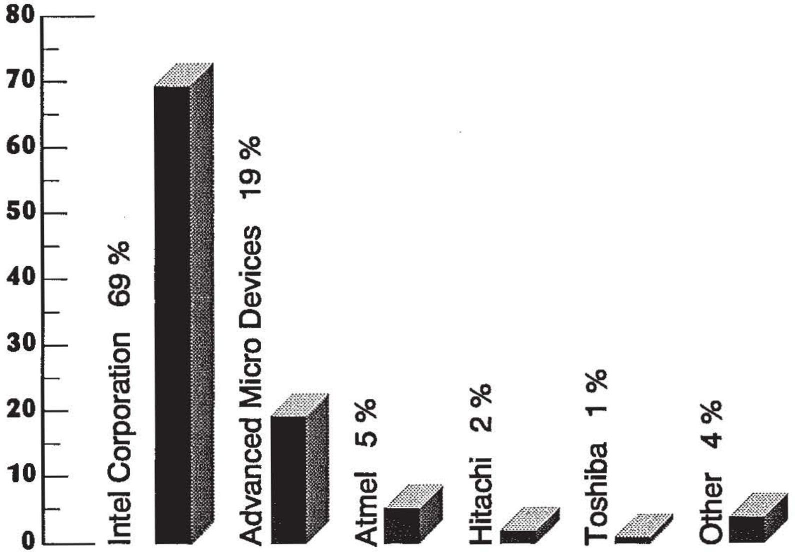


Figure 3.1: Dataquest 1992 Flash Memory Market Share (by company)

NOR FLASH MEMORY

(Examples: Intel Corporation, Advanced Micro Devices, Hitachi, Mitsubishi, NEC, SGS-Thompson, Fujitsu, Toshiba Corporation)
NOR flash memory was introduced by Intel Corporation in 1988, using the company's ETOX™ (EPROM Thin Oxide) process technology. Since that time, products based on similar technologies have been announced by several other semiconductor vendors. Figure 3.2 compares the ETOX flash memory cell with an EPROM (Erasable Programmable Read-Only Memory) cell. The similarity in this revolutionary approach is clear; NOR flash memory derives from an EPROM base. The key difference is in the silicon oxide thickness between the floating gate and substrate. This thinner oxide is the key to NOR flash memory operation; we'll see why in a moment.

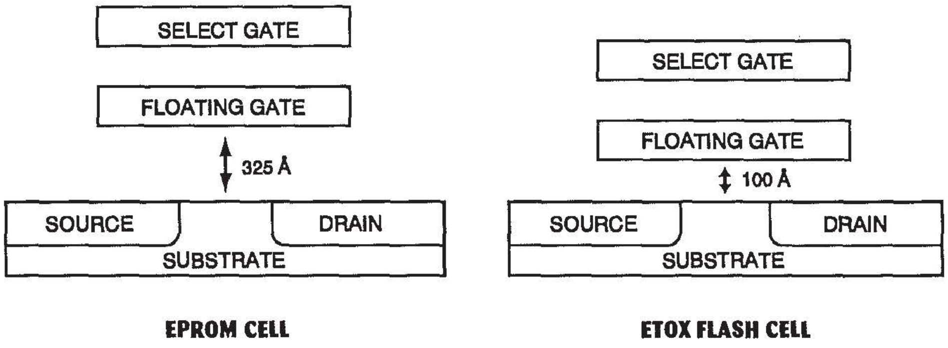


Figure 3.2: ETOX™ Flash Memory Cell Similarities Leverage EPROM Learning Curve

When shipped from the vendor, the default state of all cells in a NOR flash memory is one, corresponding to an *erased* condition. Figure 3.3 shows the voltages present on the cell when read. When erased, the floating gate of the flash memory cell does not block the cell from being turned on by the applied voltages on the select gate and drain. The resulting current is sensed at the transistor source, and translated to a one at the memory output pin.

Figure 3.4 shows a portion of a flash memory array and the interconnection of the various transistors. Device addresses enable specific wordlines and bitlines; in combination they select one transistor within the array per device output. This organization also explains the NOR name for this architecture; any "on" transistor (i.e., a selected, erased cell) in the chain results in the earlier-described current draw, sensed at the end of the chain and converted to an output one.

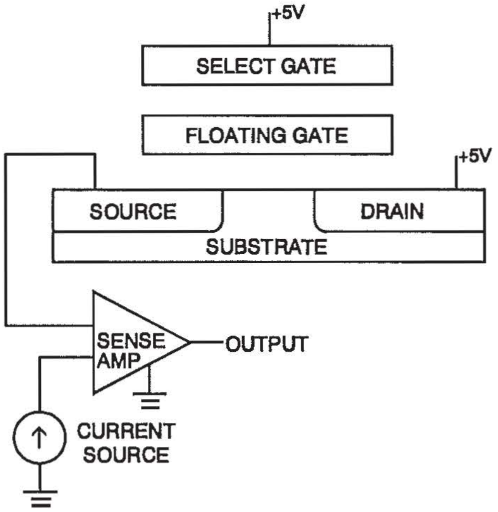


Figure 3.3: ETOX™ Flash Memory Cell Being Read

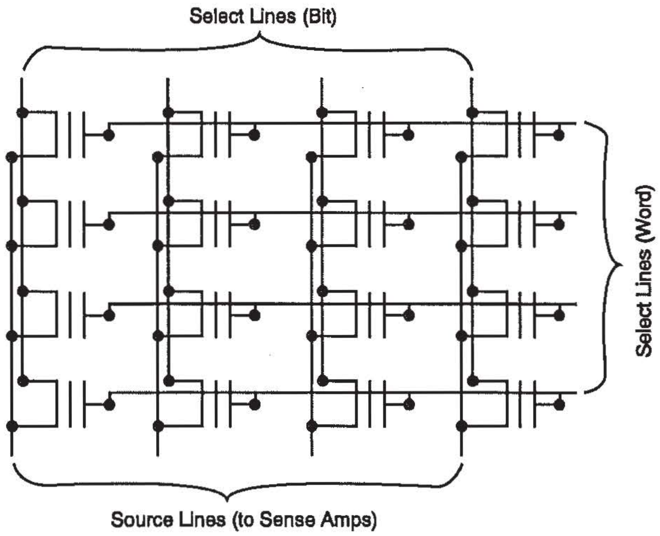


Figure 3.4: NOR Flash Memory Array Interconnect

Program

Changing a flash memory cell (or bit) to a zero is called *programming*. NOR flash memory employs the same programming mechanism as EPROM, namely hot electron injection. Figure 3.5 shows an ETOX flash memory cell being programmed. As electrons travel from the source to the drain through the substrate, the electric field generated by high voltage on the select gate causes some of the highest energy electrons to jump the gap and collect on the floating gate. What's the result? Referring back to Figure 3.3, we see that the electrons now present on the floating gate counteract the voltage on the select gate and prevent the flash memory cell from turning on. No current flows from drain to source, resulting in a zero on the memory output pin.

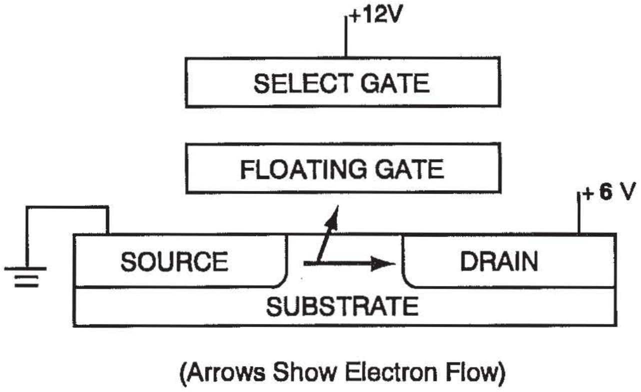


Figure 3.5: ETOX™ Flash Memory Cell Being Programmed

NOR flash memory cells can be selectively programmed to zero. In other words, programming is a *bit-level* operation. On a byte-wide flash memory device, for example, one bit of a selected byte can be programmed to zero, leaving the other seven bits at one. Later programming of the same byte can change other bits to zero in the same way. However, one key point to note about NOR flash memory (and about other flash memory approaches, too) is that *programming only changes ones to zeros*. Here lies a fundamental difference between flash memory and other rewriteable memory technologies like RAM. To change programmed zeros back into ones, we must use a different mechanism, called erase.

Erase

EPROMs are erased by ultraviolet light. As shown in Figure 3.6, the extra energy generated by UV light enables electrons on the floating gate (put there by programming) to overcome the inherent semiconductor energy potential and return to the substrate. After erasure, an EPROM cell once again reads as a one. To allow UV light to shine on all EPROM cells on a device array, the package must include a built-in glass window. As manufacturing lithographies become smaller and smaller, it becomes harder and harder to ensure that UV light can reach all array cells. The window requirement also puts limits on how small the device package can become.

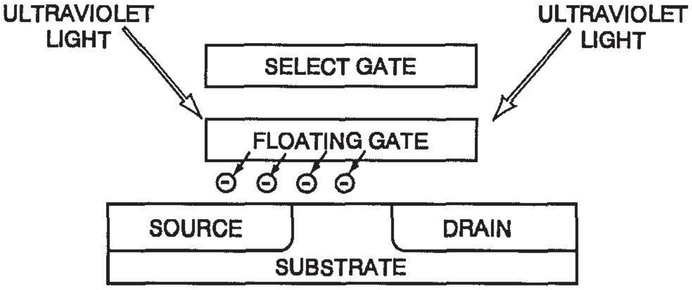


Figure 3.6: EPROM Cell Being UV Erased

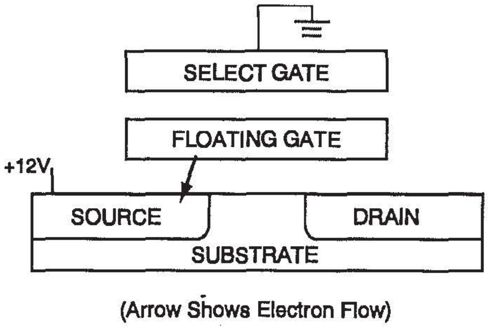


Figure 3.7: ETOX™ Flash Memory Cell Being Erased

Rather than using UV light, NOR flash memory cell erasure is accomplished electrically using a process called Fowler-Nordheim tunneling. Figure 3.7 shows the voltages on the flash memory cell during erase. The generated electric field pulls electrons from the floating gate. First generation *bulk-erasure* NOR flash memories erase all cells in the array at the same time. Second generation NOR devices erase in smaller blocks. Following the same train of thought, this is called *block erase*. Erase block size varies from flash memory vendor to vendor, and from device to device, based on the targeted applications.

Compared to EPROM, the array transistors in a flash memory need not be accessible to UV light exposure. This allows flash memory designers to run layers of interconnection over the cell versus around it, simplifying the design and minimizing the device die size. As an analogy, think of a multi-layer versus a single-layer printed circuit board. Also, flash memory does not require the window of an EPROM, allowing very small footprint (and less expensive) packaging[4].

Negative Gate Erase

Negative gate erase is similar but not identical to the conventional cell erase approach described earlier. Figure 3.8 shows the voltages on the flash memory cell during negative gate erase. Comparing this diagram with Figure 3.7, we see that although the voltages on the cells are different, the resultant voltage potential difference (and electric field) between gate and source is similar. Negative gate erase also uses Fowler-Nordheim tunneling to remove electrons from the floating gate.

Overerase

Removing too many electrons from the floating gate of a flash memory cell may theoretically result in an *overeased* condition (i.e., removing more electrons than were put there by a previous cell program). The effects of overerase are destructive to the flash memory device. Once overerased, a flash memory cell cannot be programmed again (within practical limits). Reads of this cell, as well as adjacent cells in the array, produce erratic and invalid results. Referring back to Figure 3.4, we see that an overerased cell, being "always on" even if not selected, overrides any valid data on the array transistor "chain". *Oops!*

---

[4] We'll see this again in Chapter 4.

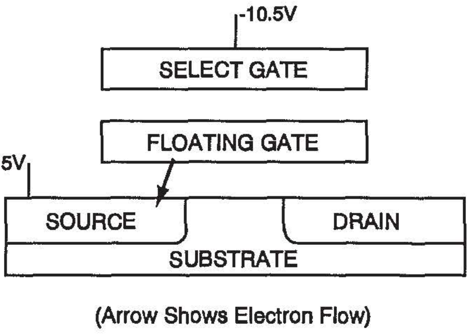


Figure 3.8: Negative Gate Erase

Fortunately, flash memory erase algorithms include built-in procedures to eliminate the potential for overerase[5]. First, cell erase (like cell programming) uses an iterative algorithm. Shown in simplified form in Figure 3.9, the built-in feedback loop ensures that the algorithm terminates and does not allow further removal of floating gate electrons once sufficient cell erase has been detected. Secondly, since all flash memory cells in a given device (or block within an device) are erased in parallel (and at approximately the same rate), preprogramming ensures that all cells are at a common initial programmed state. Without preprogramming, already-erased cells in the device or in a given erase block would be overerased while programmed cells were being erased.

Newer NOR flash memories control the erase algorithm internally, and automate both the erase preprogramming and iterative erase/verify steps. This dramatically simplifies system software algorithms and eliminates any potential for error. For more information, reference Chapter 7.

NOR Flash Memory Specifications

Table 3.1 provides a summary of NOR flash memory device characteristics, derived from data on Intel Corporation's latest-generation products. These specifications are indicative of the relative levels of performance possible today using NOR flash memory. However, exact

---

[5] If they are implemented *exactly* as published; a 'word to the wise' for system software programmers!

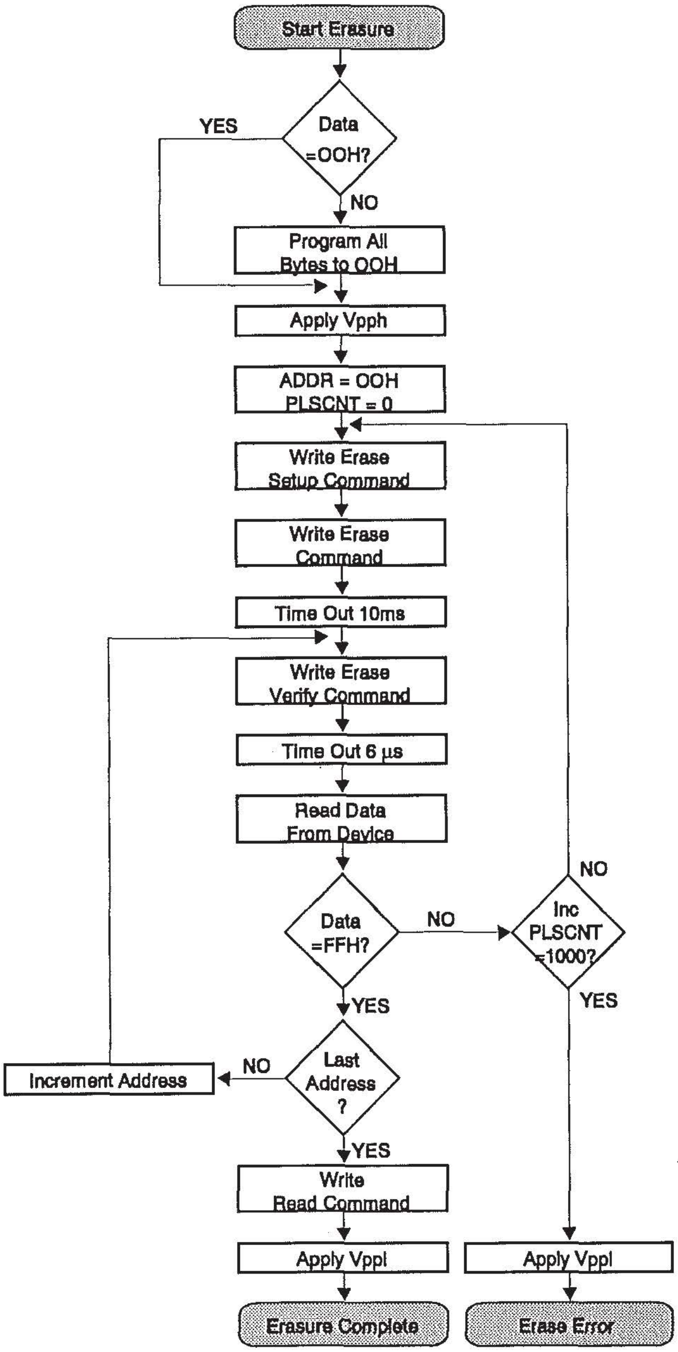


Figure 3.9: Iterative Basic Flash Memory Erase Algorithm

values will vary from device to device, from manufacturing process to manufacturing process, and from vendor to vendor[6].

Density	8 Mbit
Access Time	60 ns
Data Program Time	6 μs (min) 9 μs (typ)
Block Erase Time (64 kbyte block)	0.3 sec (min) 1.6 sec (typ)

Table 3.1: NOR Flash Memory Characteristics

Note the relatively slow erase time compared to read and program. Cell erase time is a primary function of two parameters; oxide thickness between floating gate and substrate, and internal erase voltage (it is also affected by device temperature, and by the number of times the cell has been erased previously, or *cycled*). The cell erase time of the ETOX processes is a direct result of the relatively low 12V and low current used to pull electrons from the floating gate. However, a low erase voltage also translates to excellent cell reliability and extended cycling performance. Later chapters will give examples of flash memory applications where cell erase time is (and is not) a concern, as well as discussing hardware and software techniques to hide the slow erase as a background system task.

FLASH EEPROM

(Examples: Atmel Corporation, Samsung, SunDisk, Catalyst Semiconductor)

The previous discussion showed how NOR flash memory was derived from an existing EPROM base. Similarly, flash EEPROM shares many similarities with standard EEPROMs. Figure 3.10 shows a diagram of a flash EEPROM memory cell.

---

[6] Consult vendor datasheets, application notes, and engineering reports for information on specific devices. Vendor contact information is in Appendix A.

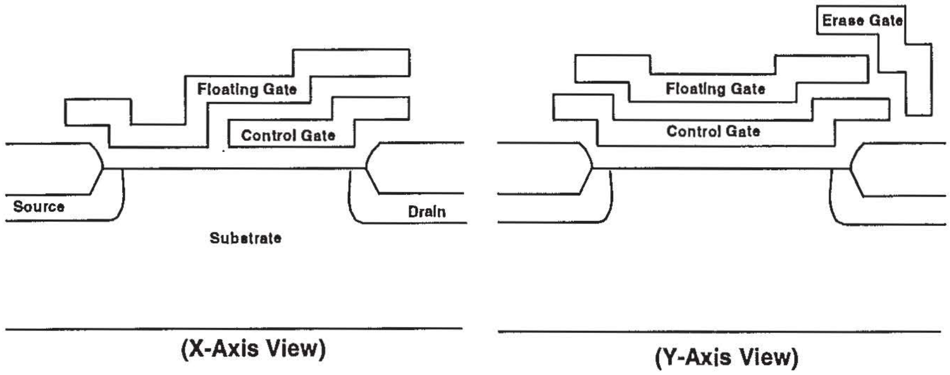


Figure 3.10: EEPROM-Based Flash Memory Cell

A standard EEPROM can be fully altered on a byte-by-byte basis. The byte erase operation is integrated in the write function, i.e., the byte is first erased and then reprogrammed with the desired data. A flash EEPROM, on the other hand, simplifies the silicon design by erasing on a block-level basis. When an EEPROM flash memory block is written, it is first erased and then programmed with data stored in an on-chip buffer.

Erase

Flash EEPROMs erase using Fowler-Nordheim tunneling, as do NOR flash memories. Most, however, use a separate erase gate per cell to collect electrons pulled off the floating gate. Regardless of the specific method, flash EEPROMs use much higher internally-generated voltages because of their greater oxide thickness compared to NOR flash, and to speed erase performance. Remember....erase is a built-in part of rewrite, not a separate operation as in the case of NOR flash. Figure 3.11 shows an EEPROM flash memory cell being erased.

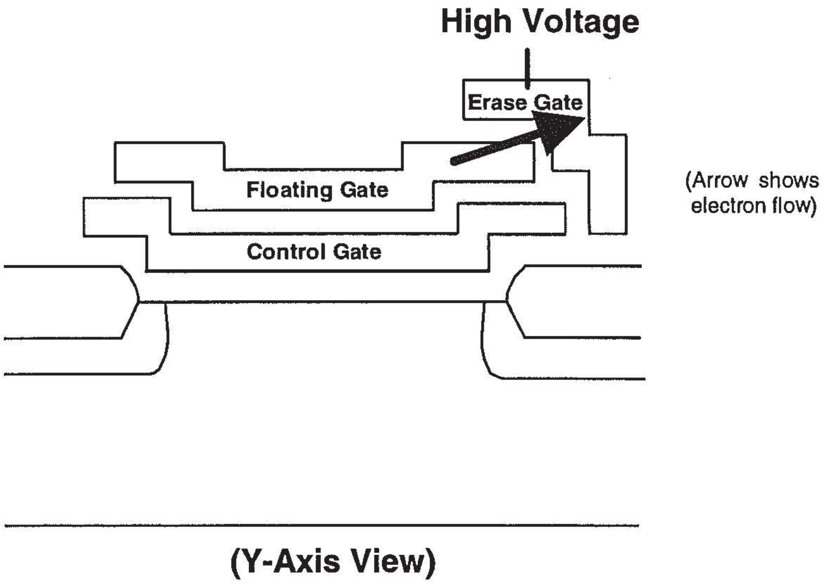


Figure 3.11: EEPROM-Based Flash Memory Cell Being Erased

Programming

Some flash EEPROMs program cells via hot electron injection. Most, however, use a reverse form of Fowler-Nordheim tunneling shown in Figure 3.12. The combination of voltages on the select gate and drain stores electrons on the floating gate, versus removing them, as seen with Fowler-Nordheim erasure. Again, high internal voltages are used for fastest programming performance.

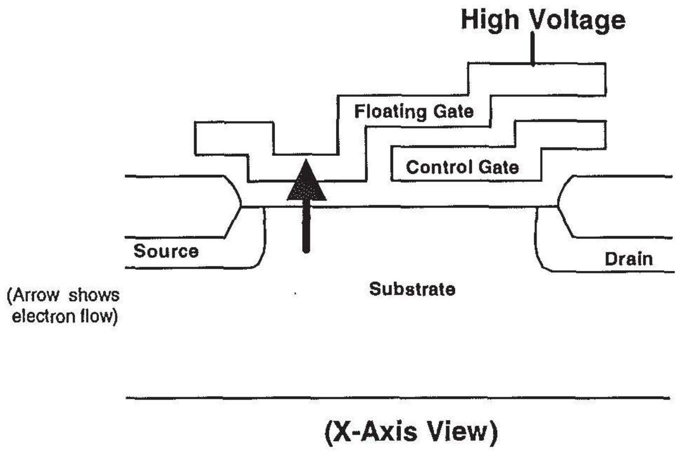


Figure 3.12: EEPROM-Based Flash Memory Cell Being Programmed

Flash EEPROM Memory Specifications

Table 3.2 summarizes flash EEPROM memory characteristics[7]. Since erase is a built-in part of the flash EEPROM program algorithm, flash EEPROMs speed up the erase process time compared to NOR flash memory, primarily via the higher internal voltages on the EEPROM cell. However, over time this may potentially have a negative impact on cell reliability. As the EEPROM cell undergoes repeated erasure, the high electrical field can break down the thin oxide region, causing failure. Some EEPROM vendors have implemented redundant cell and internal error-correction schemes to combat this "Achilles Heel".

Density	1 Mbit
Access Time	90 ns
Data Program Time	150 µs

Table 3.2: EEPROM Flash Memory Characteristics

---

[7] Taken from Atmel Corporation documentation.

NAND FLASH MEMORY

(Example: Toshiba Corporation)
NAND flash memory is a relatively new technology approach pioneered by Toshiba Corporation. As shown in Figure 3.13, the NAND flash memory cell looks very much like a NOR cell! However, the periphery logic designed into NAND is very different, and the internal program and erase approaches most closely resemble flash EEPROM methods.

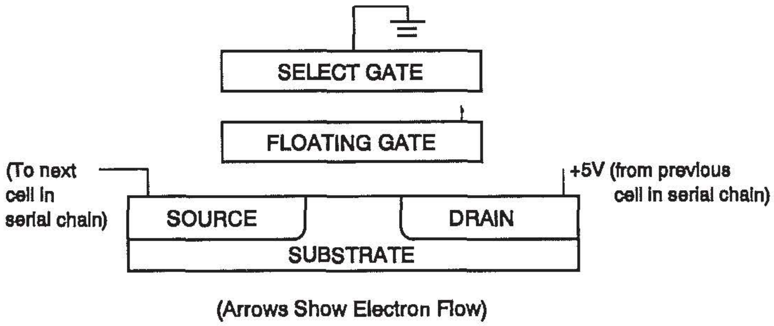


Figure 3.13: NAND Flash Memory Cell Being Read

Like Figure 3.4, Figure 3.14 shows the interconnection of transistors in a NAND array. Data sensing along the chain is serial in nature, and the architecture reflects its name.

Program and Erase

NAND flash memory cells program and erase via reverse and forward Fowler-Nordheim tunneling, respectively. Figures 3.15 and 3.16 show the internal voltages on the cell in each case. Note that unlike flash EEPROM memory tunneling, NAND flash memory applies voltages to the substrate itself, in addition to the select gate.

NAND Flash Memory Specifications

Table 3.3 shows initial specifications for Toshiba's first NAND flash memory-based device. NAND flash memory primarily targets solid state disk drive replacement applications, and the feature set reflects this, with fine-resolution blocking and fast cell erase. However note the slow initial read access time due to serial data read, which may limit broad application usage. Some NAND devices include error detection and correction (EDAC) cells and associated EDAC logic.

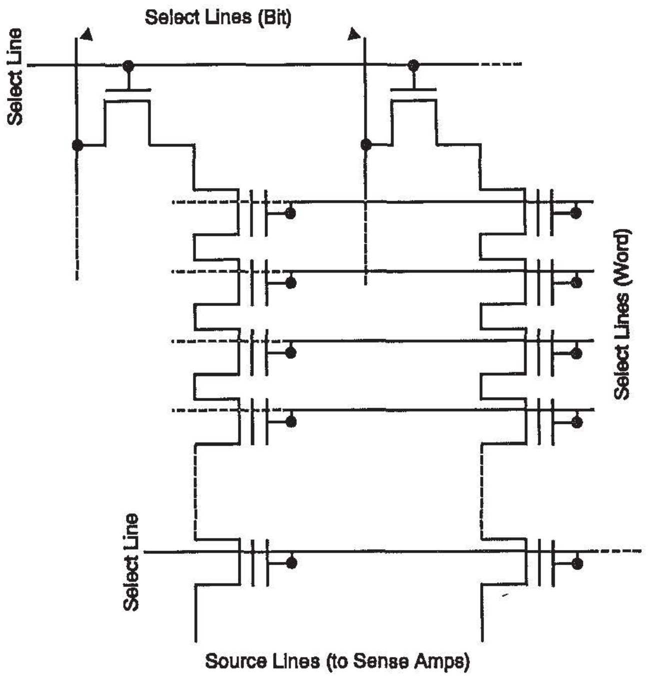


Figure 3.14: NAND Flash Memory Array Interconnect

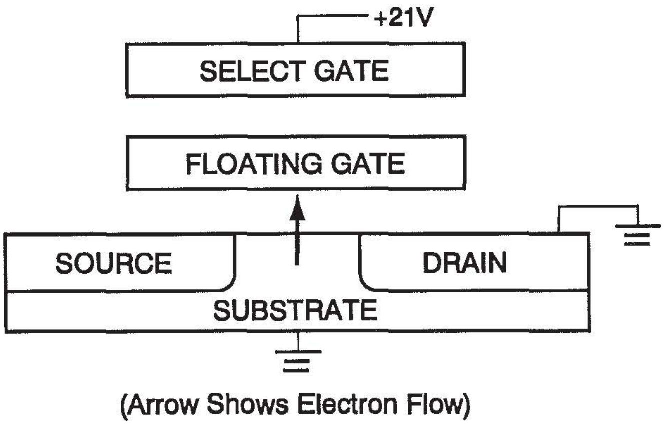


Figure 3.15: NAND Flash Memory Cell Being Programmed

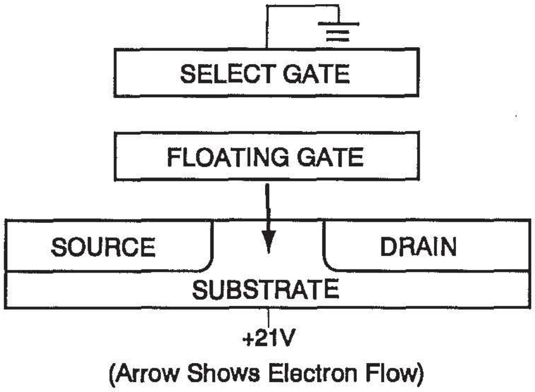


Figure 3.16: NAND Flash Memory Cell Being Erased

Density	4 Mbit
Access Time	15 µs (initial read) 80 ns (subsequent serial access)
Data Program Time	4 ms (min)
Block Erase Time (4 kbyte block)	6 ms

Table 3.3: NAND Flash Memory Characteristics

WHAT'S ALL THIS *CYCLING* STUFF, ANYWAY?

The subject of cycling is quite possibly the most abused (by companies supplying flash memory) and most misunderstood (by companies buying and using flash memory in their system designs) of any topic you'll find discussed in this book! All sorts of outlandish claims have been made, are being made, and will probably be made in the future, concerning the cycling capabilities of various flash and "flash-like" memory technologies. To confuse you even further, concepts such as MTBF (mean time before failure) are often used in conjunction with cycling specifications. Flash memory vendors often mean well (from a marketing perspective) when they include these numbers, but since an industry standard for the determination or calculation of MTBF doesn't exist, it is often subject to liberal interpretation and modification. Therefore, MTBF numbers for different flash memory devices and technologies cannot be directly compared without knowing the recipes that were used and the assumptions that were made when the measurements were taken.

We are going to explain cycling in its most fundamental definitions for you, and provide guidelines by which you can calculate your own MTBF numbers for your specific flash memory design and implementation. Our goal here is to cut through all the meaningless marketing hype and provide you with valid, useful information.

What is cycling? A cycling number is:

a) The minimum number of times a flash memory device (or block within a device) can be erased and programmed in a reasonable amount of time without loss of device functionality, at

b) A specified failure rate percentage, or FIT (failure-in-time) level.

Flash memory vendors often ignore the latter part of the above definition when publishing their cycling specifications. What good is it to know how many times you can erase an array of flash memory cells if you have no idea of the probability that some of the cells will fail before reaching this cycle count? A parallel can be drawn here with stereo equipment, where inflated claims are sometimes made of an amplifier's output power capability without mentioning how distorted the output signal was when this power was measured. What good is it to hear loud music if you can't understand it? (Of course, with some forms of popular modern music this could be seen as a positive!) Similarly, what good is it to be able to erase flash memory to an extended number of cycles if the media is essentially unusable when it reaches this cycle count? Clearly both parts of the cycling definition are valuable and useful information.

Failure Analysis

Before each flash memory device is shipped to a customer, it undergoes extensive testing to screen out known and detectable failure mechanisms both in the circuitry itself and in the manufacturing process on which the device was made. Even after this testing, it is known and accepted that a certain very small (hopefully!) number of devices will eventually fail, even when operated at all recommended specifications. Some sources of this failure, common to all flash memories as well as other memory technologies, are listed below:

- Package Integrity Failures
- Random Circuitry Failures
- Data Reliability Failures (i.e., programmed zeros turning back into ones)
- Program Failures (inability to change a one to a zero), and
- Erase Failures (inability to change a zero back to a one)

Reputable flash memory vendors spend a great deal of time and effort calculating and predicting their failure rates. Published reliability reports contain these predicted failure percentages, and are available for your inspection. We'll restrict the following discussion to the last two failures listed above, program and erase (or cycling) failures.

How and why does a flash memory cell fail due to cycling? Two different mechanisms combine here; one a more "destructive" phenomenon (oxide breakdown) and the other "non-destructive" in nature (electron trapup).

Oxide Breakdown

Notice the thin oxide region between the substrate and floating gate regions in Figure 3.10. As a flash memory cell is repeatedly erased and reprogrammed, the electrons move back and forth through the oxide region under an electric field. This stresses the oxide, and in its most severe form can result in oxide breakdown and a short circuit between oxide and substrate, rendering the cell non-functional. High quality oxide with low probability of defects, as well as a lowered electric field to minimize oxide stress, are ways that flash memory vendors can minimize the likelihood of oxide breakdown.

Electron Trapup

Recall that the earlier definition of cycling included the phrase "erased and programmed in a reasonable amount of time". This is key to the definition of electron trapup. As a flash memory cell accumulates higher and higher cycle counts, electrons become trapped in the oxide region, lowering electron mobility through the oxide and resulting in increased program and erase times. The program and erase algorithms must apply more pulses to program or erase the cell sufficiently to ensure data integrity and retention. Since the impact of electron trapup is simply a failure to program or erase within an allowed time and not a "hard" failure of the cell itself, we call it a "non-destructive" phenomenon.

Mean Time Before Failure

With cycling and failure rate data, and with a good understanding of how flash memory will be used in your system, you can calculate MTBF values for your specific design. As an example, we'll use the Intel 28F008SA 1 Mbyte FlashFile™ memory in a configuration of 20 chips (20 Mbytes total).

The Intel 28F008SA is rated for 100,000 cycles on each of its sixteen 64 kbyte blocks (independent of any other block). Data taken through 10,000 cycles shows no cycling failures, translating to a 0% cycling failure rate (pretty impressive!). Therefore, for this example we'll use the more stringent device failure rate of .01%, which encompasses *all* device failure mechanisms listed earlier in this chapter. The value 0.01% is the historic worst-case device failure rate seen with production-rated Intel flash memories, and the 28F008SA should perform at least this well (if not better).

A 0.01% failure rate (translating to 100 FITS or failures-in-time) means that fewer than 1 in 10,000 devices will fail after 10,000 cycles and 1,000 hours of operation. The scenario under which we'll calculate MTBF assumes that a 10 kbyte file is written to the 20 Mbyte array of flash memory every 10 minutes; a pretty rigorous set of assumptions if you think about it!

A flash-friendly file system could use a linked list structure to write multiple copies of a file and fill up clean flash memory, marking old versions of the file "dirty" but not erasing them immediately[8]. This significantly minimizes cycling of flash memory media. Therefore, given the file and flash memory array sizes, we can make the following calculations:

(20 Mbyte array) / (10 kbyte file) = 2,000 file writes can be done before an array erase is required
(2000 file writes/erase) x (10,000 cycles per 28F008SA block) = 20×10^6 file writes
(20×10^6 file writes) x (10 minutes/write) x (1 hr/60 minutes) = 3.33×10^6 hours MTBF

This means that our 20 Mbyte flash memory array has a Mean Time Between Failures of over 3 million hours, at a failure rate of 0.01%. Not bad, eh?

Extended Cycling-The Flash Memory Manufacturer's Options

Earlier when defining cycling, we inferred that the easiest way some flash memory vendors achieve extended cycling was by downplaying the negatives and accentuating the positives of their technology approaches. This, while true, is not the only means of reaching the extended cycling “Holy Grail”! Several other concrete tradeoffs have been made by various flash memory suppliers, both in technology and architecture, in pursuit of this goal.

Oxide breakdown can be eliminated by producing very high quality, uniform oxide for each flash memory cell. This is much more difficult than it might first appear, and in fact is probably the most complex problem that semiconductor vendors have struggled with as they attempt to ramp up their flash memory manufacturing capabilities. The oxide layer, at 100 Å thick, is made by laying down several layers of silicon *atoms*, no simple task. Remember, too, that for an 8 Mbit flash memory, not one cell but over *8 million* must be manufactured correctly to yield a functional device, and that potentially several hundred devices can be made from each 6” or 8” silicon wafer.

Another technology tradeoff can be made with respect to the internal electric field during program and erase, which is a function of the

---

[8] See Chapter 9 for more information.

magnitude of the internal voltages. A lower electric field lowers the stress on the oxide (a positive) but also slows program and erase times (a negative). Intel Corporation, with its ETOX flash memory approach, has made this choice, and has added device functionality to minimize the system performance impact of the resultant slow block erase time[9].

Where flash memories use higher internal voltages (flash EEPROM and NAND flash memories), added circuitry attempts to circumvent the impact of oxide breakdown and resultant cell damage. EEPROMs often use redundancy schemes which lower cycling failures at the expense of doubling cell size and adding complexity. Toshiba's NAND flash memory integrates error detection and correction (EDAC or ECC) directly on the silicon to mask the device impact of single cell failures. While potentially extending the cycling capability of the device, this approach adds complexity and die size to each device, and also impacts read performance.

Extended Cycling-What Can You Do?

What can you do to match the cycling requirements of your design to an appropriate flash memory architecture? First and foremost, fully analyze the cycling you truly require, and take all possible steps to minimize this cycling. A design that uses flash memory for embedded code storage may only be erased and reprogrammed ten times through its lifetime. On the other hand, a memory card used for file storage may have blocks of flash memory updated thousands or hundreds of thousands of times. Specifically with respect to file storage, Chapter 9 will explain how software companies have re-architected file storage beyond the hard drive paradigm to match the unique characteristics and capabilities of flash memory. These concepts, while possibly not directly applicable to your specific design, will provide examples of cycle minimization and management, linked list structures, and wear leveling.

In Chapter 7 we'll discuss the system software algorithms that initiate and control flash memory erase and program. In cases where erase failure has occurred due to non-destructive electron trapup, this chapter

---

[9] Upcoming chapters will discuss flash memory automation, the RY/$\overline{\text{BY}}$ output and erase suspend/resume capability.

will show you how to extend cycling by supplying the flash memory media with additional erase and program pulses.

Finally, it's your responsibility to understand the conditions under which various flash memory vendors have calculated their products' cycling capabilities, and to request additional information if needed. By correctly interpreting not only minimum cycling information but also the failure rates associated with this cycling, you can intelligently compare and choose among the many flash memory offerings in today's market, as they match the requirements of your design.

SUMMARY

The basic concept of the flash memory cell is relatively simple. Again referencing Figure 3.3 as an example, storing electrons on the floating gate changes the stored cell data from a one to a zero, and removing them changes it back to a one. The challenge for flash memory vendors has been to make flash memory:

- Simple, with the smallest possible cell and minimal periphery logic, translating to a small die size and lowest silicon cost,
- Manufacturable, with a technology development approach that can be easily and cheaply moved to the vendor's production line, and
- Feature-set-rich, with technologies and devices that answer the requirements of their target markets.

The flash memory market is still in its infancy. The system designer has a wide range of product offerings from multiple flash memory vendors to choose from, based on several unique technology approaches. In Chapter 2, we've already covered flash memory applications in detail, and discussed the features that are of highest importance in each case. In combination with the information from this chapter, you'll be able to choose the flash memory that makes the most sense for your design!

Index

27C010, 134
28F001BX, 48
28F008SA, 41
28F010, 139
28F020, 139
28F256A, 139
28F512, 139

3.3V, 125

82365SL, 220, 283

abort programming, 151
aborting internal automation, 168
access time, 4, 92
adapter characteristics, 294, 332
adapter characteristics table, 310
address decode, 75, 99, 317
address inputs, 194
address latch, 76
address space, 350
advanced client services, 280
Advanced Micro Devices, 25, 163
algorithm, 30
aliasing, 195
alterability, 5
applications, 2, 11
architecture, 2
ASCIIZ, 237, 341
Atmel, 32, 168
attribute memory, 193, 350, 356
automated algorithms, 149, 162
automated block erase, 156
automated program/erase, 183

background erase, 14
bar-code scanner, 14
basic compatibility layer, 350
battery, 6
battery life, 19
battery voltage detect, 211
benchmark, 265
bidirectional bus interface, 130
BIOS, 16, 70, 99, 171, 237, 275, 286, 348, 363
bipolar, 108
bit-alterability, 13
bitlines, 25, 115
block, 24, 33, 36, 83
block erase, 29, 69, 84, 96, 115, 150, 168
block erase algorithm, 156
block size, 29
Boot Block, 69, 82, 110, 131
boot drive, 239
bootstrap, 235
buffer, 92, 130, 221
buffering, 126, 222, 224
bulk memory services, 280
bulk-erase, 29, 115, 140, 150, 173
bus contention, 95
bus interface, 75
bus transceiver, 76, 95
bus width, 358
busy, 84
bypass capacitor, 112, 122, 124
byte erase, 33, 96
byte program, 96

caching, 98
capacitive loading, 90, 92, 99
capacitors, 122
card configuration, 361
card detect pins, 205
card enable, 198

card information structure, 193, 210, 321, 350
card insertion, 338, 347
card insertion/removal, 222
card interface, 349
card offset, 315
card physical, 349
card removal, 338
card reset, 206
card services, 275, 280, 363, 368
card status, 336, 346
card voltage, 209
CARDRV.EXE, 280
Catalyst Semiconductor, 32
$\overline{\text{CE}}$, 78
cell, 4
cell architecture, 1, 6
cell density, 6
cell erase, 5
cell failure, 43
cell redundancy, 5
cell reliability, 5
cell size, 24
cell transistor, 4, 5
cellular phone, 18
Cerquad package, 49
chaining, 233
character device, 235
chip enable, 75, 94, 99, 136
chip erase algorithm, 143
chip select, 75, 78, 94
CIS, 321, 350
CL-PD6720, 224
clean-up, 249, 259
clean-up efficiency, 267
client services, 280
client utilities, 280, 365
cluster size, 233
CMOS, 108, 123, 126, 131
command interface, 134
common memory, 193, 350, 356
comparator, 102
component cost, 24
component management registers, 206
CONFIG.SYS, 286
configuration information, 298
configuration option register, 207, 340
configuration tuple, 361
cost, 2, 4
CPU, 4
current draw, 24
current limiting, 129
current specifications, 106
current spike, 116, 123
cycle, 24, 32, 41
cycle leveling, 263, 268
cycle management, 43
cycle minimization, 43
cycling, 38, 40, 42, 43, 143
cycling delta, 265

data accumulation, 12, 14
data bus, 76, 94, 99, 196
data bus bandwidth, 76
data enable, 77
data latching, 212
data organization layer, 350
data path size, 320
data polling, 167
data verification, 138
data/lookup tables, 14
Databook, 224, 344
DB86082, 221
DC-DC converter, 208
DDE, 241, 247
debugging, 141
decoupling capacitor, 112, 122
deep powerdown, 82, 131
deep powerdown mode, 110, 114
defragmenting, 251
$\overline{\text{DEN}}$, 77
device addresses, 25
device delays, 99
device density, 5
device driver, 233
device geometry, 358
device ID, 356
device information tuple, 316
device package, 28
device speed code, 316
device temperature, 32
die size, 43
digital cellular phone, 18
DIP package, 46
direct-read, 5
directory, 231, 239
dirty sectors, 249
disk drive, 5, 229
disk drive emulator, 66, 239
disk drive template, 244
disk imaging, 243
domains, 229

double-word, 76
drain, 25
DRAM, 2, 4, 5, 98
driver installation, 369

EDAC, 36
EEPROM, 3, 5, 13, 23, 32, 43, 106, 163
electric field, 5, 27, 29, 40, 42, 113
electron charge, 5
electron mobility, 41
electron trapup, 41, 43
embedded algorithms, 163
embedded code, 16
energy consumption, 114, 131
EPROM, 2, 5, 16, 25, 48, 49, 71
EPROM programming, 134
erase, 27, 30, 36, 39, 41, 42, 43
erase block size, 116, 358
erase confirm, 145
erase current, 116
erase mode, 112, 115
erase performance, 33
erase pulse, 116
erase setup, 145
erase suspend, 71
erase suspend/resume, 160
erase time, 32, 170
erase verify, 30, 146, 181
erase voltage, 32, 79, 113
erased condition, 25
error detection, 214
error detection and correction, 5, 36, 43, 347
ESD protection, 127
ETOX, 2, 25, 32, 43, 106
ExCA, 207, 220, 277, 285, 287, 298, 348
Exchangeable Card Architecture, 207
execute-in-place, 228, 304
extended cycling, 43

failure mechanisms, 40, 41
failure rate, 39, 40, 41, 42
failure recovery, 269
fanout, 108
FAT, 231, 239, 244, 258
FCB, 237
FDD, 20
feature comparison, 12
field size, 361
file allocation table, 231
file control blocks, 237
file storage, 43
file system, 42
file usage, 239
firmware, 16
FIT, 39, 41
flash card driver, 277, 367
flash drive, 66
flash file system, 227, 274, 279, 368
flash memory card, 13, 20, 57, 62
flash memory sources, 172
flash optimized file system, 240
flash specifications, 86
FlashFile, 82, 110, 131
flight recorders, 13
floating gate, 25, 29, 32, 40, 79, 113, 115
foreground clean-up, 262
forward-biasing, 129
Fowler-Nordheim tunneling, 28, 33, 36
FSSD, 227
Fujitsu, 25, 214
full erase, 116

garbage, 250
GetAdapter, 302
GetAdapterCount, 289
GetPage, 312
GetSocket, 331
GetSSInfo, 290, 342, 344
GetStatus, 340, 345, 346
GetVendorInfo, 343
GetWindow, 309
glitch detect, 78
glue-logic, 273

handheld instrumentation, 14
hardware interfacing, 73, 187
hardware interleaving, 99
hardware interrupt, 170
HCT, 126
HDD, 15
high-speed specifications, 90
Hitachi, 25, 168
hold time, 96
hot electron injection, 27, 34
hot insertion, 62

I/O access, 212
I/O mapping, 214, 293
I/O space, 304
IDE, 66, 228

identification and revision register, 305
IDT, 130
in-system write, 71
indirect indexing, 283
input voltage specifications, 126
InquireAdapter, 292, 299, 303, 310, 325, 330, 334
InquireSocket, 333, 335, 336
InquireWindow, 304, 318, 319, 321, 322, 325, 326, 357
installable device driver, 233
INT 13H, 235, 243
INT 1AH, 285
INT 21H, 237, 253, 265
INT 25H, 236, 243
INT 2FH, 253
integrated converter, 122
Integrated Device Technology Corporation, 130
integrated drive, 228
integration, 133
Intel, 25, 149, 189
interface controller, 220, 283
interface logic, 102
interface status register, 283
interleaving, 14, 95, 99, 103, 359
internal automation, 82, 151, 163
internal voltage, 5, 43
internal voltage conversion, 113
interrupt filter, 206, 242
interrupt latency, 203
interrupt request, 200
interrupts, 141, 225, 299, 333, 345
IO.SYS, 233
ISA, 330

J-Lead, 49
JEDEC, 69, 91
JEDEC identifier, 360
JEIDA, 62, 188
JIT, 70
just-in-time, 70

laser printer, 15
LCC package, 49
leakage current, 114
lifetime, 24
linear mapped memory, 214
Linear Technology Corporation, 118
linked list, 258
linked list structures, 43
lithographies, 28
locality, 100
lockout voltage, 78

M-Systems, 247
magnetic mass storage, 3, 5, 6, 19
manufacturability, 2
manufacturing, 24, 42
market share, 24
mass file storage, 19
MAX705, 81
Maxim Integrated Products, 119
MDT4P05, 80
medical instrumentation, 12
memory banks, 359
memory card, 62, 187
memory card offset, 328
memory characterisitcs table, 305
memory comparison, 6
memory density, 13
memory lifetime, 5
memory mapping, 212, 214
memory output, 25
memory space, 304
memory technology driver, 277, 281
metaformat, 349
microcontroller, 219
Microsoft, 252, 253
Mitsubishi, 25
mixed-voltage bus, 130
mixed-voltage design, 125
modular solutions, 122
monolithic file system, 279
MOSFET, 130
Motorola, 77, 80, 118
MS Flash File System, 252
MS-FLASH.SYS, 279
MTBF, 38, 41, 42
MTD, 277
multiple block erase, 168
multiplex interrupt, 253
multiplexing, 76

NAND, 23, 36, 43, 106, 168
NEC, 25, 168
negative gate erase, 29
non-destructive failure, 41
nonvolatile, 4, 6
nonvolatility, 16, 23
NOR, 23, 26, 30, 80, 106

$\overline{\text{OE}}$, 77, 78
on-board update, 69
on-chip buffer, 33
open-drain output, 130
operating range, 90
operating voltage, 106
OTP-ROM, 49
output buffer, 99, 109
output bus, 95
output drive, 92
output enable, 76, 199
output enable time, 92
output loading, 99
overerase, 29
oxide, 25, 32
oxide breakdown, 5, 40
oxide thickness, 33, 35

packaging, 45
page, 102
page enable, 326
page programming, 168
paged memory mapping, 217
paging, 222
parallel erase, 175, 179
parallel programming, 175, 176
PBX switcher, 14
PC card interface controller, 220
PCIC, 283, 298, 305, 323
PCMCIA, 20, 62, 187, 210, 211, 219, 220, 223, 277, 285, 287, 305, 348, 349, 354, 363
PCMCIA card types, 63
PCMCIA-ATA, 66, 228
performance, 14, 102, 265
    program/erase, 23
    read, 23
Performance Semiconductor, 130
periphery logic, 24, 36
$\overline{\text{PGM}}$, 136
pinout, 73
platter seek, 5
PLCC, 49
point of sale terminals, 13
portability, 13
portable computer, 20
power, 5, 14
power consumption, 80, 82, 105, 113, 130, 132
power management, 108, 294, 296
power management table, 334
power management techniques, 130
power profile, 113
power transitions, 83
powerdown, 82, 110
powerdown mode, 114
POWERGOOD, 84
pre-condition, 143
preprogramming, 30, 143
pricing, 2
processor loading, 76
production volume, 1
program, 41, 42, 43, 96
program mode, 111, 115
program setup, 140
program verification, 151
program verify, 115, 140, 141, 142
program voltage, 79, 112, 113
program/erase performance, 23
program/erase voltage, 138
programming, 27
programming algorithm, 138
programming current, 115
programming voltage, 207
PROM, 4
PROM programmer, 70
PROM programming, 68
protected mode, 344
PSOP, 51
pulse count, 140
$\overline{\text{PWD}}$, 69

RAM, 3, 77
RAM interface, 71
RAMDRIVE, 228
ramp delay, 132
READ, 78
read, 76, 103, 114
read access, 99, 101
read access time, 36, 99
read array mode, 161
read block size, 358
read current, 108
read current profile, 108
read cycle time, 93
read delay, 94, 102
read mode, 108
read performance, 23, 43, 92
read status register, 160
read timing, 93
read/write, 77
ready/busy, 84, 200
real-time clock, 141

reclamation, 247
redirector, 253
redundancy, 5, 43
refresh, 6
$\overline{\text{REG}}$, 193
register-based memory mapping, 212
reliability, 32, 35, 40, 99, 113, 117, 122, 128, 268
remote sensing, 14
removability, 291
$\overline{\text{RESET}}$, 81, 206
reset, 82, 169, 173, 340
resource management, 280, 364
reverse pinout, 57
rewrite, 5, 33
ROM, 3, 16, 77
ROM drive, 243
ROM scan, 286
root directory, 244
rotation delay, 5
RP, 69
RY/$\overline{\text{BY}}$, 82

Samsung, 32
saturation, 108
SCM, 247
sector, 229, 239, 246
select gate, 25, 36
sensing instrumentation, 14
Series 1 Card, 139, 188
Series 2 Card, 200, 203, 206
serpentining, 57
SetAdapter, 296, 303, 309, 333, 345, 347
SetPage, 313, 315, 329
SetSocket, 301, 337, 347
setup command, 151
setup time, 95
SetWindow, 298, 314, 323, 326
SGS-Thompson, 25
shadowing, 98
shock tolerance, 13
signal states, 91
silicon, 42
silicon oxide, 25
SIMM package, 59
socket characteristics, 332
socket configuration, 337
socket services, 220, 222, 281, 284, 348
socket windows, 293
software algorithms, 43
software delay, 141
software delay loop, 81
software interface, 174
software metaformat, 349
software polling, 170
SOJ package, 54
solid-state drive, 227
SOP package, 54
source, 25, 29
spare block, 249
specifications, 90, 106
SRAM, 4, 5, 13, 19, 98
standby, 114, 131
standby mode, 109
state machine, 151, 161
status change interrupt, 282
status register, 84, 151, 153, 158, 160, 169
stop timers, 141, 146
substrate, 25, 40
SunDisk, 32
supply voltage, 106
suspend/resume, 160
switching, 108
system cost, 99
system performance, 98
system reset, 82

TCIC-2/N, 221, 224
temperature tolerance, 13
test conditions, 108
testing, 349
Texas Instruments, 130
timing parameters, 85
toggle bit, 167
Toshiba, 25, 36, 168
transceiver, 76, 92, 95
transistor, 4
transistor gate, 115
transistor source, 25, 115
transition points, 90
tri-state, 130
TSOP package, 48, 54, 69
TTL, 108, 125
tunneling, 28
tuple, 316, 352, 361
tuple byte, 354
tuple chain, 354
tuple code, 353
tuple field, 354
tuple link, 353
tuple list, 353, 354

ultraviolet light, 28
update, 68, 170
update methods, 45
update performance, 113
updateability, 6, 23
UV, 28

V_{CC}-only, 106
VDISK, 228
vendor release, 342
verify, 143, 178
verify voltage, 113
virtual addressing, 249
voltage conversion, 113, 118
voltage converter, 79
voltage converter features, 121
voltage droop, 118
voltage monitoring, 81
voltage ramp, 81, 115
voltage ramp rates, 129
voltage requirements, 24
voltage sense, 210
voltage spikes, 122
voltage switching, 80, 208
voltage tolerance, 117
voltage translation, 126, 130
volume growth, 1
V_{PP} feedback, 81

$\overline{\text{WAIT}}$, 204
wait states, 102, 321
wakeup, 110, 131
$\overline{\text{WE}}$-less memory, 78
wear leveling, 43
window, 304, 306
window base address, 308, 310, 322
window enable, 319
window number, 325
window page, 311, 325
window size, 309, 314, 318
window state, 319
wire or, 203
word, 76
wordlines, 25, 115
WORM, 253
wrap-around, 195
$\overline{\text{WRITE}}$, 78
write, 33, 103
write cycle time, 95
write enable, 77, 199
write performance, 131
write protect, 78, 199, 312, 326, 357
write state machine, 151

XIP, 228, 304

You are welcome to send us comments or questions concerning this or other Annabooks products, or to request a catalog of our products and seminars.

Annabooks
11848 Bernardo Center Drive, Suite 110
San Diego, CA 92128

616-673-0870

1-800-462-1042

616-673-1432 FAX