




# MC68322

## Integrated Printer Processor User's Manual

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters can and do vary in different applications. All operating parameters, including "Typicals" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and  are registered trademarks of Motorola, Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

Windows Printing System™ is a registered trademark of Microsoft Corporation.  
PostScript® is a registered trademark of Adobe Systems, Inc. Centronics® is a registered trademark of  
Centronics Inc. All other trademarks are the property of their respective owners.

## ABOUT THIS MANUAL

The *MC68322 Integrated Printer Processor User's Manual* contains information about the programming capabilities, registers, and overall operation of the MC68322 device.

### CONVENTIONS

The following conventions should help you navigate through this manual. Anything that is not on this list is in plain text.

- Signals, pins, lines, and bit names appear in uppercase text.
- Register acronyms appear in uppercase text, but their full names are in lowercase.
- Cross references appear in initial-cap bold text.
- Instructions appear in lowercase bold text.

All acronyms and mnemonics are defined the first time they appear in each section. The easiest way to start using this manual is to use the index to find the topic you're interested in.

### SUPPLEMENTAL DOCUMENTATION

There are two manuals available from Motorola that will enable you to have a more well-rounded reference source for the MC68322. To order them, see the back cover of this manual for the Motorola Literature Distribution Center contact information [or click here to go to the LDC website](#).

- The *M68000 Family Programmer's Reference Manual* (M68000PM/AD) provides instructions and detailed information on the EC000 core and other devices.
- The *MC68322 Integrated Printer Processor Product Brief* (MC68322P/D) provides a brief description of the MC68322's capabilities.

### GIVE US YOUR OPINION

We are constantly trying to make our documentation easier to access and use, so please give us your feedback. You can either print out and send us the form on the following page or fill out the survey on the web at [http://www.mot.com/SPS/ADC/site/docs\\_survey.html](http://www.mot.com/SPS/ADC/site/docs_survey.html). You can also visit the Motorola Imaging and Storage Division website at <http://www.mot.com/isd> for information about applications, errata, and other products. This manual is also available in PDF format at that site.

**MOTOROLA**  
**IMAGING AND STORAGE DIVISION**  
**CUSTOMER DOCUMENTATION SURVEY**

Fill out this form and fax it to the ISD Information Development team at (512) 891-8593.

Title of Manual: \_\_\_\_\_

1. Did the information in this document appear to be organized in a logical manner? \_\_\_\_\_
2. Was the level of writing appropriate for you as a user of this Motorola product? \_\_\_\_\_
3. Were the illustrations and graphics clear and easy to understand? \_\_\_\_\_
4. Some information, such as signal summaries, may have been duplicated in other sections for the purpose of making it easier for you to use. Did you find it useful? \_\_\_\_\_
5. Did any of the technical information assume too much prerequisite knowledge? \_\_\_\_\_
6. If you answered yes to #5, do you need more task-oriented information? \_\_\_\_\_
7. Were there enough example applications in the manual? \_\_\_\_\_
8. The Information Development team is considering the placement of documentation on CD-ROM in the future. Does your computer have a CD player? \_\_\_\_\_
9. What information should we add to the next version of this document?

10. What information should we delete from the next version of this document?

11. Was there technical information in this document that you have a question about? Explain.

Thank you for your comments. They will be used to improve this and other Motorola customer documentation.

Name: \_\_\_\_\_

Title: \_\_\_\_\_

Company: \_\_\_\_\_

Address: \_\_\_\_\_

City/State/Zip/Country: \_\_\_\_\_

Email Address: \_\_\_\_\_

Phone Number: \_\_\_\_\_

# TABLE OF CONTENTS

Paragraph Number	Title	Page Number
<b>Section 1</b>		
<b>Introduction</b>		
1.1	Features .....	1-2
1.2	Processors and Modules .....	1-3
1.2.1	The EC000 Core .....	1-3
1.2.2	Graphics Unit .....	1-4
1.2.3	Bus Interface Unit .....	1-5
1.2.4	System Integration Module .....	1-5
1.2.5	DRAM Controller .....	1-6
1.2.6	DMA Interface .....	1-6
1.2.7	Parallel Port Interface .....	1-6
1.3	Internal Memory Map .....	1-6
1.4	Understanding the MC68322 .....	1-8
1.4.1	Printer Languages .....	1-8
1.4.2	Bitmap .....	1-9
1.4.3	Banding .....	1-10
1.4.4	Halftoning .....	1-10
1.4.5	Duplex Printing .....	1-11
<b>Section 2</b>		
<b>Signal Descriptions</b>		
2.1	Address Bus .....	2-3
2.2	Data Bus .....	2-3
2.3	System Interface .....	2-4
2.3.1	Reset (RESET) .....	2-4
2.3.2	System Clock .....	2-4
2.3.3	High Impedance Mode .....	2-5
2.4	External Bus Master Interface .....	2-6
2.5	DRAM Interface .....	2-7
2.6	DMA Interface .....	2-8
2.7	Printer Communication Interface .....	2-8
2.8	Print Engine Video Controller Interface .....	2-8
2.9	Parallel Port Interface .....	2-9

# TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
<b>Section 3</b>		
<b>The Core</b>		
3.1	Programming Model .....	3-1
3.2	Data Types and Addressing Modes .....	3-3
3.3	Instruction Set Summary .....	3-4
<b>Section 4</b>		
<b>Bus Operation</b>		
4.1	EC000 Core Read Cycle .....	4-1
4.2	EC000 Core Write Cycle .....	4-4
4.3	Interrupt Acknowledge Bus Cycle .....	4-6
4.4	Reset Operation .....	4-8
4.5	External Bus Master .....	4-9
4.5.1	MC68322 Bus Arbitration .....	4-9
4.5.2	External Bus Master Read Cycle .....	4-10
4.5.3	External Bus Master Write Cycle .....	4-11
4.5.4	Illegal Address Interrupt .....	4-12
<b>Section 5</b>		
<b>Interrupt and Exception Handling</b>		
5.1	Internal Interrupts .....	5-1
5.1.1	Hardware Interrupts .....	5-2
5.1.2	Software Interrupts .....	5-3
5.2	External Interrupts .....	5-4
5.3	Timer Module .....	5-6
5.4	Core Exception Handling .....	5-7
5.4.1	Processing Specific Exceptions .....	5-10
5.4.2	Multiple Exceptions .....	5-13
5.4.3	Exception Bus Cycles .....	5-14
5.5	Module Soft-Reset Register .....	5-14
<b>Section 6</b>		
<b>System Integration Module</b>		
6.1	Chip-Select Registers And Banks .....	6-1
6.2	Synchronous and AsynchronouS Chip-Select Access Timing .....	6-4

# TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
<b>Section 7</b>		
<b>DRAM Controller</b>		
7.1	DRAM Registers and Banks .....	7-1
7.1.1	Base Address and Size Fields .....	7-1
7.1.2	ROM Mode .....	7-2
7.1.2.1	Functional Description .....	7-2
7.1.2.2	Timing Example .....	7-3
7.1.2.3	Address Demultiplexing Circuit .....	7-4
7.1.2.4	Operational Example .....	7-4
7.2	DRAM Control Register .....	7-5
7.3	DRAM Timing Modes .....	7-5
7.4	DRAM Accesses .....	7-6
7.4.1	DRAM Refresh Cycles .....	7-6
7.4.2	DRAM Read Cycles .....	7-7
7.4.3	DRAM Write Cycles .....	7-8
7.4.4	DRAM Bus Arbitration .....	7-9
7.4.5	DRAM Burst Accesses .....	7-10
7.5	Power-Up Sequence .....	7-10
<b>Section 8</b>		
<b>DMA Interface</b>		
8.1	DMA Configuration Registers .....	8-2
8.1.1	Transfer Address Fields .....	8-2
8.1.2	Transfer Count Fields .....	8-3
8.1.3	Flush Request Fields .....	8-3
8.2	GDMA Control Register .....	8-3
8.3	DMA Speed Register .....	8-4
8.4	DMA Interrupt Event Registers .....	8-5
8.5	Initiating A DMA Operation .....	8-6
8.6	DMA Transfers .....	8-6
8.6.1	PDMA Transfers .....	8-7
8.6.2	GDMA MC68322 Bus Read and Write Cycles .....	8-7
8.6.3	GDMA DRAM Bus Read and Write Cycles .....	8-8
8.7	DMA Transfer Termination .....	8-9
8.7.1	Normal Termination .....	8-9
8.7.2	Bad Address Termination .....	8-10
8.7.3	Core-Forced Termination .....	8-10

# TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
------------------	-------	-------------

## Section 9 Parallel Port Interface

9.1	PPI Registers .....	9-2
9.1.1	Parallel Port Interface Register .....	9-2
9.1.2	Parallel Port Control Register .....	9-4
9.1.3	PPI Interrupt Event Register .....	9-6
9.2	Hardware Handshaking .....	9-8
9.2.1	Compatibility Handshaking .....	9-8
9.2.2	ECP Handshaking .....	9-9
9.2.2.1	Command Byte Detection .....	9-9
9.2.2.2	RLE Decompression.....	9-10
9.2.3	Disabling Hardware Handshaking .....	9-10
9.3	Software-Controlled Handshaking .....	9-11
9.4	Digital Filtering .....	9-11
9.5	Error Cycles .....	9-12
9.6	Parallel Port Data Bus Latching .....	9-13
9.7	PPI on Reset .....	9-14
9.8	PPI Data Transfer Rate .....	9-14

## Section 10 Print Engine Interface

10.1	Print Engine Interface Registers .....	10-2
10.1.1	Printer Communication Register .....	10-2
10.1.2	PVC Control Register .....	10-3
10.1.3	Printer Control Block Register Set .....	10-5
10.1.4	PVC Interrupt Event Register .....	10-6
10.1.5	Printer Communication Interrupt Event Register .....	10-8
10.2	Printer Communication Protocol .....	10-8
10.3	Print Engine Interface Operation .....	10-9
10.3.1	Synchronous/Asynchronous PVC Operation .....	10-10
10.3.2	Command Operation .....	10-11
10.3.2.1	CCLK Supplied By MC68322 .....	10-11
10.3.2.2	CCLK Supplied By Print Engine .....	10-12
10.3.3	Status Operation .....	10-13
10.3.3.1	CCLK Supplied By MC68322 .....	10-13
10.3.3.2	CCLK Supplied By Print Engine .....	10-14
10.3.4	PLL Video Clock Divisor .....	10-15
10.4	PVC On Reset .....	10-16
10.5	PVC Video Data Timing .....	10-16



# TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
10.5.1	1X Video Clock(PVCCR Bit 0 = 0) .....	10-17
10.5.2	VCLK Rising Edge (PVCCR Bit 1 = 11) .....	10-17
10.5.3	Border Polarity High (PVCCR Bit 5 = 0) .....	10-17

## Section 11 RISC Graphics Processor

11.1	RGP Registers .....	11-2
11.1.1	RGP Start Register .....	11-2
11.1.2	RGP Diagnostic Register .....	11-2
11.1.3	RGP Interrupt Event Register .....	11-2
11.2	RGP Basic Operation .....	11-3

## Section 12 Graphic Operations

12.1	Types of Bitmaps .....	12-1
12.2	Graphic Operands .....	12-2
12.3	Types of Graphic Operands .....	12-3
12.4	Boolean Codes .....	12-3
12.5	Bit Block Transfers .....	12-5
12.6	Scanline Transfers .....	12-5
12.6.1	Scanline Tables and Bit String Specifiers .....	12-6
12.6.2	Scanline Run Operation .....	12-8
12.6.3	Executing During Banded Applications .....	12-9
12.6.4	Halftone Companion Tables .....	12-10
12.7	Scanline and Halftone Table Example .....	12-13
12.8	BitBLT and Scanline Order Execution .....	12-14
12.9	Location and Address Constraints .....	12-15

## Section 13 Graphic Orders

13.1	Types of Graphic Orders .....	13-1
13.1.1	Initialization .....	13-1
13.1.2	Program Flow Control .....	13-3
13.1.3	Bit Block Transfer .....	13-3
13.1.4	Expanded Bit Block Transfer .....	13-3
13.1.5	Scanline Transfer .....	13-4
13.2	Sequence of the Display List .....	13-5
13.3	Graphic Order Addresses .....	13-5
13.3.1	Physical vs Logical Address .....	13-6

# TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
13.3.2	Duplex Addresses .....	13-6
13.4	Band Number and Band Faults .....	13-7
13.5	Graphic Order Descriptions .....	13-8

## Section 14 Electrical and Thermal Characteristics

14.1	Maximum Ratings .....	14-1
14.2	Thermal Characteristics .....	14-1
14.3	DC Electrical Specifications .....	14-2
14.4	AC Electrical Specifications .....	14-3
14.4.1	Clock and Reset Timing .....	14-3
14.4.2	MC68322 Bus Timing .....	14-4
14.4.3	DRAM Timing .....	14-12
14.4.4	IDMA Timing .....	14-13
14.4.5	Print Engine Interface Timing .....	14-14
14.4.6	Interrupt Timing .....	14-16
14.4.7	Parallel Port Interface Timing .....	14-17
14.4.8	External Bus Master Timing .....	14-18

## Section 15 Ordering Information and Mechanical Data

15.1	Ordering Information .....	15-1
15.2	Pin Assignment .....	15-1
15.3	Mechanical Data .....	15-3

## Appendix A In-Circuit Emulation Interface

A.1	ICE Interface Signals .....	A-1
A.1.1	ICE Signal Descriptions .....	A-2
A.2	ICE Adaptor Board Design .....	A-4
A.3	ICE Adaptor Board Schematics .....	A-6
A.3.1	In-Circuit Emulation Interface .....	A-12
A.4	ICE Pin Assignment .....	A-16

# TABLE OF CONTENTS (Continued)

Paragraph Number	Title	Page Number
<b>Appendix B Applications</b>		
B.1	Configuring The MC68322 .....	B-1
B.2	Configuring The DRAM and DRAM SIMM .....	B-2
B.3	Configuring The Flash EPROM .....	B-4
B.4	Configuring The Random Control Logic .....	B-7
B.5	Configuring The Serial EEPROM .....	B-8
B.6	Configuring The In-Circuit Emulation .....	B-9
B.7	Configuring The Parallel Port .....	B-10
B.8	Configuring The Generic Print Engine Interface .....	B-11
B.9	MC68322 Memory Map Initialization Example .....	B-12
B.10	MC68322 Internal Registers Sample Code .....	B-13

## **Appendix C Memory-Mapped Register Summary**

C.1	MC68322 Mask Register .....	C-3
C.2	Test Register .....	C-3

## **Appendix D Alternate Pin Functions**

D.1	Pins .....	D-1
D.2	State During Reset .....	D-2
D.3	Registers .....	D-2
D.4	Input Pin Mode .....	D-3
D.5	Buzzer .....	D-3
D.6	In-Circuit Emulation .....	D-3
D.7	Operation Example .....	D-4

## **Appendix E Glossary**

## **Index**

# LIST OF ILLUSTRATIONS

Figure Number	Title	Page Number
1-1.	MC68322 Block Diagram .....	1-3
1-2.	Graphics Unit Data Flow Diagram .....	1-5
1-3.	16M Memory Map .....	1-7
1-4.	256M Memory Map .....	1-7
1-5.	Memory Map Address Register .....	1-7
1-6.	Bitmap Structure .....	1-9
1-7.	Unpacked And Packed Bitmaps .....	1-9
1-8.	Duplex Laser Printer Paper Path .....	1-11
1-9.	Example of a Duplex Printing Operation .....	1-12
2-1.	Functional Signal Groups .....	2-1
2-2.	CLK1 Phase Relationship .....	2-4
3-1.	EC000 Core Programming Model .....	3-2
4-1.	Read Cycle Flowchart .....	4-2
4-2.	External Timing Diagram to Chip-Selects Banks .....	4-2
4-3.	External Timing Diagram to Chip-Select Banks .....	4-3
4-4.	Word and Byte Read Cycle Timing Diagram to DRAM .....	4-3
4-5.	Write Cycle Flowchart .....	4-4
4-6.	Word and Byte Write Cycle Timing Diagram to Chip-Selects .....	4-5
4-7.	Word Write Cycle Timing Diagram to DRAM .....	4-6
4-8.	Internal Interrupt Acknowledge Cycle .....	4-7
4-9.	Interrupt Acknowledge Cycle Timing Diagram .....	4-7
4-10.	Reset Operation Timing Diagram .....	4-8
4-11.	Bus Arbitration Timing Diagram .....	4-10
4-12.	External Bus Master Read Cycle .....	4-11
4-13.	External Bus Master Write Cycle .....	4-12
5-1.	Software Interrupt Event Register .....	5-3
5-2.	External Interrupt Registers (EXIR0/2–EXIR1/3) .....	5-4
5-3.	Timer Register .....	5-6
5-4.	Timer Interrupt Event Register .....	5-6
5-5.	General Exception Processing Flowchart .....	5-7
5-6.	General Form of an Exception Stack Frame .....	5-8
5-7.	Module Soft-Reset Register .....	5-14

# LIST OF ILLUSTRATIONS (Continued)

Figure Number	Title	Page Number
6-1.	Chip-Select Register (CSR7–CSR0) .....	6-1
6-2.	Chip-Select DMA Timing and Recovery Registers .....	6-3
6-3.	Synchronous Read or Write Timing Diagram .....	6-5
6-4.	Asynchronous Read or Write Timing Diagram .....	6-6
6-5.	Special Asynchronous Read or Write Timing Diagram .....	6-6
7-1.	DRAM Register (DRAM5–DRAM0) .....	7-1
7-2.	DRAM Timing Mode 1 (Read Cycle, ROM Mode = 0) .....	7-3
7-3.	DRAM Timing Mode 1 (Read Cycle, ROM Mode = 1) .....	7-3
7-4.	Address Demultiplexing Example .....	7-4
7-5.	DRAM Control Register .....	7-5
7-6.	DRAM Refresh Cycle .....	7-6
7-7.	DRAM Timing Mode 0 (Read Cycle, ROM Mode = 0) .....	7-7
7-8.	DRAM Timing Mode 1 (Read Cycle, ROM Mode = 0) .....	7-7
7-9.	DRAM Timing Mode 2 (Read Cycle, ROM Mode = 0) .....	7-8
7-10.	DRAM Timing Mode 0 (Write Cycle) .....	7-8
7-11.	DRAM Timing Mode 1 (Write Cycle) .....	7-9
7-12.	DRAM Timing Mode 2 (Write Cycle) .....	7-9
8-1.	PDMA and GDMA Configuration Registers .....	8-2
8-2.	GDMA Control Register .....	8-3
8-3.	DMA Speed Register .....	8-4
8-4.	DMA Interrupt Event Registers .....	8-5
8-5.	GDMA MC68322 Bus Read Or Write Cycle .....	8-8
8-6.	Byte-Sized DMA DRAM Write Transfer .....	8-9
8-7.	Word-Sized DMA DRAM Write Transfer .....	8-9
9-1.	Parallel Port Interface Controller Block Diagram .....	9-1
9-2.	Parallel Port Interface Register .....	9-2
9-3.	Parallel Port Control Register .....	9-4
9-4.	PPI Interrupt Event Register .....	9-6
9-5.	Compatibility Mode Timing Diagram .....	9-8
9-6.	ECP Mode Timing Diagram .....	9-9
9-7.	Error Cycle Timing Diagram .....	9-13
9-8.	Parallel Port Data Latch Timing Diagram .....	9-13
10-1.	Printer Communication Register .....	10-2
10-2.	PVC Control Register .....	10-3
10-3.	Printer Control Block Register Set .....	10-5
10-4.	PVC Interrupt Event Register .....	10-6

## LIST OF ILLUSTRATIONS (Continued)

Figure Number	Title	Page Number
10-5.	Printer Communication Interrupt Event Register .....	10-8
10-6.	PVC Video Interface State Diagram .....	10-10
10-7.	Command Operation—MC68322 Supplies CCLK .....	10-12
10-8.	Command Operation—Print Engine Supplies CCLK .....	10-13
10-9.	Status Operation—MC68322 Supplied CCLK .....	10-14
10-10.	Status Operation—Print Engine Supplied CCLK .....	10-14
11-1.	RGP Start Register .....	11-2
11-2.	RGP Diagnostic Register .....	11-2
11-3.	RGP Interrupt Event Register .....	11-2
12-1.	Eight Common Graphic Operation Transfers .....	12-3
12-2.	256 Possible Boolean Coded Graphic Operation Transfers .....	12-4
12-3.	Bit String Specifier Formats .....	12-7
12-4.	Scanline Run Operation .....	12-8
12-5.	Illegal Bit String Specifier Use .....	12-9
12-6.	32-Bit Halftone Specifier Format .....	12-10
12-7.	48-Bit Halftone Specifier Format .....	12-11
12-8.	Scanline and Halftone Table Example .....	12-13
13-1.	Controlling Left and Right Clipping of Expanded Bit Maps .....	13-4
13-2.	Halftone Specification for bitBLT Operations .....	13-7
13-3.	Destination bitBLT to Banded Bitmap—0° Page .....	13-10
13-4.	Destination bitBLT to Banded Bitmap—180° Page .....	13-10
13-5.	Source/Destination bitBLT to Banded Bitmap—0° Page .....	13-12
13-6.	Source/Destination bitBLT to Banded Bitmap—180° Page .....	13-12
13-7.	Source/Halftone/Destination bitBLT to Banded Bitmap—0° Page .....	13-15
13-8.	Source/Halftone/Destination bitBLT to Banded Bitmap—180° Page .....	13-15
13-9.	Expanded Source, Destination bitBLT To Banded Bitmap, 0° Page .....	13-18
13-10.	Expanded Source, Destination bitBLT To Banded Bitmap, 180° Page .....	13-18
13-11.	Expanded Source, Halftone, Destination bitBLT To Banded Bitmap, 0° Page .....	13-22
13-12.	Expanded Source, Halftone, Destination bitBLT To Banded Bitmap, 180° Page .....	13-22
13-13.	Destination bitBLT to Frame .....	13-23
13-14.	Source/Destination bitBLT to Frame .....	13-24
13-15.	Source/Halftone/ Destination bitBLT to Frame .....	13-26
13-16.	Expanded Source, Destination bitBLT To Frame .....	13-28
13-17.	Expanded Source, Halftone, Destination bitBLT To Frame .....	13-31
13-18.	Destination bitBLT to Unbanded Bitmap .....	13-32

## LIST OF ILLUSTRATIONS (Continued)

Figure Number	Title	Page Number
13-19.	Source/Destination bitBLT to Unbanded Bitmap .....	13-33
13-20.	Source/Halftone/Destination bitBLT to Unbanded Bitmap .....	13-35
13-21.	Expanded Source, Destination bitBLT To Unbanded Bitmap .....	13-37
13-22.	Expanded Source, Halftone, Destination bitBLT to Unbanded Bitmap .....	13-40
13-23.	Banded Bitmap Parameters .....	13-44
13-24.	Halftone Bitmap Parameters .....	13-46
13-25.	Unpacked Source Bitmap .....	13-47
13-26.	Destination Scanline Transfer to Banded Bitmap, 0° Page .....	13-50
13-27.	Destination Scanline Transfer to Banded Bitmap, 180° Page .....	13-50
13-28.	Halftone, Destination Scanline Transfer to Banded Bitmap, 0° Page .....	13-53
13-29.	Halftone, Destination Scanline Transfer to Banded Bitmap, 180° Page .....	13-53
13-30.	Destination Scanline Transfer to Frame .....	13-54
13-31.	Halftone, Destination Scanline Transfer to Frame .....	13-56
13-32.	Destination Scanline Transfer to Unbanded Bitmap .....	13-57
13-33.	Halftone, Destination Scanline Transfer to Unbanded Bitmap .....	13-59
14-1.	Clock AC Timing .....	14-3
14-2.	Reset AC Timing .....	14-3
14-3.	Read Access (2:2:1:3) .....	14-5
14-4.	Read Access (2:4:-1:3) .....	14-5
14-5.	Read Access (4:2:-1:3) .....	14-6
14-6.	Read Access (2:2:3:3) .....	14-6
14-7.	Read Access (2:4:1:3) .....	14-7
14-8.	Read Access (4:2:1:3) .....	14-7
14-9.	Read Access (2:6:-1:3) .....	14-8
14-10.	Read Access (4:4:-1:3) .....	14-8
14-11.	Read Access (6:2:-1:3) .....	14-9
14-12.	Write Access (4:2:-1:3) .....	14-9
14-13.	Write Access (4:2:1:3) .....	14-10
14-14.	Write Access (4:4:-1:3) .....	14-10
14-15.	DMA Read Cycle AC Timing .....	14-11
14-16.	DMA Write Cycle AC Timing .....	14-11
14-17.	DRAM Read Cycle AC Timing .....	14-12
14-18.	DRAM Write Cycle AC Timing .....	14-12
14-19.	DMA Request/Acknowledge AC Timing .....	14-13
14-20.	Print Engine Interface Input AC Timing .....	14-15
14-21.	Print Engine Interface Output AC Timing .....	14-15
14-22.	Video Clock AC Timing .....	14-15
14-23.	PVC AC Timing .....	14-16
14-24.	Print Engine Interface AC Timing .....	14-16
14-25.	Interrupt Interface AC Timing .....	14-16
14-26.	Parallel Port Interface AC Timing .....	14-17

# LIST OF ILLUSTRATIONS (Continued)

Figure Number	Title	Page Number
14-27.	External Bus Master Read Cycle AC Timing .....	14-18
14-28.	External Bus Master Write Cycle AC Timing .....	14-19
14-29.	External Bus Master Bus Arbitration AC Timing .....	14-19
14-30.	External Bus Master Multiple Cycle AC Timing .....	14-19
15-1.	MC68322 160-Lead Plastic Quad Flat Pack (PQFP) .....	15-2
15-2.	160 Pin QFP Package Dimensions .....	15-3
A-1.	ICE Interface Block Diagram .....	A-5
A-2.	MC68322 PGA Pinout .....	A-6
A-3.	MC68000 Emulator Connection .....	A-7
A-4.	PGA Connector .....	A-8
A-5.	Test Points .....	A-9
A-6.	ICE Adaptor Board .....	A-10
A-7.	ICE Adaptor Board—Silkscreen .....	A-11
A-8.	ICE Reset AC Timing .....	A-12
A-9.	ICE Read Cycle AC Timing .....	A-13
A-10.	ICE Write Cycle AC Timing .....	A-14
A-11.	ICE Interrupt Acknowledge Cycle AC Timing .....	A-15
A-12.	ICE Bus Arbitration AC Timing .....	A-15
B-1.	MC68322 Connection .....	B-2
B-2.	DRAM Connection .....	B-3
B-3.	DRAM SIMM Connection .....	B-4
B-4.	Flash EEPROM Connection (1 of 2) .....	B-5
B-5.	Reset Circuit .....	B-7
B-6.	Front Panel Buffers and Latches .....	B-8
B-7.	4-Kbit Serial EEPROM Connection .....	B-8
B-8.	MC68322 ICE Interface .....	B-9
B-9.	Parallel Port Connector Interface .....	B-10
B-10.	Print Engine Interface .....	B-11
B-11.	Initialized Memory Map From Code Example .....	B-12



# LIST OF TABLES

Table Number	Title	Page Number
2-1.	Signal Summary .....	2-2
2-2.	HI-Z and TEST Combinations .....	2-5
2-3.	DRAM Address Multiplexer .....	2-7
3-1.	Processor Data Formats .....	3-3
3-2.	Effective Addressing Modes .....	3-3
3-3.	Notational Conventions .....	3-4
3-4.	Instruction Set Summary .....	3-6
5-1.	Hardware Interrupt Events .....	5-2
5-2.	External Interrupt Polarity .....	5-5
5-3.	Exception Vector Assignments .....	5-9
5-4.	Exception Grouping and Priority .....	5-13
6-1.	Size Field Encoding .....	6-3
6-2.	Synchronous Timing Values .....	6-4
7-1.	DRAM Size Options .....	7-2
7-2.	DRAM Timing Modes .....	7-5
8-1.	DM Field Encoding .....	8-4
10-1.	SLC and SRC Encodings .....	10-3
10-2.	PLL Video Clock Divisor .....	10-15
12-1.	Graphic Operation Data Operand Constant Values .....	12-2
12-2.	Bit String Specifier Field Definitions .....	12-8
12-3.	bitBLT and Scanline Execution Times .....	12-14
13-1.	Graphic Order Organization .....	13-2
13-2.	Graphic Orders Sorted by Opcode .....	13-8
13-3.	Supported Scaling Factors .....	13-17

## LIST OF TABLES (Continued)

Table Number	Title	Page Number
A-1.	ICE Interface Signal Summary .....	A-1
A-2.	Data Strobe Control of Data Bus .....	A-2
A-3.	Function Code Outputs .....	A-3
C-1.	Memory-Mapped Register Set .....	C-1
D-1.	ALTPIN SEL Bit Descriptions .....	D-1

## SECTION 1 INTRODUCTION

The MC68322 is a high-performance integrated printer processor that combines an MC68000 compatible EC000 core processor, a RISC graphics processor (RGP), a print engine video controller (PVC), and numerous system integration features on a single integrated circuit. It is the first of Motorola's M68000 Family designed specifically for nonimpact printers. The MC68322 provides a unique solution for new designs as well as an excellent migration path for existing M68000-powered printers. Additionally, the new chip finds ready application to the inkjet printer and multifunction-peripheral (fax/modem/printer) markets and other embedded control applications, which require very fast bit manipulations. The dual processor and dual bus architecture gives the MC68322 the ability to deliver excellent performance. Historically, printer applications have been solved using a single general-purpose processor with external application specific circuitry. The MC68322 employs a highly specialized, multiprocessor architecture that enables the user to take advantage of memory reduction techniques. This design implementation provides a technically superior and more cost effective system solution.

The specialized display list banding techniques (executed by the RGP) enable system memory requirements to be significantly reduced. The use of software memory reduction techniques alone (an approach taken by conventional controllers) lack the power needed to handle complex pages, causing the controller to fall back to lower resolution or reduced page throughput. The MC68322 optimizes overall system performance by integrating an EC000 core, RGP, and PVC using a unique dual bus architecture. This architecture eliminates bus contention between processing units and modules creating a true parallel processing environment. The additional bandwidth allows each processing unit to operate at peak performance. Working in conjunction with an on-chip, programmable, bursting DRAM controller, the processing units are capable of achieving outstanding throughput. These dedicated processing units enable the MC68322 to produce 600 dpi images using substantially less memory than conventional controllers. The MC68322 extends these benefits to low-cost 4-8 ppm printers.

The MC68322 significantly reduces component count, board space, power consumption and their inherent costs while yielding higher reliability and shorter design time. It also provides support for toner conservation, thus enabling the print controller to conserve toner when printing in draft mode. The MC68322 also provides the perfect printing environment for users of complex page description languages (such as PCL and PostScript®) and less scaleable graphics imaging models such as Windows Printing System™ and QuickDraw®). Complete code compatibility with the M68000 Family gives the designer access to a broad base of established real-time kernels, operating systems, languages, applications, and development tools, many of which are optimized for embedded processing and printing applications.

## 1.1 FEATURES

The following list summarizes the main features of the MC68322:

- Static EC000 Core Processor
  - ❑ Complete code compatibility with M68000 Family
  - ❑ Glueless interface to peripherals
  - ❑ 256M address range
- Graphics Unit
  - ❑ Memory reduction techniques
    - Run length encoded scan line tables
  - ❑ RISC graphics processor
    - Processes multi-operation graphics orders from display list
    - Requires significantly less bitmap image memory due to hardware banding capability
    - Dedicated graphics bus allows up to 8 ppm performance at 600 dpi resolution
  - ❑ Print engine video controller
    - Converts bitmap image to serial datastream and feeds print engine
    - Generic, programmable, nonimpact printer communications interface
    - Toner conservation technique
  - ❑ Dedicated high-performance DMA controller for graphics unit operations
- Bus Interface Unit
  - ❑ Dual bus architecture allows separate buses to function independently
  - ❑ Distributed processing optimizes system performance
  - ❑ Write buffer for EC000 core enhances performance
- System Integration Module
- Eight Programmable Chip Selects
  - ❑ 256K to 256M of address space
  - ❑ Independently programmable timing parameters for each bank
  - ❑ Integrated system timer
- DRAM System Integration Module
  - ❑ Supports 512K, 2M, and 8M DRAM bank sizes
  - ❑ Directly controls up to 6 banks of DRAM; supports up to 48M of DRAM
  - ❑ Programmable transparent refresh of DRAM banks
  - ❑ Bursting DRAM interface
- General-Purpose DMA Controller Module
  - ❑ Provides high-speed downloads to and from DRAM
- IEEE 1284 Parallel Port Controller Module
  - ❑ DMA controller supports 2M/sec bidirectional communication transfers
- 16-, 20-, or 25-MHz Operation
- 160-Pin Plastic Quad Flat Packaging

## 1.2 PROCESSORS AND MODULES

To improve total system throughput and reduce part count, board size, and cost of system implementation, the M68300 Family integrates intelligent peripheral modules with typical glue logic on-chip. The MC68322 consists of two processor units (the EC000 core and RGP) and six modules that assist them (the PVC, bus interface unit, system integration module, DRAM system integration module, DMA controller, and parallel port interface). Figure 1-1 illustrates the MC68322 block diagram.

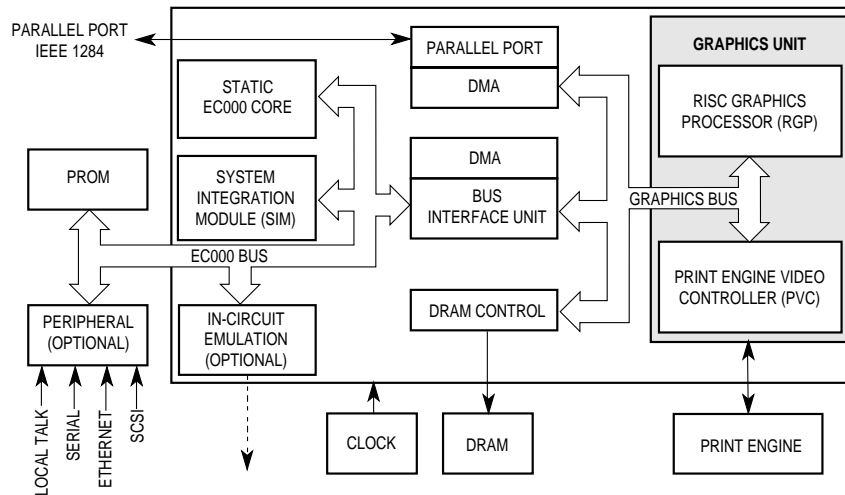


Figure 1-1. MC68322 Block Diagram

### 1.2.1 The EC000 Core

The MC68322 contains a static, low-power, 16-bit microprocessor (EC000 core), which performs general-purpose computing, I/O and exception handling, and display list rendering. The core has a 16-bit data path that is upward compatible with 32-bit machines. It also has a 32-bit internal architecture with internal 32-bit data and address registers, and an extended address range. The address range is 28 bits for internal register decoding of chip-selects and DRAM controller functions. This address range allows for full code compatibility with existing M68000 Family-based designs and future upward compatibility to higher performance designs. The core is register and memory-map compatible with the industry standard MC68000, MC68EC000, and MC68HC000 processors.

The MC68322 is designed to support in-circuit emulation with existing emulators so that new hardware and software designs being ported to the MC68322 can be tested rapidly. This is accomplished by providing signals in a 208-pin grid array (PGA) package that are not available in the 160-pin quad flat pack (QFP) package.

## 1.2.2 Graphics Unit

The graphics unit performs all graphics functions required by complex PDLs, such as bit-block transfer (bitBLT). A bitBLT is a CPU-intensive function of logically bits from one memory location to another. The graphics unit acts as a dedicated hardware execution engine to perform the bitBLT function with virtually no intervention from the core. bitBLT operations are performed very fast by the graphics unit, supporting one, two, and three operand bitBLT operations to yield 256 logical bitBLT operation combinations.

The graphics unit contains two independent processing units that can function in parallel with the core—a RISC graphics processor (RGP) and a print engine video controller (PVC). Both processing units perform burst read and write accesses to DRAM through the DRAM controller. The RGP is a high-performance bit-image processor optimized for the 16-bit DRAM controller on the MC68322. It achieves performance levels that enable the MC68322 to be used effectively in banding applications or in other high-speed, high-density bitmapped graphics products. The RGP is comprised of four major blocks:

- Graphic order parser
- Graphic order execution unit
- Writeback logic
- Band control registers

The PVC contains a generic nonimpact printer communication interface, which can be used with most of the printers currently on the market. The communications interface is 8-bit synchronous full duplex and supports almost all laser and inkjet printers. Internal interrupt events (if enabled) indicate that a serial command has been sent or a serial status has been received. This interface accesses a memory-mapped register called the printer communication interface register, which contains 8-bit command and status fields. Using these fields, the printer communication interface controls the  $\overline{\text{CBSY}}$  and  $\overline{\text{SBSY}}$  signals to provide a handshake that communicates between the PVC and the print engine. In addition to this communication interface, the PVC also provides for serialization of the bitmap image data through the video data output at a clock rate specified by the video clock input. A digital phase-locked loop is also provided for those printers that do not supply a video clock source.

The RGP interprets a list of special instructions called graphic orders (a display list that the core or host application processor generates) to render a banded bitmap page image. After a page or band image is rendered by the RGP, the PVC converts the bitmap image into a serial datastream and transfers the rendered page image through the video port to the print engine. Both the RGP and PVC require only a minimal amount of initialization and intervention by the core to produce an image and transfer it to the print engine. Figure 1-2 illustrates the data flow of the graphics unit.

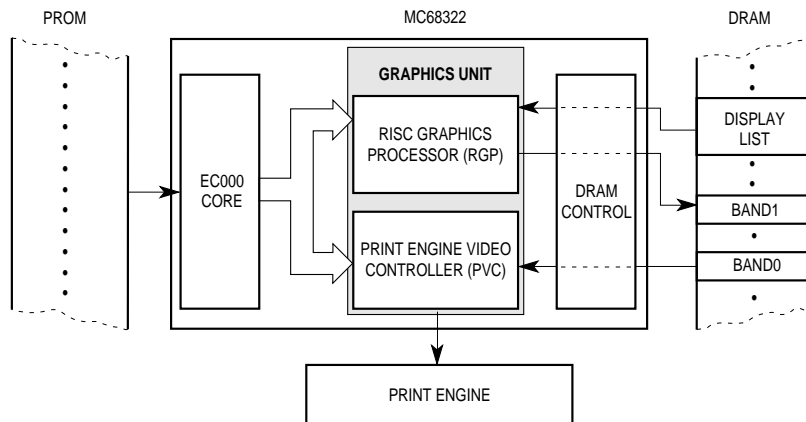


Figure 1-2. Graphics Unit Data Flow Diagram

### 1.2.3 Bus Interface Unit

The dual bus architecture of the MC68322 allows the printing workload to be distributed among processing units and executed in parallel. The bus interface unit (BIU) allows the core and graphics unit, which reside on the MC68322 bus and graphics bus, to function independently. This is done through an arbitration unit which accommodates core accesses to DRAM residing on the graphics bus. However, to print pages correctly, the graphics unit gets higher priority than the core for DRAM accesses. The core performs instruction and PROM data fetches without any impact to graphics bus operations. The BIU contains a single-word, writeback buffer that reduces peak bus traffic generated by multiple active modules. The writeback buffer provides a no-wait state write profile to the core and delays the write until the graphics unit stops using the graphics bus.

### 1.2.4 System Integration Module

The system integration module (SIM) provides the ROM, PROM, and peripheral chip-selects. It contains eight chip-select banks that can be programmed to decode addresses and supply internal  $\overline{DTACK}$  termination. These eight chip-select banks are individually programmable for an address range of 256K to 64M. They can be located anywhere within the 256M memory map and can be either contiguous or disjointed, as required by the operating environment. Also, each chip-select bank can be independently size or disabled.

The chip-selects for each bank can be set up to provide a wide range of timing parameters, such as setup, access, hold, and recovery times for both read and write bus cycles. The MC68322's SIM provides internal bus cycle auto-acknowledge and the asynchronous  $\overline{WAIT}$  signal allows external devices to insert additional wait states as needed. The SIM also allows SRAM to be added to the MC68322 bus for system stack space, temporary data storage, or as a buffer for peripheral data.

### 1.2.5 DRAM Controller

The MC68322 provides a fully integrated bursting DRAM controller containing six DRAM banks of varying programmable sizes and locations. They can be located contiguously or disjointedly, as required by the operating environment. The DRAM controller multiplexes addresses to provide up to 8M of DRAM address space per bank. The timing parameters for each DRAM bank are preprogrammed to provide a 3-, 4-, or 5-clock access from industry standard fast-page mode DRAMs. On reset, all DRAM banks are disabled. Additionally, the DRAM controller provides a separate 16-bit DRAM data path and a write enable signal for a glueless DRAM interface. DRAM refresh cycles are carried out with  $\overline{CAS}$  before  $\overline{RAS}$  refresh cycles. The DRAM refresh rate is fully programmable and the controller performs refreshes from system reset until it is initialized.

### 1.2.6 DMA Interface

The DMA interface contains two DMA controllers—a single-ended general-purpose DMA (GDMA) and a dedicated parallel port interface DMA (PDMA) controller. The DMA interface can be programmed to transfer data from a high-speed I/O peripheral to DRAM with minimal intervention from the core.

### 1.2.7 Parallel Port Interface

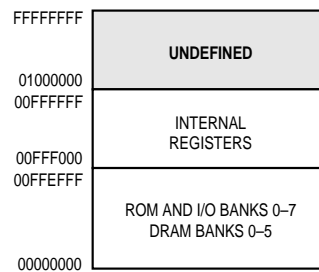
The MC68322 contains a direct, IEEE 1284 Level 2 compliant, bidirectional 8-bit PPI. The PPI supports four IEEE 1284 communications modes—compatibility (Centronics™), nibble, byte, and enhanced capabilities port (ECP). It also fully supports all variants of these modes, including device ID requests and run-length encoded data compression. The PPI contains specialized hardware to provide automatic handshaking during forward data transfers. When hardware handshaking is used in conjunction with the PDMA, transfer rates as high as 2M/sec and up can be achieved in the ECP forward mode. The hardware handshaking can also be completely disabled for the software to directly control the parallel port interface signals and support new protocols. Control and data signals provide a glueless interface to the parallel port.

## 1.3 INTERNAL MEMORY MAP

The MC68322 uses memory-mapped registers that occupy 4K of memory space. With these registers the hardware configuration and timing can be set, the status information can be read, and the PVC, RGP, DMA, and PPI interfaces can be controlled. All registers can be written and read, except for a few read-only and write-only registers that are noted. For more information about each register, see its corresponding module's section. **Appendix C Memory-Mapped Register Summary** discusses all the registers and their location in memory during power-up.

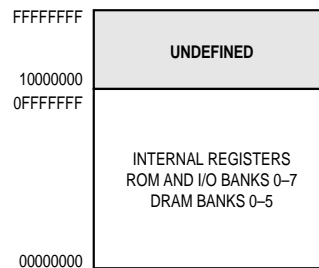
Register operations are implemented within one MC68322 bus cycle for both read and write operations and are completed without asserting any wait states. The registers should only be read and/or written as 16-bit words. All register addresses are on word boundaries. The MC68322 powers up with a 16M memory map with the registers occupying the upper 4K of the 16M of memory space. They are located at address range 0x00FFF000 through 0x00FFFFFF. The MC68322 memory map for a 16M memory space is illustrated in Figure 1-3.





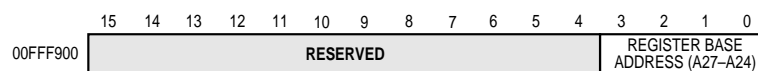
**Figure 1-3. 16M Memory Map**

After power-up, the MC68322 can be configured for the full 256M address space and the registers can be moved to start at a higher address. The MC68322 memory map for a 256M memory space is illustrated in Figure 1-4.



**Figure 1-4. 256M Memory Map**

The memory map address register contains bits 27–24 of the register set's base address and is illustrated in Figure 1-5. This register is used to relocate the memory-mapped registers within the memory map.



**Figure 1-5. Memory Map Address Register**

DRAM, ROM, or I/O can be programmed individually to reside anywhere in the memory map. The address space for registers, ROM or I/O (chip-selects), and DRAM can overlap. In case of an address overlap, registers have the highest priority, then chip-selects, and finally the DRAM. Only the device with the highest priority responds to the access.

## 1.4 UNDERSTANDING THE MC68322

Familiarity with some of the basic printer operation concepts is key to understanding how the MC68322 works. These concepts include understanding printer languages, bitmaps, banding, halftoning, and duplex printing.

### 1.4.1 Printer Languages

There are three basic types of printer languages:

1. Printer Control Language (PCL)—A term coined by Hewlett Packard when LaserJet printers were first introduced. It embodies a relatively simple set of escape sequences reminiscent of ANSI 3.64. Of the common printer languages, PCL is considered moderately complex.
2. Page Description Languages (PDL)—Actual programming languages. The major players are Adobe's PostScript and Microsoft's TrueImage and they resemble other languages like BASIC, FORTRAN, and C, except that PDLs are interpreted rather than compiled. The instructions for how the page is to be formed are described in lexical verbs such as FINDFONT and MOVETO. This means that the parsing and interpretation of these languages must be done in the printer engine itself. Generally, PDLs describe one page at a time and each page is a separate PDL program. Also, PDLs are considered highly complex.
3. Document Description Languages (DDL)—Similar to PDLs in that they are programming languages with lexical verbs. The difference between DDLs and PDLs is that DDLs generally describe an entire document consisting of multiple pages. This increases the storage requirements of a printer in that the entire document must be parsed and interpreted before any printing can begin. Like PDLs, DDLs are considered highly complex.

## 1.4.2 Bitmap

A bitmap is a two dimensional array of memory bits. A scanline is one row in the array. There is no special term for each column of the array. The junction point of a scanline and a column is a pixel. The bitmap width, called the X dimension, is the number of pixels in each scanline. The bitmap height, called the Y dimension, is the number of scanlines in the bitmap array. Figure 1-6 illustrates these terms.

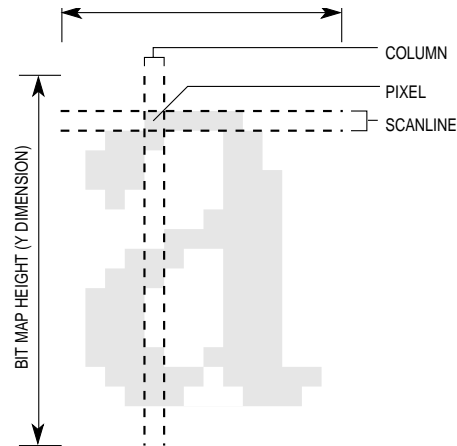


Figure 1-6. Bitmap Structure

A bitmap can be stored in memory as either packed or unpacked, as illustrated in Figure 1-7. Packed bitmaps occupy less memory than unpacked bitmaps. In an unpacked bitmap, each scanline begins on a byte or word boundary. In a packed bitmap, scanlines follow one another without regard to byte or word boundaries. The MC68322 supports both packed and unpacked bitmap structures.

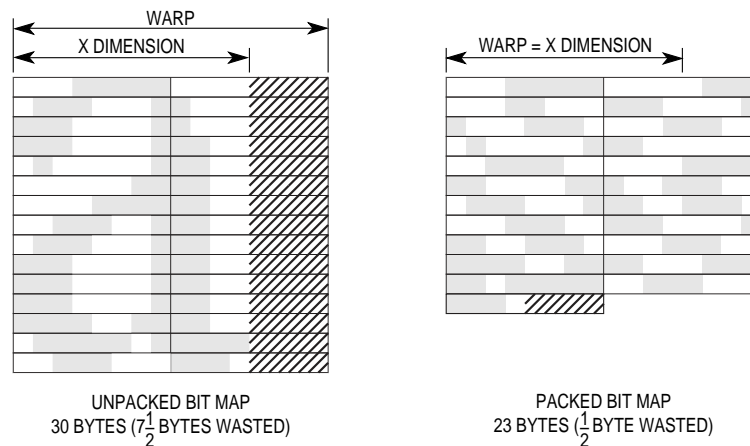


Figure 1-7. Unpacked And Packed Bitmaps

## Introduction

---

In addition to width and height dimensions, an unpacked bitmap also possesses an attribute known as the bitmap warp, which is the distance between the beginning of each consecutive scanline. The bitmap warp is the value used to obtain Y dimension movement within the bitmap. For example, to move from one pixel to a pixel in the same column, but in the next lower scanline, simply add the bitmap warp to the current position in memory. A packed bitmap also has a bitmap warp that is equal to the width of the bitmap.

### 1.4.3 Banding

Banding is a process in which the page to be printed is constructed in a series of partial page images or bands. To better accommodate banding, the MC68322 allows the page image to be represented in an intermediate form. In this intermediate form, the page image is represented by a series of graphic orders, which are collectively called a display list and are maintained in memory.

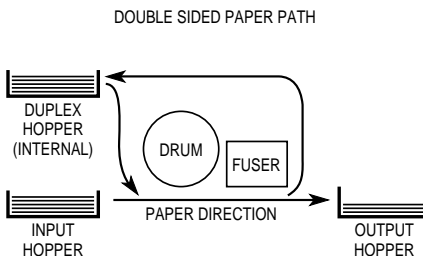
The PDL or PCL emulator firmware running on the core generates the display list before the print engine begins the actual printing process. The RGP executes the graphic orders in the display list to build the bitmapped image in bands as the print engine is started. Generally, band  $n+1$  is being constructed while band  $n$  is being output to the print engine.

### 1.4.4 Halftoning

Halftoning involves applying a pattern or halftone screen to a data transfer to modify its appearance. Halftone screens are used to produce shades of gray in a monochrome printing environment such as printing presses, dot matrix printers, or laser printers. Halftone screens are commonly seen in newspapers because that is where photographs with levels of gray are represented with a medium that only allows black and white. Halftone screens are repetitive in both the X and Y dimensions of a bitmap array. For example, to perform shading, a 10101010 pattern might be applied to the even scanlines and a 01010101 pattern to the odd scanlines.

### 1.4.5 Duplex Printing

The MC68322 supports duplex printing applications. Duplex printing is the operation of placing an image on both sides of a page before it leaves the printer. In a duplex laser printer, paper travels out of the input hopper and under a drum to receive a toner image. The paper then travels through a fuser to set the toner onto the first side of the paper and into an internal duplex hopper. Next, the paper moves out of the duplex hopper and under the drum again to receive the second toner image, this time on the reverse side of the paper. Finally, once the second image has been fused, the paper is placed in the output hopper.



**Figure 1-8. Duplex Laser Printer Paper Path**

Since the image is placed on both sides of the page in duplex printing, image orientation is important. When the page is turned to read either side, both images must appear right-side up. To achieve the correct image orientation, the physical characteristics of the print engine and the format of the printed page must be taken into consideration. For example, it is important to know how the page is turned over to expose both sides to the drum inside the print engine as well as how the page is going to be bound in the completed document.

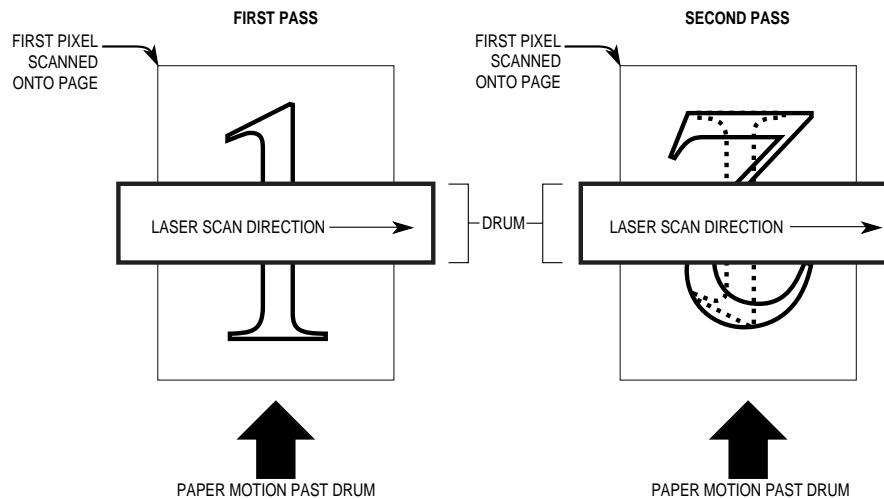
Images printed on the opposite side of a page may have to be rotated 180°. Understandably, these types of pages are called 180° pages. They require pixel data to be transmitted from the page image bitmap starting at the bottom-right corner and then continuing from right to left, bottom to top. Pages that do not require this type of rotation are called 0° pages and they are transmitted starting at the top-left corner and then continuing from left to right, top to bottom.

In banding applications, 180° pages require special attention. Since the page image is transmitted to the print engine in bottom to top order, bands must be generated in this order too. The MC68322 directly supports 180° page rendering and printing as required by duplex banding applications. The RGP and graphic order instruction set are specifically designed to render images either top to bottom or bottom to top, thus enabling banding of both 0° and 180° pages on duplex printers.

## Introduction

There are two elements to keep in mind when determining image orientation—the feed edge and the binding edge. The feed edge of a page is the edge that is first fed into the print engine. The binding edge of a page is the edge that will be used in the binding process. For example, if a document is to be stored in a three-ring binder, the binding edge would be the edge of the paper where the holes are punched. To determine whether the second side of a page needs to be rotated, the feed edge must be compared to the binding edge. If the feed edge is the same as the binding edge, then the second side does not need to be rotated. However, if the feed edge is different from the binding edge, then the second side must be rotated 180° to have the proper orientation between both sides of the page.

Figure 1-9 illustrates a duplex printing operation. The paper is fed by its short edge when the binding edge is defined as the long edge. This means the second pass image should be rotated 180°. During the first pass, paper travels from the input hopper and under the drum and an image is placed on the page. After fusing, the page is placed face down in the duplex hopper. During the second pass, paper travels from the duplex hopper and under the drum and another image is placed on the page. After fusing, the page is placed in the output hopper with the second pass image facing up. To properly orient the two images on the page the second pass data must be sent to the printer in right to left and bottom to top order so that a 180° image rotation will occur.



**Figure 1-9. Example of a Duplex Printing Operation**

# SECTION 2

## SIGNAL DESCRIPTIONS

This section contains brief descriptions of the MC68322 input and output signals as illustrated in the figure below.

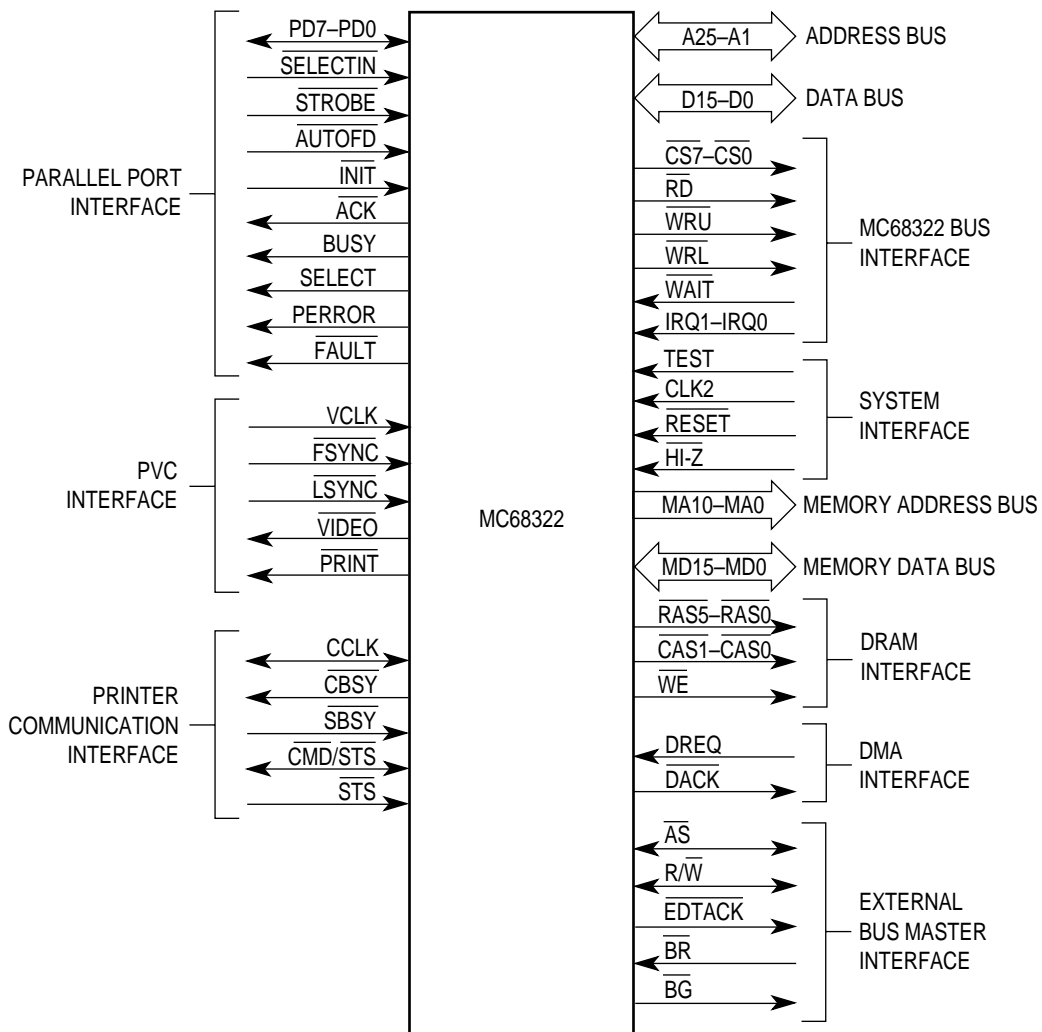


Figure 2-1. Functional Signal Groups

Table 2-1. Signal Summary

SIGNAL NAME	MNEMONIC	INPUT/OUTPUT	ACTIVE STATE	THREE-STATE	
				ON $\overline{BG}$	ON $\overline{HI-Z}$
Address Bus	A25–A1	Input/Output	—	Yes	Yes
Parallel Port Acknowledge	$\overline{ACK}$	Output	Low	No	Yes
Address Strobe	$\overline{AS}$	Input/Output	Low	Yes	Yes
Parallel Port Autofeed	$\overline{AUTOFD}$	Input	Low	—	—
Bus Grant	$\overline{BG}$	Output	Low	No	Yes
Bus Request	$\overline{BR}$	Input	Low	—	—
Parallel Port Busy	BUSY	Output	High	No	Yes
Column Address Strobe	$\overline{CAS1-CAS0}$	Output	Low	No	Yes
Command Busy	$\overline{CBSY}$	Output	Low	No	Yes
Command Clock	CCLK	Input/Output	—	No	Yes
2X System Clock	CLK2	Input	—	—	—
Command/Status Data	$\overline{CMD/STS}$	Input/Output	Low	No	Yes
Chip Select	$\overline{CS7-CS0}$	Output	Low	No	Yes
Data Bus	D0–D15	Input/Output	—	No	Yes
DMA Data Acknowledge	$\overline{DACK}$	Output	Low	—	Yes
DMA Data Request	DREQ	Input	Programmable	—	—
External Master DTACK	EDTACK	Output	Low	No	Yes
Parallel Port Fault	$\overline{FAULT}$	Output	Low	No	Yes
Frame Synchronization	$\overline{FSYNC}$	Input	Programmable	—	—
Parallel Port Initialization Input	$\overline{INIT}$	Input	Low	—	—
Interrupt Request	$\overline{IRQ1-IRQ0}$	Input	Programmable	—	—
Line Synchronization	$\overline{LSYNC}$	Input	Programmable	—	—
DRAM Multiplexed Address Bus	MA10–MA0	Output	—	No	Yes
DRAM Memory Data Bus	MD15–MD0	Input/Output	—	No	Yes
Parallel Port Data Bus	PD7–PD0	Input/Output	—	No	Yes
Parallel Port Paper Error	PERROR	Output	Low	No	Yes
Print Request	$\overline{PRINT}$	Output	Programmable	No	Yes
Read/Write	R/ $\overline{W}$	Input/Output	Read-High Write-Low	Yes*	Yes
DRAM Row Address Strobe	$\overline{RAS5-RAS0}$	Output	Low	No	Yes
Read Strobe	$\overline{RD}$	Output	Low	No	Yes
Reset	$\overline{RESET}$	Input	Low	—	—
Status Busy	$\overline{SBSY}$	Input	Low	—	—
Parallel Port Selected	SELECT	Output	High	No	Yes
Parallel Port Select In	$\overline{SELECTIN}$	Input	Low	—	—
Parallel Port Data Strobe	$\overline{STROBE}$	Input	Low	—	—



Table 2-1. Signal Summary (Continued)

SIGNAL NAME	MNEMONIC	INPUT/OUTPUT	ACTIVE STATE	THREE-STATED	
				ON $\overline{BG}$	ON HI-Z
Status Data	STS	Input	Low	—	—
High Impedance	$\overline{HI-Z}$	Input	Low	—	—
Power Input	$V_{CC}$	Input	—	—	—
Video Clock	VCLK	Input	Programmable	—	—
Video	$\overline{VIDEO}$	Output	Programmable	No	Yes
Processor Wait	$\overline{WAIT}$	Input	Low	—	—
DRAM Write Enable	$\overline{WE}$	Output	Low	No	Yes
Write Enable - Lower	$\overline{WRL}$	Output	Low	No	Yes
Write Enable - Upper	$\overline{WRU}$	Output	Low	No	Yes

NOTE: \* Becomes An Input On  $\overline{BG}$ , Which Effectively Three-States The Signal.



**Note:** Assertion and negation are used to specify forcing a signal to a particular state. *Assertion* and *assert* refer to a signal that is active or true. *Negation* and *negate* refer to a signal that is inactive or false. These terms are used independent of the voltage level (high or low) that they represent.

## 2.1 ADDRESS BUS

This 25-bit, bidirectional, three-state bus is capable of directly addressing 64M of data. The address bus acts as an output when the core or general-purpose DMA is accessing the chip-select banks and it acts as an input when the device is in an external bus master mode. In the external bus master mode, the address bus is sent to the MC68322's internal decode circuits.

## 2.2 DATA BUS

This 16-bit, bidirectional, three-state bus is the general-purpose data path. It acts as an input when one of the following conditions occur:

- When data from ROM, PROM, or I/O is required in the form of data or instructions.
- When an external bus master performs a write cycle.

On the other hand, it acts as an output when one of the following conditions occur:

- When data is sent to an I/O device.
- When an external bus master is reading data.

During an interrupt acknowledge cycle, the data bus is not used.

## 2.3 SYSTEM INTERFACE

### 2.3.1 Reset ( $\overline{\text{RESET}}$ )

The  $\overline{\text{RESET}}$  signal is an input only signal that causes a total system reset, thus resetting the processor and external devices. This is different than a reset caused by the **reset** instruction, which does not reset external devices or internal registers. In effect, the internal state of the processor is not affected. Regardless, using the **reset** instruction on the MC68322 is not recommended.

### 2.3.2 System Clock

To develop the internal clocks needed by the MC68322, CLK2 is internally buffered. The MC68322 divides the system clock (CLK2) frequency by two to generate a CLK1 signal that is used internally by the core and most modules. The input frequency of the CLK2 signal is twice the system frequency.

CLK2 can be slowed or stopped to reduce device and system power consumption. However, CLK2 is necessary to refresh DRAM, and complete removal of the CLK2 signal can cause a loss of data in DRAM.

The internal CLK1 signal functions continuously through reset. The phase relationship between CLK1 and CLK2 is determined by the trailing edge of  $\overline{\text{RESET}}$ . Figure 2-2 illustrates the internal timing of the MC68322 in which  $\overline{\text{RESET}}$  is doubly synchronized and the trailing edge is used to synchronize CLK1.

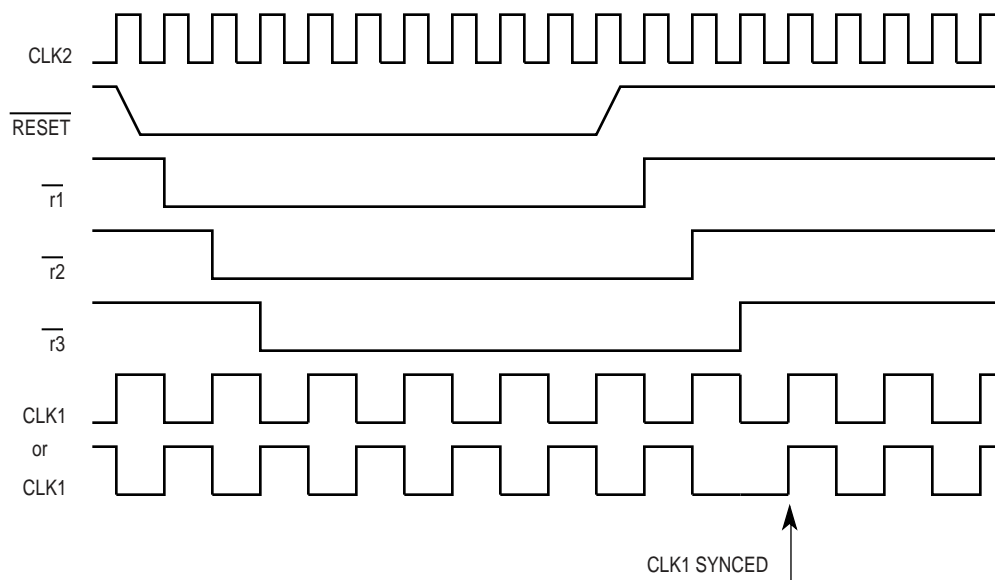


Figure 2-2. CLK1 Phase Relationship

CLK1 can be generated with external logic, as illustrated in the following equations. This logic runs off CLK2, takes system reset in as RESETIN, and generates CLK1 and RESET as outputs.

```
!rst1x      :=      !RESETINx;      "Sync reset input (internal use)
!RESETx     :=      !rst1x;         "Reset to MC68322
!rst2x      :=      !RESETx;        "Delayed reset (internal use)
!CLK1       :=      CLK1            "Toggle
#           :=      RESETx & !rst2x "Sync at trailing edge of RESET
```

Where:

```
!      =      Inverted
&      =      Logical AND
#      =      Logical OR
:=     =      Registered Output
=      =      Combination; Non-Registered Output
;      =      Line Termination
"      =      Comment
```

This external logic works in a similar manner as the logic used inside the MC68322. The external CLK1 runs continuously, even during reset, and only its phase is adjusted based on the trailing edge of the RESETIN input. The asynchronous RESETIN input is synchronized to avoid metastability problems and a synchronous RESET output signal is produced to reset the MC68322 and ensure that it generates its internal CLK1 in phase with its external CLK1.

### 2.3.3 High Impedance Mode

The high impedance mode  $\overline{\text{HI-Z}}$  and TEST pins can be used to place the MC68322 into a three-state mode that allows an in-circuit emulator to be used. Table 2-2 shows all combinations of the  $\overline{\text{HI-Z}}$  and TEST pins. See **Appendix D Alternate Pin Functions** for more information on the proper use of the input pin mode.

**Table 2-2.  $\overline{\text{HI-Z}}$  and TEST Combinations**

HI-Z	TEST	FUNCTION
0	0	Three-State All Outputs
0	1	Input Pin Mode, A(22:25) Three-States
1	0	Normal Mode, A(22:25) Enabled
1	1	Special Test Mode (Do Not Use)

## 2.4 EXTERNAL BUS MASTER INTERFACE

The following signals control the MC68322 bus operation.

PIN NAME	DESCRIPTION
$\overline{AS}$	<b>Address Strobe</b> —The active low $\overline{AS}$ signal indicates a valid address on the address bus. $\overline{AS}$ is an output when the core or internal DMA initiates an access on the MC68322 bus and an input when an external bus master controls the MC68322 bus.
$R\overline{W}$	<b>Read/Write</b> —This signal defines a data bus transfer as a read (active high) or write cycle (active low). $R\overline{W}$ is an output when the core or internal DMA initiates an access on the MC68322 bus, and an input when an external bus master has control of the MC68322 bus.
$\overline{EDTACK}$	<b>External Bus Master Data Transfer Acknowledge</b> —This output signal is sent to an external bus master to indicate that the data transfer is complete. When $\overline{EDTACK}$ is recognized during a read cycle, the external bus master latches the data and terminates the bus cycle. When $\overline{EDTACK}$ is recognized during a write cycle, the bus cycle is terminated.
$\overline{BR}$	<b>Bus Request</b> —This active low input is ORed with all other devices that can be bus masters. This active-low input signal informs the core that another device is ready to be the bus master.
$\overline{BG}$	<b>Bus Grant</b> —This active low output indicates to all other potential bus masters that the MC68322 bus is available. $\overline{BG}$ will assert after the assertion of $\overline{BR}$ , but only after all bus cycles have terminated.
$\overline{CS7-CS0}$	<b>Chip-Select</b> —These signals are output only and can be programmed to provide from 256K to 64M decode. These signals continue to function as they are programmed when an alternate bus master has control of the MC68322 bus.
$\overline{RD}$	<b>Read</b> —This signal is an output only strobe that is asserted during a read operation on the MC68322 bus. A read cycle can be initiated by the core, internal DMA, or external bus master. The read strobe remains negated during an MC68322 bus write cycle.
$\overline{WRU}$	<b>Upper Write Strobe</b> —This strobe is an output only signal that is asserted during a write operation on the MC68322 bus. A write cycle can be initiated by the core, internal DMA, or external bus master. The upper write strobe asserts during all word write operations and during byte write operations to the upper portion of the data bus (D15-D8). $\overline{WRU}$ remains negated during a read and a lower byte write cycle.
$\overline{WRL}$	<b>Lower Write Strobe</b> —This strobe is an output only signal that asserts during a write operation on the MC68322 bus. A write cycle can be initiated by the core, internal DMA, or external bus master. The lower write strobe asserts during all word write operations and during byte write operations to the lower portion of the data bus (D7-D0). $\overline{WRL}$ remains negated during a read and an upper byte write cycle.
$\overline{WAIT}$	<b>Wait</b> —This input only signal that extends an MC68322 bus cycle beyond the programmed values. Be aware that $\overline{WAIT}$ can only prolong bus cycles for chip-select banks.
$\overline{IRQ1-IRQ0}$	<b>External Interrupt Request</b> —These input only signals have programmable assertion levels and are used to connect external interrupting devices to the MC68322. These two signals are sent through the internal interrupt controller before posting an interrupt to the core.

## 2.5 DRAM INTERFACE

The following signals control the DRAM bus operation:

PIN NAME	DESCRIPTION
MA10–MA0	<b>Memory Address Bus</b> —These 11 output only signals connect to the internally multiplexed DRAM address bus. They directly drive the memory address bus to a DRAM array. The low-order address signals change to provide bursting capability. See Table 2-3 for a list of DRAM address multiplexing values.
MD15–MD0	<b>Memory Data Bus</b> —This signal connects to a 16-bit bidirectional three-stateable memory data bus. The memory data bus is used to transfer byte- and word-sized data to and from DRAM.
$\overline{\text{RAS5}}$ – $\overline{\text{RAS0}}$	<b>Row Address Strobe</b> —These output signals provide row address strobes for external DRAM. $\overline{\text{RASx}}$ asserts when a memory reference occurs that is internally decoded for the DRAM bank(s).
$\overline{\text{CAS1}}$ – $\overline{\text{CAS0}}$	<b>Column Address Strobe</b> —These output signals provide the column address strobe timing for the external DRAM. The $\overline{\text{CAS1}}$ signal asserts when a byte write operation occurs to the upper memory data bus (MD15-MD8). $\overline{\text{CAS0}}$ asserts when a byte write operation occurs to the lower memory data bus (MD7-MD0). However, both $\overline{\text{CAS1}}$ and $\overline{\text{CAS0}}$ assert for byte-sized read operations and word-sized read and write operations.
$\overline{\text{WE}}$	<b>Write Enable</b> —This output signal asserts when an external DRAM access write cycle is initiated providing the write control for external DRAM.

**Table 2-3. DRAM Address Multiplexer**

ROW ADDRESS	COLUMN ADDRESS	MEMORY ADDRESS	DRAM SIZE × 16 BITS
A10	A01	MA0	4 Mbit, 1 Mbit, 256 Kbit
A11	A02	MA1	4 Mbit, 1 Mbit, 256 Kbit
A12	A03	MA2	4 Mbit, 1 Mbit, 256 Kbit
A13	A04	MA3	4 Mbit, 1 Mbit, 256 Kbit
A14	A05	MA4	4 Mbit, 1 Mbit, 256 Kbit
A15	A06	MA5	4 Mbit, 1 Mbit, 256 Kbit
A16	A07	MA6	4 Mbit, 1 Mbit, 256 Kbit
A17	A08	MA7	4 Mbit, 1 Mbit, 256 Kbit
A18	A09	MA8	4 Mbit, 1 Mbit, 256 Kbit
A20	A19	MA9	4 Mbit, 1 Mbit
A22	A21	MA10	4 Mbit

## 2.6 DMA INTERFACE

The following signals control the DMA interface. They are used to transfer data from the MC68322 bus to DRAM or vice versa.

PIN NAME	DESCRIPTION
DREQ	<b>Data Request</b> —This input signal, whose polarity is programmable, is asserted by a peripheral device to request a transfer between the internal core bus and DRAM. The assertion of the DREQ signal starts a DMA operation.
$\overline{\text{DACK}}$	<b>Data Acknowledge</b> —This active-low output signal indicates that a DMA transfer is complete.

## 2.7 PRINTER COMMUNICATION INTERFACE

The following signals communicate with the print engine. Due to various interfaces with different print engines, some signals may not be needed.

PIN NAME	DESCRIPTION
$\overline{\text{CBSY}}$	<b>Command Busy</b> —This output only signal indicates that a command byte is being sent to the print engine.
$\overline{\text{SBSY}}$	<b>Status Busy</b> —This input only signal indicates that a status is ready to be received from the print engine.
CCLK	<b>Command Clock</b> —This bidirectional signal is used to clock command and status data between the MC68322 and the print engine. It is not a free running clock and remains inactive until $\overline{\text{CBSY}}$ or $\overline{\text{SBSY}}$ is asserted. The print engine or the MC68322 can supply CCLK. The direction of this pin is programmable and engine dependent.
$\overline{\text{CMD/STS}}$	<b>Command/Status</b> —This bidirectional signal is provided because some print engines require command and status on the same line. It is used to exchange command and status information between the print engine and the MC68322. The direction of this pin is programmable and engine dependent.
STS	<b>Status</b> —This input signal is used by the print engine to supply data to the MC68322. Data sent through this signal is synchronous with the CCLK.

## 2.8 PRINT ENGINE VIDEO CONTROLLER INTERFACE

The print engine video controller (PVC) interface consists of five signals designed to interface directly to most laser print engines and input/output polarities are programmable. The following signals are used to transfer data from the MC68322 to the print engine.

PIN NAME	DESCRIPTION
VCLK	<b>Video Shift Clock</b> —This input signal is a free-running clock that is used to drive the video transfer. The print engine or an onboard oscillator can supply VCLK.
$\overline{\text{FSYNC}}$	<b>Frame Synchronize</b> —This input only signal indicates frame synchronization. The print engine asserts the $\overline{\text{FSYNC}}$ signal to begin a page. The active polarity of this signal is programmable.
$\overline{\text{LSYNC}}$	<b>Line Synchronize</b> —This input signal indicates that the print engine is ready to accept data for the next scanline. The active polarity of this signal is programmable.
$\overline{\text{PRINT}}$	<b>Print Request</b> —This output signal indicates that the MC68322 is ready to begin printing. The assertion of this signal initiates the printing process. The active polarity of this pin is programmable.
$\overline{\text{VIDEO}}$	<b>Video Data</b> —This output signal provides the serial video data to the printer. The default polarity is low for active video and high for inactive video. The VIDEO output driver can sink and source 24 mA. The active polarity of this signal is programmable.

## 2.9 PARALLEL PORT INTERFACE

The MC68322 has 17 pins dedicated to parallel port communications. These pins are designed to interface to an IEEE 1284-compatible or compliant host and meet all electrical driver/receiver requirements for Level 2 compliance. All inputs are TTL-compatible and received with Schmitt triggers with over 200 mV of hysteresis. All outputs are symmetrical and can sink and source 16 mA at 0.4 and 3.0 V, respectively. This provides a direct connection (through series resistors) between the MC68322 and the parallel port connector, thus no external buffering or latching logic is required.

The following signal descriptions are for compatibility mode operation only. Control signals carry different meanings when other IEEE 1284 modes are used. When other modes are discussed, the IEEE 1284 signal name is provided in parentheses following the MC68322 pin name. Applications that do not require a parallel port can use these pins as general-purpose, individually controllable I/O pins.

PIN NAME	DESCRIPTION
PD7-PD0	<b>Parallel Port Data Bus</b> —This 8-bit, bidirectional, three-stateable bus is used to exchange data between an external host computer and the MC68322.
$\overline{\text{SELECTIN}}$	<b>Parallel Port Select In</b> —This input signal is used by the parallel port interface to request “on-line” status information.
$\overline{\text{STROBE}}$	<b>Data Strobe</b> —This input signal indicates when valid data is present on the parallel port data bus.
$\overline{\text{AUTOFD}}$	<b>Parallel Port Autofeed</b> —This input signal indicates autofeed control.
$\overline{\text{INIT}}$	<b>Initialization Input</b> —This input signal is used to initialize parallel port input control.
$\overline{\text{ACK}}$	<b>Parallel Port Acknowledge</b> —This output signal indicates that a transfer on the parallel port data bus is complete.
BUSY	<b>Parallel Port Busy</b> —This output signal indicates that the parallel port is busy.
SELECT	<b>Parallel Port Selected</b> —This output signal indicates that the device on the parallel port is “on-line” or “off-line”.
PERROR	<b>Parallel Port Error</b> —This output signal indicates that a problem exists with the paper in the printer. It could mean that the printer has a paper jam or is out of paper.
$\overline{\text{FAULT}}$	<b>Parallel Port Fault</b> —This output signal indicates that an error condition exists with the printer. It could mean that the printer is out of toner or has been taken offline.

## SECTION 3 THE CORE

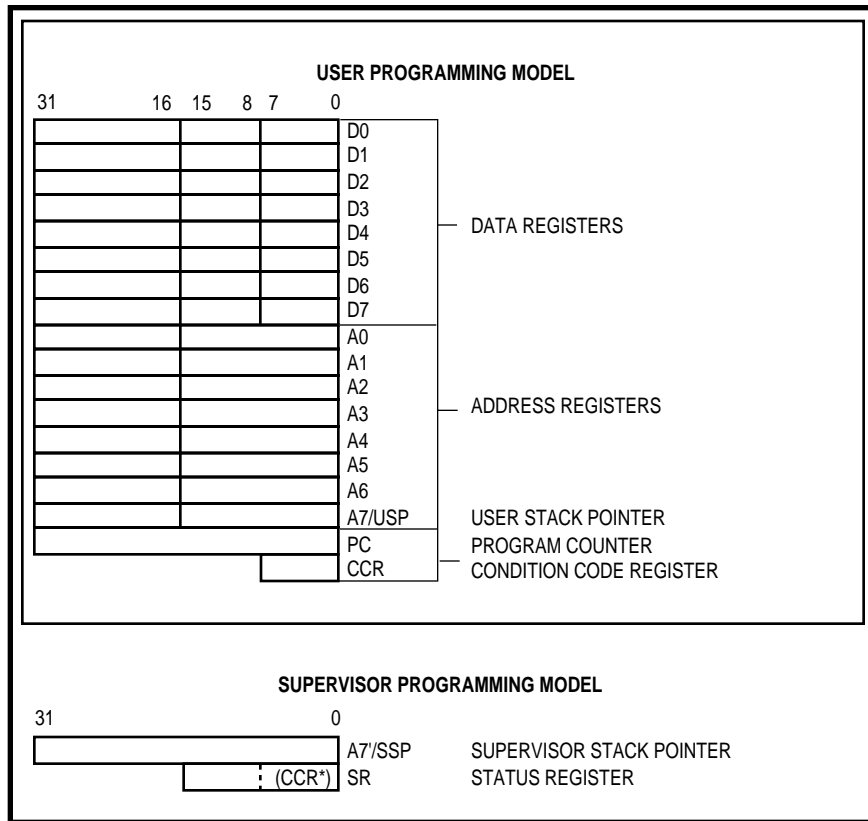
The MC68322 has an embedded EC000 core that controls its operation. The full architecture provides for 32-bit address and data register operations via the 16-bit data bus and internal 32-bit address bus. The core has the following features:

- Eight 32-bit address registers
- Eight 32-bit data registers
- 4G direct addressing range
- Fifty-six instructions
- Operations on five data types
- Memory-mapped input/output
- Fourteen addressing modes

### 3.1 PROGRAMMING MODEL

The EC000 core programming model is separated into two modes of access—user and supervisor. The user mode provides the execution environment for a majority of the application programs. The supervisor mode allows some additional instructions and privileges that the operating system and other system software use. The *M68000 Family Programmer's Reference Manual* can be another source for programming model information.





NOTE: \*CCR is also illustrated in the user programming model.

**Figure 3-1. EC000 Core Programming Model**

As illustrated in Figure 3-1, the user mode provides access to sixteen 32-bit, general-purpose registers, a 32-bit program counter, and an 8-bit condition code register. The first eight registers (D0–D7) are used as data registers for byte (8-bit), word (16-bit), and long-word (32-bit) operations. The second set of seven registers (A0–A6) and the user stack pointer (A7/USP) can be used as software stack pointers and base address registers. In addition, the address registers can be used for word and long-word operations. However, all 16 registers can be used as index registers. The condition code register provides information on integer overflow, zeros, negatives, carries, and extends. It is contained in the low-order byte of the status register.

The supervisor mode provides access to two supplementary registers—status register (high-order byte) and supervisor stack pointer (A7/SSP). The status register has access to the condition codes, but also includes the interrupt mask (in the high-order byte) with eight levels of interrupts available. It also indicates whether the core is in trace or supervisor mode.

## 3.2 DATA TYPES AND ADDRESSING MODES

The core supports the basic data formats of the M68000 Family. The instruction set supports operations on other data formats, such as memory addresses. The operand data formats supported by the core are the standard two's-complement data formats defined in the M68000 Family architecture. Registers, memory, or instructions themselves can contain integer unit operands. The operand size for each instruction is either explicitly encoded in the instruction or implicitly defined by the instruction operation. Table 3-1 lists the data formats for the core. Refer to the *M68000 Family Programmer's Reference Manual* for details on data format organization in registers and memory.

**Table 3-1. Processor Data Formats**

OPERAND DATA FORMAT	SIZE
Bit	1 Bit
Binary-Coded Decimal (BCD)	8 Bits
Byte Integer	8 Bits
Word Integer	16 Bits
Long-Word Integer	32 Bits

The core also supports the basic addressing modes of the M68000 Family. The register indirect addressing modes support postincrement, predecrement, offset, and index capabilities. The program counter relative mode also supports indexing and offsetting. Table 3-2 lists a summary of the data addressing modes for the core. Refer to the *M68000 Family Programmer's Reference Manual* for details on the core's effective addressing modes.

**Table 3-2. Effective Addressing Modes**

ADDRESSING MODES	SYNTAX
Register Direct Addressing Data Register Direct Address Register Direct	EA = Dn EA = An
Absolute Data Addressing Absolute Short Absolute Long	EA = (Next Word) EA = (Next Two Words)
Program Counter Relative Addressing Relative With Offset Relative With Index and Offset	EA = (PC) + d <sub>16</sub> EA = (PC) + d <sub>8</sub>
Register Indirect Addressing Register Indirect Postincrement Register Indirect Predecrement Register Indirect Register Indirect With Offset Indexed Register Indirect With Offset	EA = (An) EA = (An), An    An + N An    An -N, EA = (An) EA = (An) + d <sub>16</sub> EA = (An) + (Xn) + d <sub>8</sub>
Immediate Data Addressing Immediate Quick Immediate	DATA = Next Word(s) Inherent Data
Implied Addressing Implied Register	EA = SR, USP, SSP, PC

### 3.3 INSTRUCTION SET SUMMARY

Table 3-3 lists the notational conventions used throughout this manual and Table 3-4 summarizes the core instruction set by opcode. In the syntax descriptions, the left operand is the source operand and the right operand is the destination operand.

**Table 3-3. Notational Conventions**

SINGLE- AND DOUBLE-OPERAND OPERATIONS	
≠	Not equal.
+	Arithmetic addition or postincrement indicator.
–	Arithmetic subtraction or predecrement indicator.
×	Arithmetic multiplication.
÷	Arithmetic division or conjunction symbol.
~	Invert; operand is logically complemented.
L	Logical AND
V	Logical OR
≈	Logical exclusive OR
	Source operand is moved to destination operand.
	Two operands are exchanged.
<	Relational test; true if source operand is less than destination operand.
>	Relational test; true if source operand is greater than destination operand.
<operand>	Any double-operand operation.
<operand> tested	Operand is compared to zero and the condition codes are set appropriately.
<operand> sign-extended <operand>	All bits of the upper portion are made equal to the high-order bit of the lower portion.
<operand> shifted by <count>	The source operand is shifted by the number of count.
<operand> rotated by <count>	The source operand is rotated by the number of count.
bit number of <operand>	Selects a single bit of the operand.
OTHER OPERATIONS	
TRAP	1 S-bit of SR; SSP – 4 SSP; PC (SSP); SSP – 2 SSP; SR (SSP); Vector Address PC
STOP	Enter the stopped state, waiting for interrupts.
<operand>10	The operand is BCD; operations are performed in decimal.
If <condition> then <operations> else <operations>	Test the condition. If true, the operations after “then” are performed. If the condition is false and the optional “else” clause is present, the operations after “else” are performed. If the condition is false and “else” is omitted, the instruction performs no operation. Refer to the Bcc instruction description as an example.
REGISTER SPECIFICATION	
An	Any Address Register n (example: A3 is address register 3)
Ax, Ay	Source and destination address registers, respectively.
Dn	Any Data Register n (example: D5 is data register 5)

**Table 3-3. Notational Conventions (Continued)**

Dx, Dy	Source and destination data registers, respectively.
Rn	Any Address or Data Register
Rx, Ry	Any source and destination registers, respectively.
Xn	Index Register—An, Dn, or suppressed.
<b>DATA FORMAT AND TYPE</b>	
<fmt>	Operand Data Format: Byte (B), Word (W), Long (L)
<b>SUBFIELDS AND QUALIFIERS</b>	
#<xxx> or #<data>	Immediate data following the instruction word(s).
<b>SINGLE- AND DOUBLE-OPERAND OPERATIONS</b>	
( )	Identifies an indirect address in a register.
[ ]	Identifies an indirect address in memory.
d <sub>n</sub>	Displacement Value, n Bits Wide (example: d <sub>16</sub> is a 16-bit displacement).
<b>REGISTER NAMES</b>	
CCR	Condition Code Register (lower byte of status register)
PC	Program Counter
SR	Status Register
<b>REGISTER CODES</b>	
C	Carry Bit in CCR
cc	Condition Codes from CCR
N	Negative Bit in CCR
U	Undefined, Reserved for Motorola Use
V	Overflow Bit in CCR
X	Extend Bit in CCR
Z	Zero Bit in CCR
<b>STACK POINTERS</b>	
SP	Active Stack Pointer
SSP	Supervisor (Master or Interrupt) Stack Pointer
USP	User Stack Pointer
<b>MISCELLANEOUS</b>	
<ea>	Effective Address
<label>	Assembly Program Label
<list>	List of registers, for example D3–D0.

**Table 3-4. Instruction Set Summary**

OPCODE	OPERATION	SYNTAX
ABCD	Source <sub>10</sub> + Destination <sub>10</sub> + X Destination	ABCD Dy,Dx ABCD -(Ay), -(Ax)
ADD	Source + Destination Destination	ADD <ea>,Dn ADD Dn,<ea>
ADDA	Source + Destination Destination	ADDA <ea>,An
ADDI	Immediate Data + Destination Destination	ADDI #<data>,<ea>
ADDQ	Immediate Data + Destination Destination	ADDQ # <data>,<ea>
ADDX	Source + Destination + X Destination	ADDX Dy, Dx ADDX -(Ay), -(Ax)
AND	Source $\wedge$ Destination Destination	AND <ea>,Dn AND Dn,<ea>
ANDI	Immediate Data $\wedge$ Destination Destination	ANDI # <data>,<ea>
ANDI to CCR	Source $\wedge$ CCR CCR	ANDI # <data>, CCR
ANDI to SR	If supervisor state then Source $\wedge$ SR SR else TRAP	ANDI # <data>, SR
ASL, ASR	Destination Shifted by <count> Destination	ASd Dx,Dy ASd # <data>,Dy ASd <ea>
Bcc	If (condition true) then PC + d <sub>n</sub> PC	Bcc <label>
BCHG	~ (<bit number> of Destination) Z; ~ (<bit number> of Destination) <bit number> of Destination	BCHG Dn,<ea> BCHG # <data>,<ea>
BCLR	~ (<bit number> of Destination) Z; 0 <bit number> of Destination	BCLR Dn,<ea> BCLR # <data>,<ea>
BKPT	Run breakpoint acknowledge cycle; TRAP as illegal instruction	BKPT # <data>
BRA	PC + d <sub>n</sub> PC	BRA <label>
BSET	~ (<bit number> of Destination) Z; 1 <bit number> of Destination	BSET Dn,<ea> BSET # <data>,<ea>
BSR	SP - 4 SP; PC (SP); PC + d <sub>n</sub> PC	BSR <label>
BTST	~ (<bit number> of Destination) Z;	BTST Dn,<ea> BTST # <data>,<ea>
CHK	If Dn < 0 or Dn > Source then TRAP	CHK <ea>,Dn
CLR	0 Destination	CLR <ea>
CMP	Destination - Source cc	CMP <ea>,Dn
CMPA	Destination - Source cc	CMPA <ea>,An
CMPI	Destination - Immediate Data cc	CMPI # <data>,<ea>
CMPM	Destination - Source cc	CMPM (Ay)+, (Ax)+
DBcc	If condition false then (Dn - 1 Dn; If Dn $\neq$ -1 then PC + d <sub>n</sub> PC)	DBcc Dn,<label>
DIVS	Destination $\div$ Source Destination	DIVS.W <ea>,Dn32/16 16r:16q
DIVU	Destination $\div$ Source Destination	DIVU.W <ea>,Dn32/16 16r:16q

**Table 3-4. Instruction Set Summary (Continued)**

OPCODE	OPERATION	SYNTAX
EOR	Source $\oplus$ Destination    Destination	EOR Dn,<ea>
EORI	Immediate Data $\oplus$ Destination    Destination	EORI # <data>,<ea>
EORI to CCR	Source $\oplus$ CCR    CCR	EORI # <data>,CCR
EORI to SR	If supervisor state then Source $\oplus$ SR    SR else TRAP	EORI # <data>,SR
EXG	Rx    Ry	EXG Dx,Dy EXG Ax,Ay EXG Dx,Ay EXG Ay,Dx
EXT	Destination Sign-Extended    Destination	EXT.W Dn    extend byte to word EXT.L Dn    extend word to long word
ILLEGAL	SSP - 4    SSP; PC    (SSP); SSP - 2    SSP; SR    (SSP); Illegal Instruction Vector Address    PC	ILLEGAL
JMP	Destination Address    PC	JMP <ea>
JSR	SP - 4    SP; PC    (SP) Destination Address    PC	JSR <ea>
LEA	<ea>    An	LEA <ea>,An
LINK	SP - 4    SP; An    (SP) SP    An, SP + d <sub>n</sub> SP	LINK An, # <displacement>
LSL,LSR	Destination Shifted by <count>    Destination	LSd Dx,Dy LSd # <data>,Dy LSd <ea>
MOVE	Source    Destination	MOVE <ea>,<ea>
MOVEA	Source    Destination	MOVEA <ea>,An
MOVE to CCR	Source    CCR	MOVE <ea>,CCR
MOVE from SR	SR    Destination	MOVE SR,<ea>
MOVE to SR	If supervisor state then Source    SR else TRAP	MOVE <ea>,SR
MOVE USP	If supervisor state then USP    An or An    US else TRAP	MOVE USP,An MOVE An,USP
MOVEM	Registers    Destination; Source    Registers	MOVEM <list>,<ea> MOVEM <ea>,<list>
MOVEP	Source    Destination	MOVEP Dx,(d <sub>16</sub> ,Ay) MOVEP (d <sub>16</sub> ,Ay),Dx
MOVEQ	Immediate Data    Destination	MOVEQ # <data>,Dn
MULS	Source $\times$ Destination    Destination	MULS.W <ea>,Dn    16 x 16    32
MULU	Source $\times$ Destination    Destination	MULU.W <ea>,Dn    16 x 16    32
NBCD	0 - (Destination <sub>10</sub> ) - X    Destination	NBCD <ea>
NEG	0 - (Destination)    Destination	NEG <ea>
NEGX	0 - (Destination) - X    Destination	NEGX <ea>
NOP	None	NOP

**Table 3-4. Instruction Set Summary (Continued)**

OPCODE	OPERATION	SYNTAX
NOT	~Destination Destination	NOT <ea>
OR	Source V Destination Destination	OR <ea>,Dn OR Dn,<ea>
ORI	Immediate Data V Destination Destination	ORI # <data>,<ea>
ORI to CCR	Source V CCR CCR	ORI # <data>,CCR
ORI to SR	If supervisor state then Source V SR SR else TRAP	ORI # <data>,SR
PEA	Sp - 4 SP; <ea> (SP)	PEA <ea>
RESET <sup>1</sup>	If supervisor state then Assert $\overline{\text{RESET}}$ Line else TRAP	RESET
ROL, ROR	Destination Rotated by <count> Destination	ROd Dx,Dy ROd # <data>,Dy ROd <ea>
ROXL, ROXR	Destination Rotated with X by <count> Destination	ROXd Dx,Dy ROXd # <data>,Dy ROXd <ea>
RTE	If supervisor state then (SP) SR; SP + 2 SP; (SP) PC; SP + 4 SP; restore state and deallocate stack according to (SP) else TRAP	RTE
RTR	(SP) CCR; SP + 2 SP; (SP) PC; SP + 4 SP	RTR
RTS	(SP) PC; SP + 4 SP	RTS
SBCD	Destination <sub>10</sub> - Source <sub>10</sub> - X Destination	SBCD Dx,Dy SBCD -(Ax),-(Ay)
Scc	If condition true then 1s Destination else 0s Destination	Scc <ea>
STOP	If supervisor state then Immediate Data SR; STOP else TRAP	STOP # <data>
SUB	Destination - Source Destination	SUB <ea>,Dn SUB Dn,<ea>
SUBA	Destination - Source Destination	SUBA <ea>,An
SUBI	Destination - Immediate Data Destination	SUBI # <data>,<ea>
SUBQ	Destination - Immediate Data Destination	SUBQ # <data>,<ea>
SUBX	Destination - Source - X Destination	SUBX Dx,Dy SUBX -(Ax),-(Ay)
SWAP	Register [31:16] Register [15:0]	SWAP Dn
TAS	Destination Tested Condition Codes; 1 bit 7 of Destination	TAS <ea>
TRAP	1 S-bit of SR; SSP - 4 SSP; PC (SSP); SSP - 2 SSP; SR (SSP); Vector Address PC	TRAP # <vector>
TRAPV	If V then TRAP	TRAPV
TST	Destination Tested Condition Codes	TST <ea>
UNLK	An SP; (SP) An; SP + 4 SP	UNLK An

NOTES:

1. Does Not Reset External Devices Or Internal Registers. Has No Effect On Any MC68322 Modules.
2. d Is Direction, L, Or R.

## SECTION 4

# BUS OPERATION

The MC68322 bus consists of the address and data buses (A25–A0 and D15–D0 respectively). These are separate parallel buses that transfer data using an asynchronous protocol. The graphics bus consists of the MC68322's DRAM address and data buses (MA10–MA0 and MD15–MD0 respectively), which are separate parallel buses on which data is transferred to and from DRAM using an asynchronous protocol. This section describes control signals and bus operation during data transfers on the MC68322 bus and graphics bus. It also describes arbitration of the bus and external bus mastership.

Except during external bus mastership, the EC000 core directs the MC68322 bus and graphics bus using the following the  $\overline{CSx}$ ,  $\overline{RD}$ ,  $\overline{WRU}$ ,  $\overline{WRL}$ ,  $R/\overline{W}$ ,  $\overline{RAS5}$ – $\overline{RAS0}$ ,  $\overline{CAS1}$ – $\overline{CAS0}$ , MA10–MA0, and  $\overline{WE}$  control signals to transfer data. These control signals indicate the bus cycle beginning and type as well as the address space and size of the transfer. They control all transfers to and from the core including:

- Internal registers read or write
- ROM read
- DRAM read or write
- I/O read or write

### 4.1 EC000 CORE READ CYCLE

During a read cycle, the core receives data from memory (DRAM or EPROM), an internal register, or a peripheral device; reading either one or two bytes of data in all cases. If the instruction specifies a word (or long-word) operation, the core reads both upper and lower bytes simultaneously. When the instruction specifies a byte operation, the core uses the internal A0 bit to determine which byte to read. Once the data is received, the core correctly positions the byte internally. Figure 4-1 illustrates a word- and byte-sized read cycle flowchart. Figure 4-2 illustrates the internal word read cycle timing diagram to chip-selects. Figure 4-3 illustrates the internal byte- and word-sized read cycle timing diagram to chip-selects, while Figure 4-4 illustrates the internal byte- and word-sized read cycle timing diagram to DRAM.



**Note:**  $\overline{AS}$  asserts for MC68322 chip-select bus operations only.



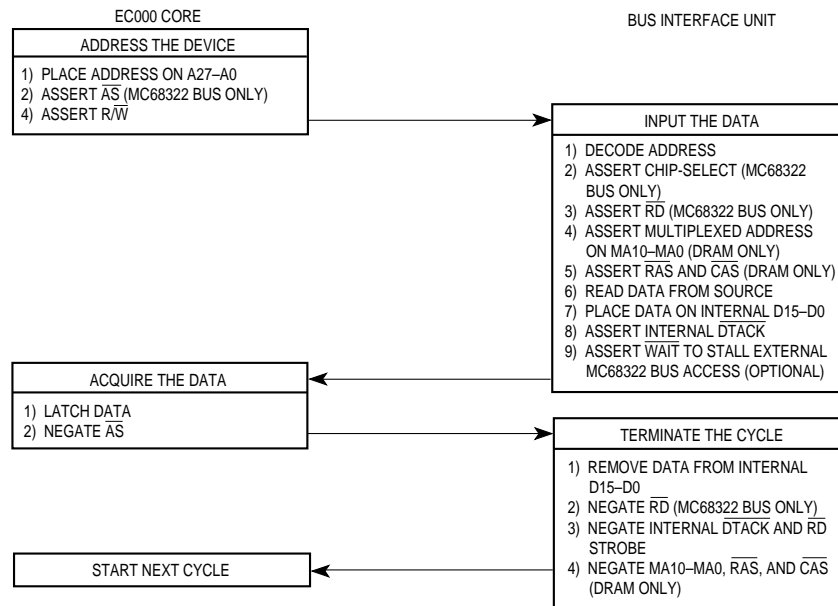
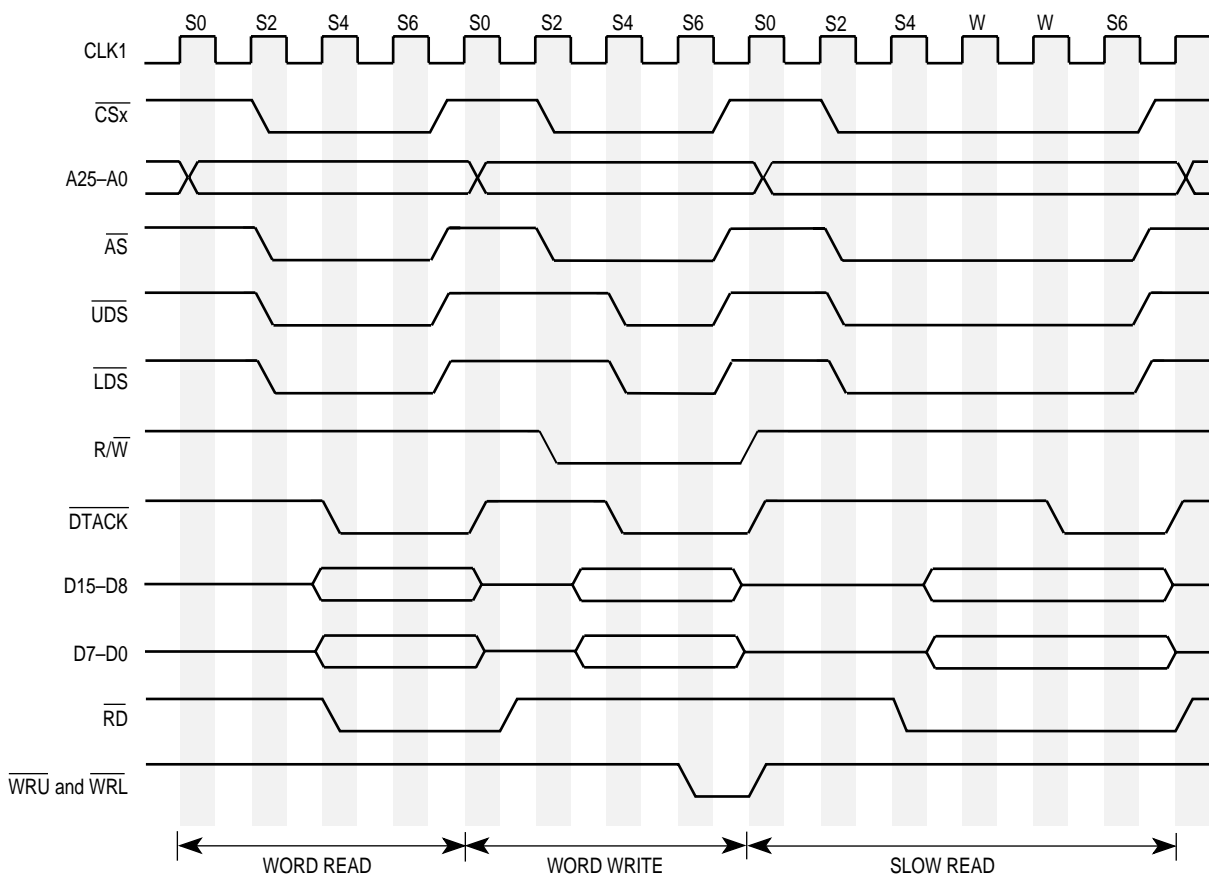
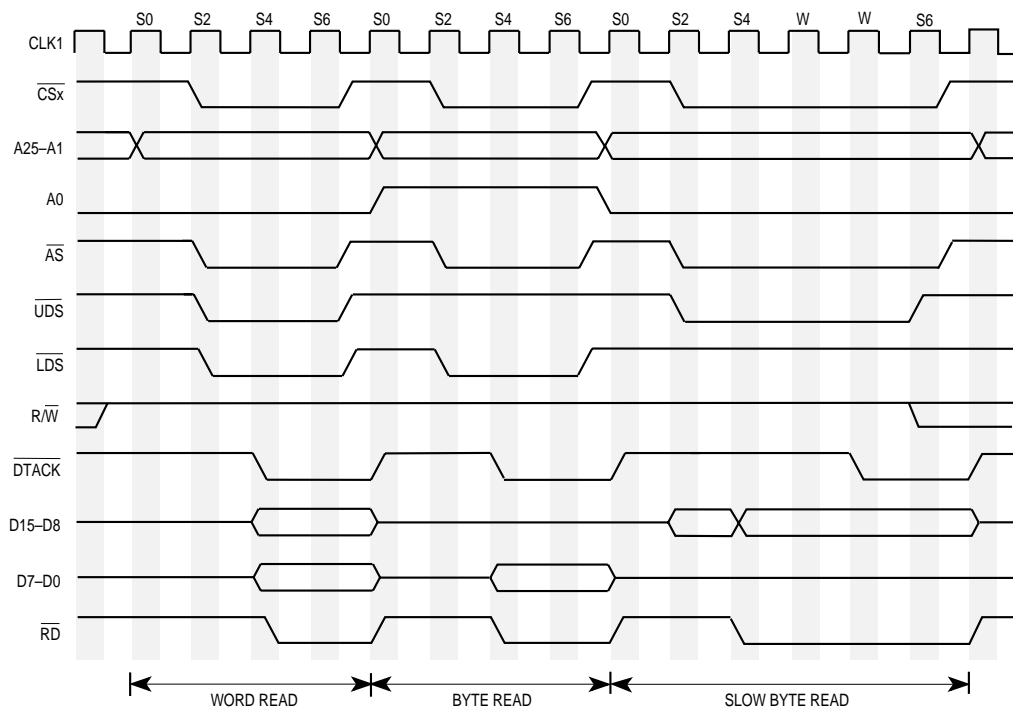


Figure 4-1. Read Cycle Flowchart



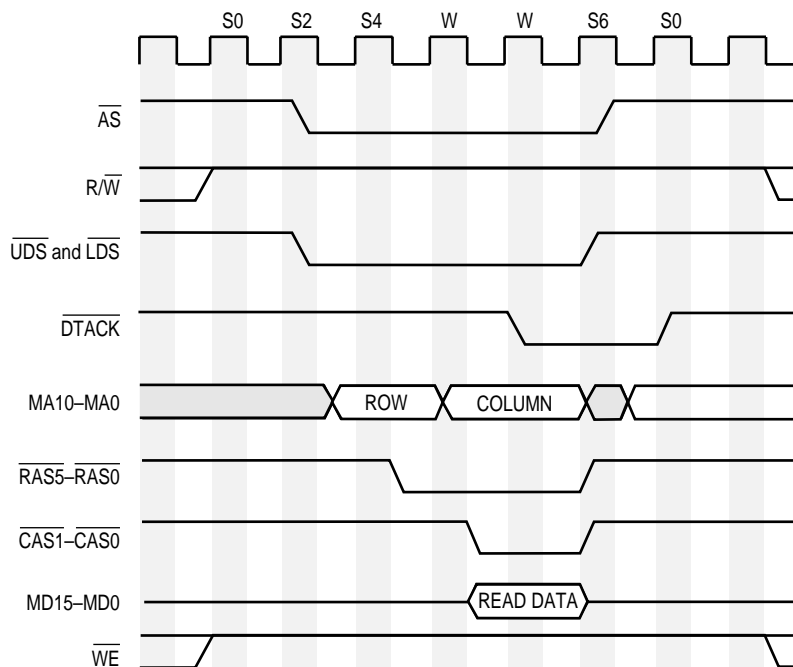
NOTE:  $\overline{UDS}$ ,  $\overline{LDS}$ , and  $\overline{DTACK}$  are internal signals only.

Figure 4-2. External Timing Diagram to Chip-Selects Banks



NOTE: A0, UDS, LDS, and DTACK are internal signals only.

**Figure 4-3. External Timing Diagram to Chip-Select Banks**



NOTE: UDS, LDS, and DTACK are internal signals only.

**Figure 4-4. Word and Byte Read Cycle Timing Diagram to DRAM**

## 4.2 EC000 CORE WRITE CYCLE

During a write cycle, the core sends data to memory (DRAM or EPROM), an internal register, or a peripheral device; writing bytes of data in all cases. If the instruction specifies a word (or long-word) operation, the core writes both upper and lower bytes simultaneously. When the instruction specifies a byte operation, the core uses the internal A0 to determine which byte to write and then issues the data strobe required for that byte.

A transfer is initiated by asserting the address strobe ( $\overline{AS}$ ) and providing a valid internal function code and address. The address is decoded by each module. Once a transfer is initiated, the core uses the timing and wait state characteristics for the active interface to pace the internal core bus cycle. Using the wait state information of the selected module, the core waits the specified number of cycles, transfers data, and then terminates the cycle by asserting the internal  $\overline{DTACK}$  signal to the core. The core, in turn, then terminates the internal core bus cycle within two clocks (S4–S7). All transfers on the internal core bus require the minimum access time of four clock cycles. Figure 4-5 illustrates the write cycle flowchart and Figure 4-6 illustrates the word and byte write cycle timing diagram to chip-selects.

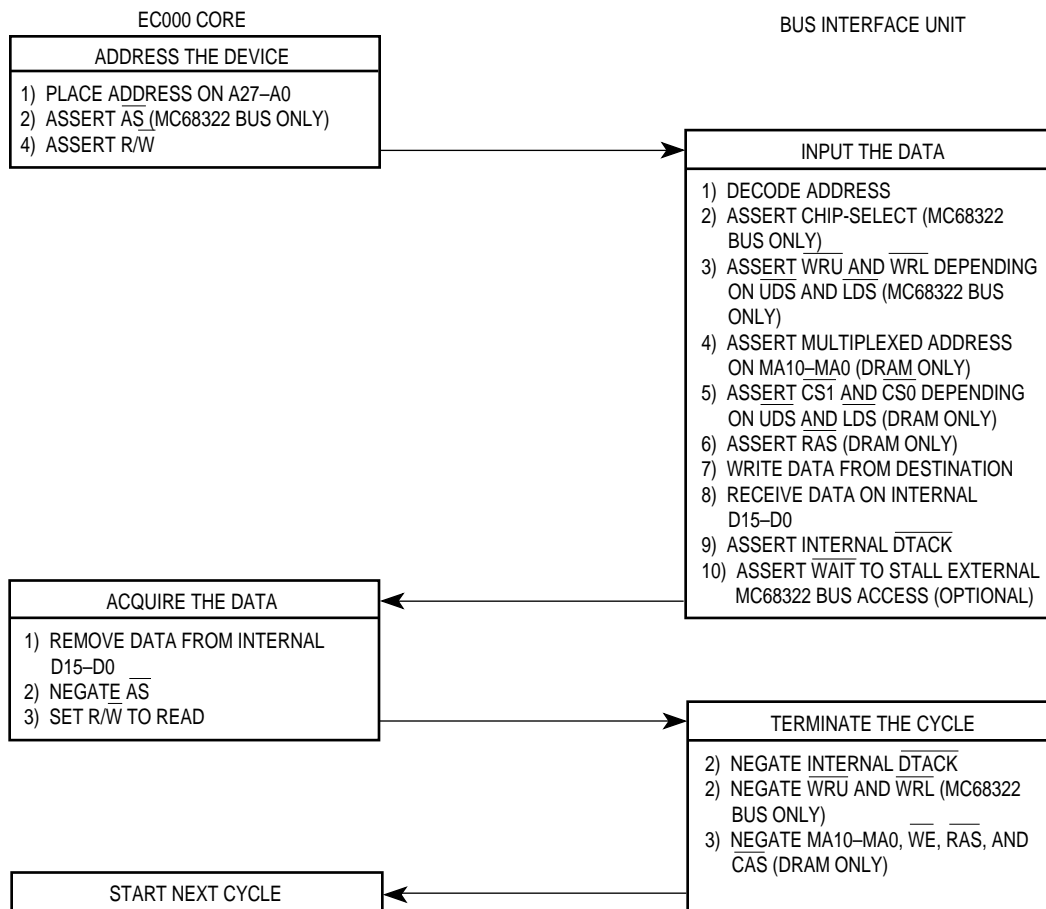
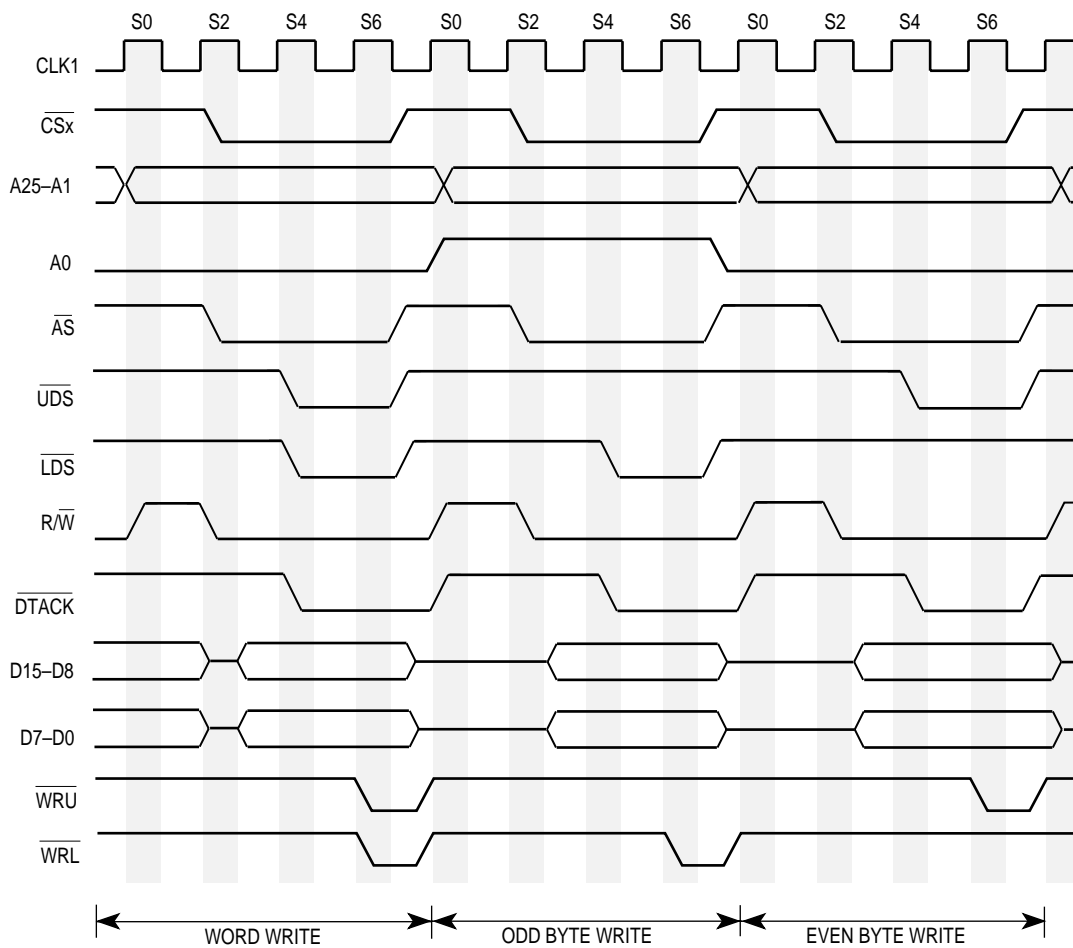


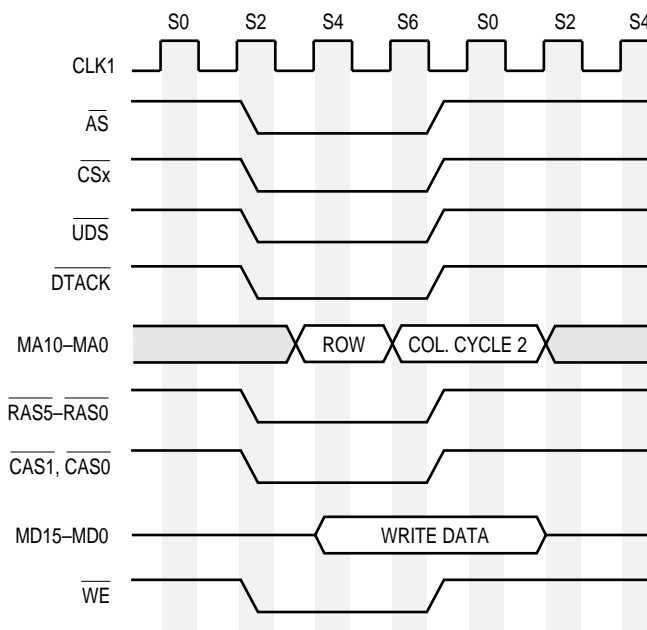
Figure 4-5. Write Cycle Flowchart



NOTE: A0,  $\overline{UDS}$ ,  $\overline{LDS}$ , and  $\overline{DTACK}$  are internal signals only.

**Figure 4-6. Word and Byte Write Cycle Timing Diagram to Chip-Selects**

When the core initiates a DRAM write cycle, the data is latched in an internal write buffer and a zero wait-state  $\overline{DTACK}$  is generated to the core. The data from the write buffer is written to DRAM through the graphics bus as soon as the bus is available. Subsequent write cycles to DRAM are held off if the write buffer is full. The write buffer allows the core to execute the next instructions while waiting for data to be written to DRAM, thus increasing the processor's performance. Figure 4-7 illustrates the word and byte write cycle timing diagram to DRAM.



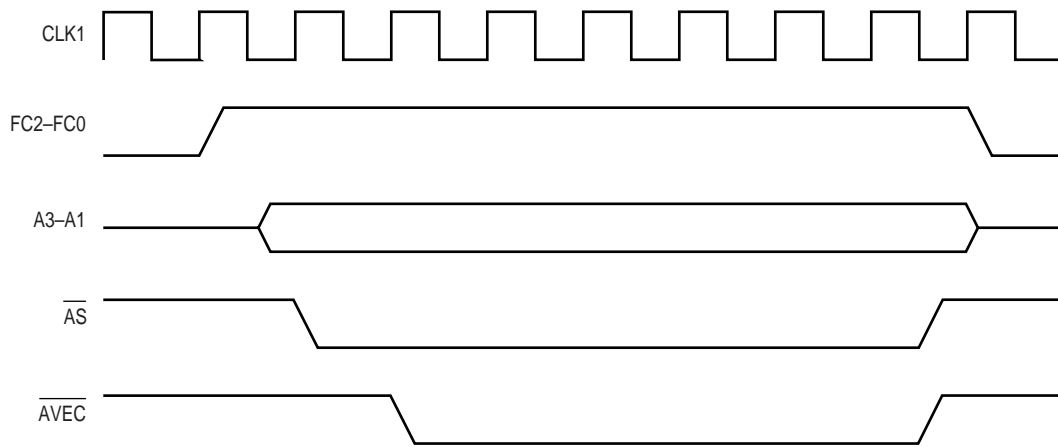
NOTE:  $\overline{UDS}$  and  $\overline{DTACK}$  are internal signals only.

**Figure 4-7. Word Write Cycle Timing Diagram to DRAM**

### 4.3 INTERRUPT ACKNOWLEDGE BUS CYCLE

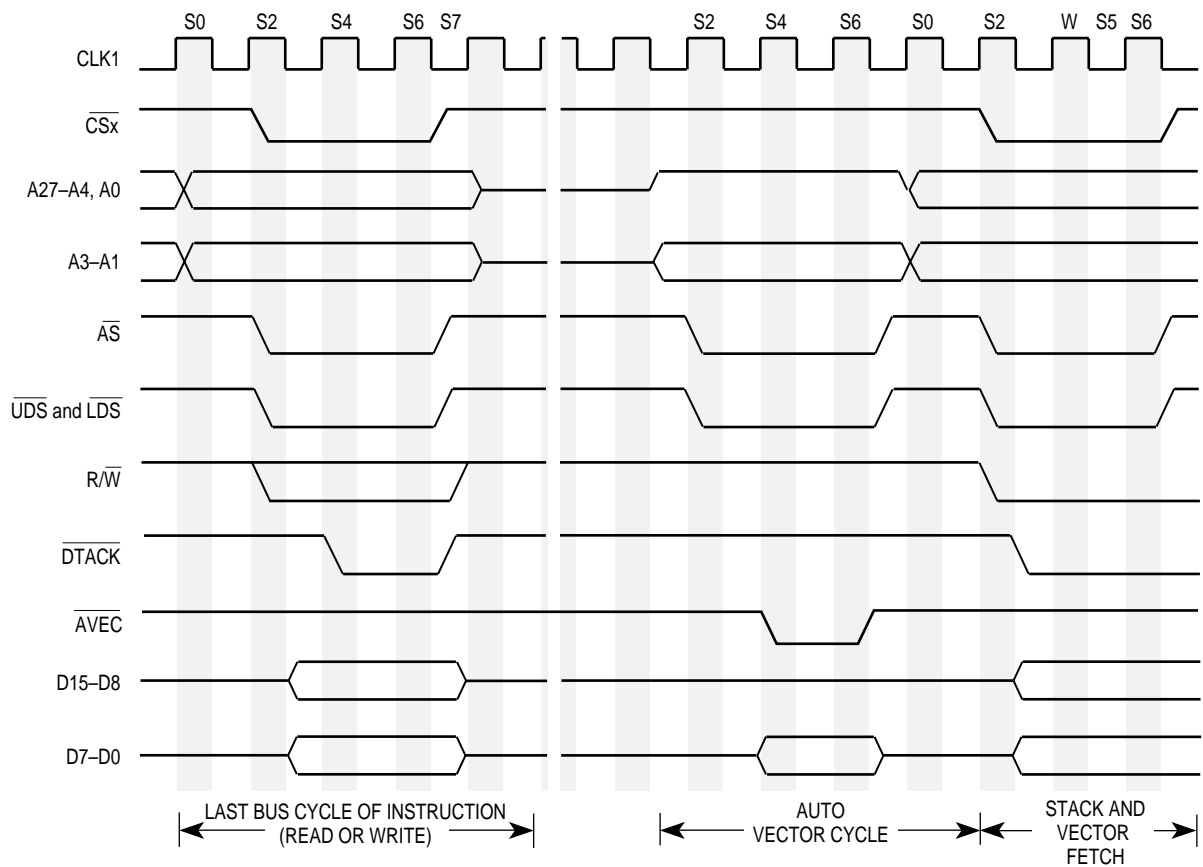
Whenever an interrupt is generated by one of the modules on the MC68322 (or when an external interrupt arrives), a bit will be set in the corresponding interrupt event register and the internal  $\overline{IPLX}$  signals will be asserted according to the level programmed for that module. In response to these active internal  $\overline{IPLx}$  signals, the core transparently initiates an interrupt acknowledge cycle. During this cycle, the core sets all internal function code signals (FC2–FC0) high, displays the interrupt level on internal A3–A1, drives A23–A4 and the internal A0 high, and initiates a read cycle. All MC68322 interrupts are autovectored.

During the interrupt acknowledge bus cycle, with  $\overline{AS}$  detected, an internal  $\overline{AVEC}$  signal is asserted to the core as an indication that the interrupt is an automatic vector interrupt.  $\overline{AVEC}$  is negated when  $\overline{AS}$  goes inactive. The internal  $\overline{IPL2}$ – $\overline{IPL0}$  signals remain asserted until the core clears the interrupt event by clearing the appropriate bit in the module's interrupt event register. The  $\overline{IPLx}$  signals can, however, change from one level to another before they are serviced if a different level interrupt is generated. As part of the interrupt handling routine, the interrupt sources of that level should be cleared. Figure 4-8 illustrates the internal interrupt acknowledge timing diagram and Figure 4-9 illustrates the interrupt acknowledge cycle timing diagram.



NOTE: These signals are internal signals only.

**Figure 4-8. Internal Interrupt Acknowledge Cycle**

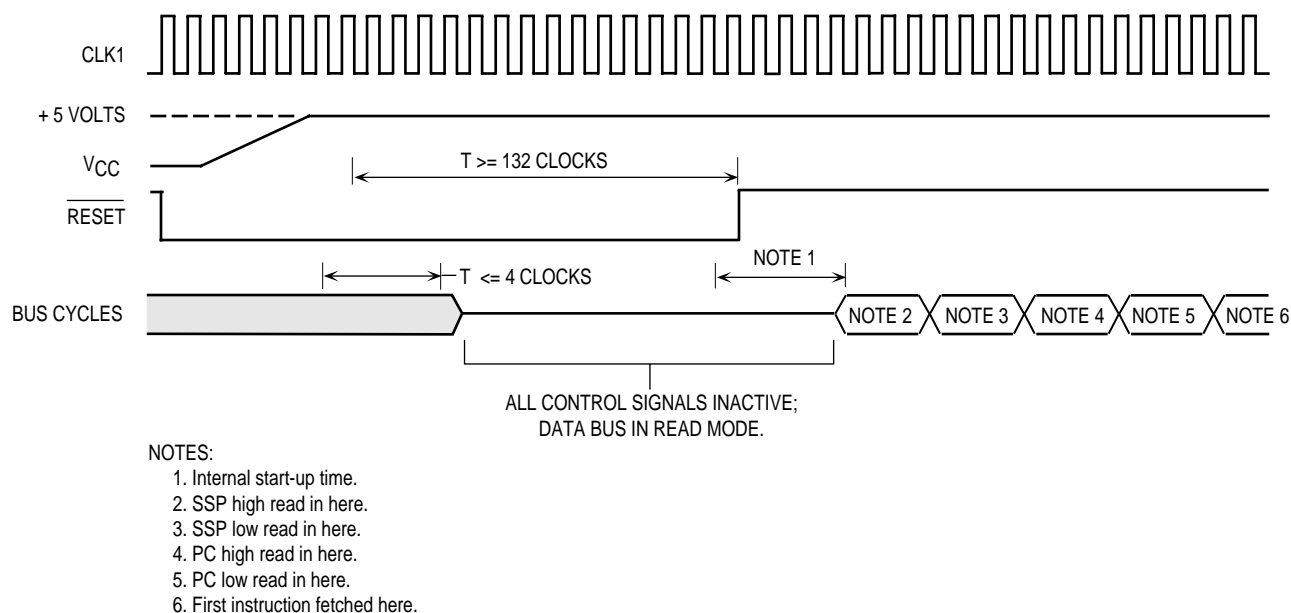


NOTE: A0,  $\overline{UDS}$ ,  $\overline{LDS}$ ,  $\overline{AVEC}$ , and  $\overline{DTACK}$  are internal signals only.

**Figure 4-9. Interrupt Acknowledge Cycle Timing Diagram**

## 4.4 RESET OPERATION

$\overline{\text{RESET}}$  is externally asserted for the initial processor reset. When  $\overline{\text{RESET}}$  is driven by an external device, the entire system (including the core and internal registers) are reset. Resetting the MC68322 initializes the internal state. The processor reads the reset vector table entry (address  $000_{16}$ ) and loads the contents into the supervisor stack pointer. Next, the processor loads the contents of address  $004_{16}$  (vector table entry 1) into the program counter. Refer to Table 5-3 for more information about exception vector assignments. The processor then initializes the interrupt level in the status register to a value of seven. Figure 4-10 illustrates the timing of the reset operation.  $\overline{\text{RESET}}$  must be asserted for at least 132 clocks for initial reset. For a subsequent external reset, asserting this signal for 10 clock cycles or longer resets the MC68322.



**Figure 4-10. Reset Operation Timing Diagram**

If the core executes the **reset** instruction, the internal state of the MC68322, its internal registers, and its external devices are all unaffected. Neither the status register nor any of the internal registers are affected by a **reset** instruction execution. Also, the  $\overline{\text{RESET}}$  signal will not assert during the execution of a **reset** instruction. However, using the **reset** instruction on the MC68322 is not recommended.

## 4.5 EXTERNAL BUS MASTER

The design of the MC68322 bus allows only one bus master at a given time. The core can be disabled as the bus master so that DMA or an external device can have full access to MC68322 resources. This provides a significant improvement in performance and takes full advantage of the high integration in the MC68322. The design also provides the external bus master with access to DRAM, chip-selects, or register memory resources in the lower 64M of memory. However, the external bus master cannot access memory above this 64M boundary. A handshake between the MC68322 and the external bus master achieves the exchange of bus mastership.

When the MC68322 is in external bus master mode, the external bus master drives many of the signals into the MC68322. The external bus master is limited in that it is unable to detect generated interrupts, can perform only word-sized operations, and cannot perform read-modify-write cycles.

### 4.5.1 MC68322 Bus Arbitration

Bus arbitration is the protocol by which an external device or DMA becomes the MC68322 bus master. The bus interface unit manages the bus arbitration signals so that the core alternates cycles between GDMA and the external master. Systems having several devices that become bus master require external circuitry to assign priorities to the devices. So when two or more external devices try to become bus master at the same time, the one having the highest priority is the bus master first. These external devices must assert the bus arbitration signals in the following sequence:

1. An external device asserts the bus request ( $\overline{BR}$ ) signal. This can be a wire-ORed signal (although it need not be constructed from open-collector devices) that informs the MC68322 that an external device requires control of the bus.
2. The MC68322 three-states A25–A1, D15–D0,  $\overline{AS}$ , and R/ $\overline{W}$ . Then it asserts the bus grant ( $\overline{BG}$ ) signal to indicate the bus is available.
3. The external device controls the bus cycle by driving the control signals and the MC68322 asserts  $\overline{EDTACK}$  to denote the end of each external bus master cycle.

$\overline{BR}$  can be issued at any time during a bus cycle or between cycles.  $\overline{BG}$  is asserted when the bus is available. When the requesting device receives  $\overline{BG}$  and more than one external device can be bus master, the requesting device should begin whatever arbitration is required. The external device asserts and maintains  $\overline{BR}$  during the entire bus cycle (or cycles) for which it is bus master. The following conditions must be met for an external device to assume mastership of the bus through the normal bus arbitration procedure:

1. The external device must have received the  $\overline{BG}$  signal through the arbitration process.
2. In a multiple bus master situation, it is important to be sure that only one processor has the bus at any given time.



The external bus master can negate  $\overline{BR}$  as soon as the MC68322 asserts  $\overline{EDTACK}$ . The bus is then granted back to the core when  $\overline{BR}$  is negated. If  $\overline{BR}$  is negated before  $\overline{EDTACK}$  is asserted,  $\overline{BG}$  remains active until  $\overline{EDTACK}$  is asserted and negated. The external bus master should not assert  $\overline{BR}$  for long periods of time because this would starve the core and other modules for memory cycles. DRAM accesses and refreshes continue regardless of bus arbitration. Figure 4-11 illustrates a bus arbitration timing diagram.

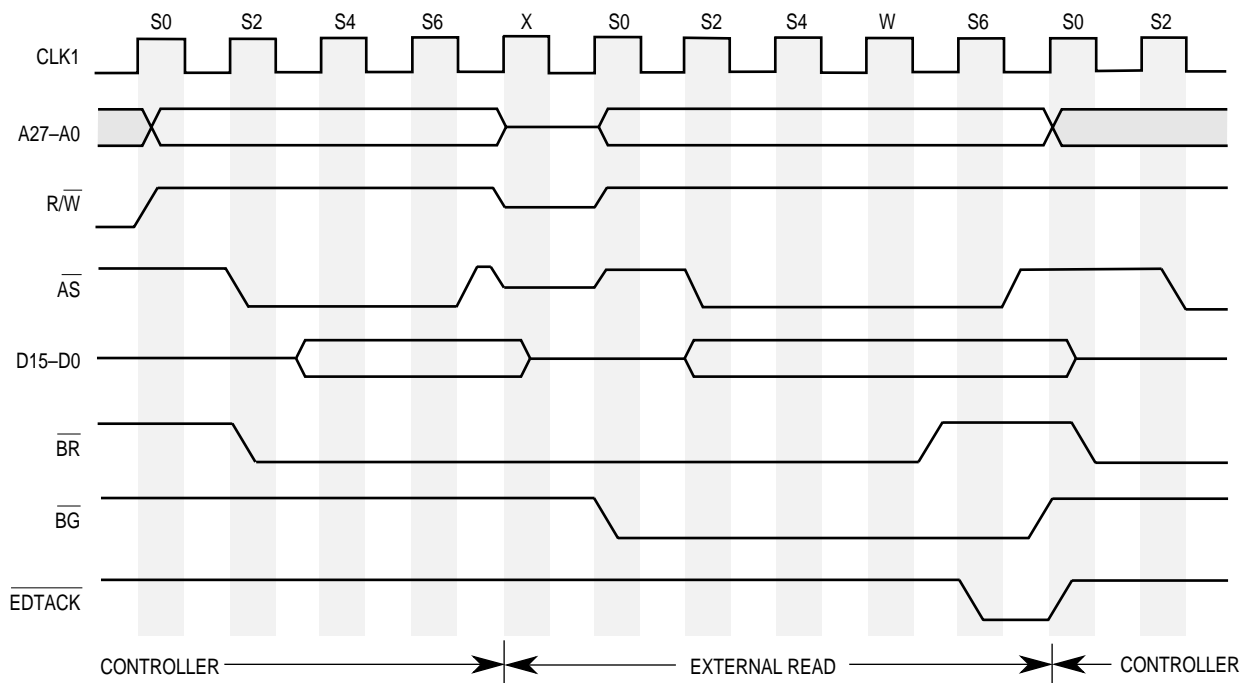
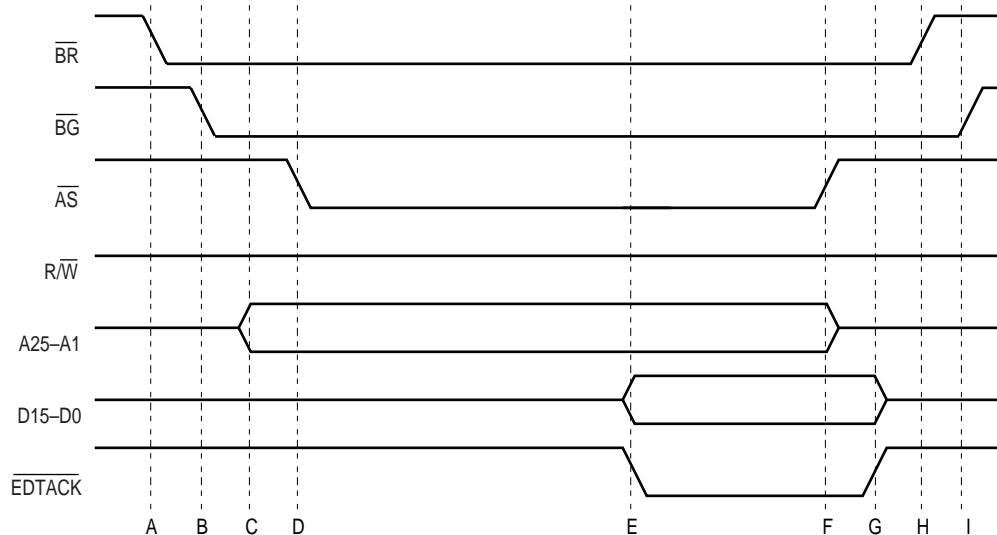


Figure 4-11. Bus Arbitration Timing Diagram

### 4.5.2 External Bus Master Read Cycle

When the external bus master reads from DRAM, chip-selects, or internal registers, the data is internally latched at the end of the read cycle. The MC68322 asserts  $\overline{EDTACK}$  and drives the data out onto the data bus. When the external bus master reads from the chip-selects, the data is latched on the CLK1 that  $\overline{RD}$  is negated. The chip-select devices stop driving the data bus when  $\overline{RD}$  is negated. When the hold time is satisfied, then the MC68322 drives the data bus with the data and  $\overline{EDTACK}$  is asserted. The hold time provides a data turnaround time starting from the time the chip-select device drives the data to the time the MC68322 drives the data. Read data is valid while  $\overline{EDTACK}$  is asserted. When the external bus master receives  $\overline{EDTACK}$ , it is free to negate  $\overline{AS}$  and to stop driving the address bus. Only word-sized read cycles are supported. Figure 4-12 illustrates a read cycle from the external bus master.



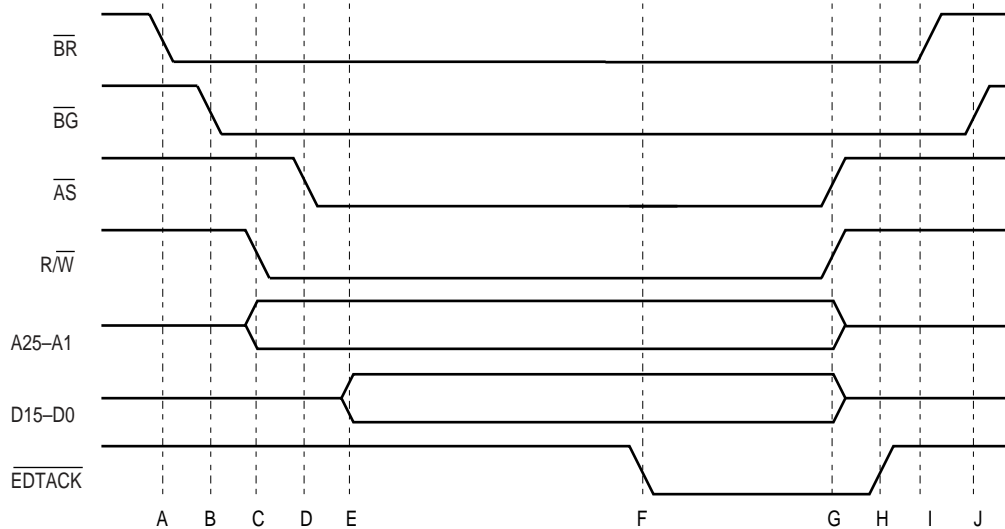
NOTES:

- A = Master requests bus by asserting  $\overline{BR}$ .
- B = MC68322 grants bus by asserting  $\overline{BG}$ .
- C = Master drives  $A_{25-A1}$ .
- D = Master asserts  $\overline{AS}$ .
- E = MC68322 drives  $D_{15-D0}$  and asserts  $\overline{EDTACK}$ .
- F = Master negates  $\overline{AS}$  and stops driving  $A_{25-A1}$ .
- G = MC68322 stops driving  $D_{15-D0}$  and negates  $\overline{EDTACK}$ .
- H = Master negates  $\overline{BR}$ .
- I = MC68322 negates  $\overline{BG}$ .

**Figure 4-12. External Bus Master Read Cycle**

### 4.5.3 External Bus Master Write Cycle

When the external bus master writes to DRAM, chip-selects, or internal registers, it asserts  $\overline{EDTACK}$  when the write cycle to the device is complete. The write data on the data bus goes directly to the device being written. Once the external bus master receives  $\overline{EDTACK}$ , it is free to negate  $\overline{AS}$  and to stop driving the data and address buses. Only word-sized write cycles are supported and there is no data strobe signal available to the external bus master. The external bus master is required to supply valid data from the beginning of the cycle. Figure 4-13 illustrates a write cycle from the external bus master.



NOTES:

- A = Master requests bus by asserting  $\overline{BR}$ .
- B = MC68322 grants bus by asserting  $\overline{BG}$ .
- C = Master drives A1-A25 and asserts  $\overline{R/W}$ .
- D = Master asserts  $\overline{AS}$ .
- E = Master drives D0-D15.
- F = MC68322 asserts  $\overline{EDTACK}$ .
- G = Master deasserts  $\overline{AS}$  and stops driving D0-D15 and A1-A25.
- H = MC68322 deasserts  $\overline{EDTACK}$ .
- I = Master deasserts  $\overline{BR}$ .
- J = MC68322 deasserts  $\overline{BG}$ .

**Figure 4-13. External Bus Master Write Cycle**

### 4.5.4 Illegal Address Interrupt

If the external bus master accesses a memory location not mapped to a register, DRAM, or chip-select, the MC68322 asserts  $\overline{EDTACK}$  immediately and waits for  $\overline{AS}$  to be negated before it negates  $\overline{EDTACK}$ . This bad access by the external bus master causes an illegal address interrupt bit to be set in the software interrupt event register. This causes a level 7 interrupt event to be posted to the core, but no interrupts are posted to the external master.

## SECTION 5

# INTERRUPT AND EXCEPTION HANDLING

The MC68322 supports two types of interrupts—internal and external. These interrupts are posted to the EC000 core through an internal interrupt controller, which uses an exception processing routine to handle the interrupt. The core's internal status register contains a 3-bit interrupt priority mask that ranges from level 0 to 7 (level 7 being the highest) mask level. Interrupts are inhibited for all priority levels less than or equal to the current mask level. Priority level 7 is a special case. Level 7 interrupts cannot be inhibited by the interrupt priority mask, thus providing a nonmaskable interrupt capability. Level 7 interrupts can be generated in two ways. First, an interrupt is generated each time the interrupt event level changes from a level below level 7. Second, a level 7 interrupt occurs if the interrupt request level is a 7 and the core priority mask is dropped from level 7 to a lower level by the execution of an instruction.

### 5.1 INTERNAL INTERRUPTS

The MC68322 modules function simultaneously and independently of the core. At the completion of a module's operation, the module is capable of posting an internal interrupt to the core through an internal interrupt controller. For example, the printer video controller (PVC) could post a page-end interrupt to signal the core that it has finished rendering a page. The controller ranks the interrupt based on a module's programmed interrupt priority level and then posts the interrupt event to the core. This type of internal interrupt event is called a hardware interrupt because it is initiated by one of the MC68322 modules.

The MC68322 supports another type of interrupt event called a software interrupt. This type of interrupt is initiated by the software by programming some internal registers on the MC68322. Once the software has initiated such an event, the interrupt will be posted to the core in exactly the same way as a hardware interrupt event. The advantage that these MC68322 hardware interrupts provide over the typical 68000 Family software exceptions is that these software interrupts can be set at a lower priority level than the current interrupt event and thus be delayed until the core runs at a lower priority level. This causes the software interrupt to be serviced sometime between the current interrupt and the next noninterrupt operation. See **Section 5.4.2.3 Instruction Traps** for more information.

### 5.1.1 Hardware Interrupts

There are 31 hardware interrupt events that can be posted from a module to the core. The interrupt events are grouped together by module and listed in Table 5-1. Each module has a corresponding interrupt level register, which is used to set the needed interrupt level for the module. Each module's interrupt event register is individually described in the section detailing the corresponding module.

**Table 5-1. Hardware Interrupt Events**

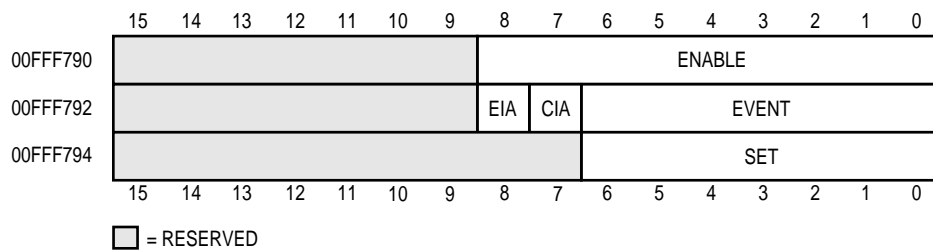
POSTING MODULE	INTERRUPT EVENTS
PVC	Illegal DRAM Address Video Underrun Band Underrun Page End Band Begin
RGP	Error Done
Printer Communication	Command Sent Status Received
Timer	Timer Interrupt
GDMA	Illegal Address DMA Complete Terminal Count Reached
PDMA	Illegal Address DMA Complete Terminal Count Reached
PPI	SELECTIN High SELECTIN Low STROBE High STROBE Low AUTOFD High AUTOFD Low INIT High INIT Low Data Received ECP Command Received Invalid Termination
External Interrupt 0	External Interrupt Source 0
External Interrupt 1	External Interrupt Source 1
External Interrupt 2	External Interrupt Source 2
External Interrupt 3	External Interrupt Source 3

A module's interrupt event register contains three basic parts—interrupt enable field, interrupt event bit fields, and interrupt level field. When an enable bit is set, it allows the interrupt event to cause an interrupt. When clear, the interrupt is masked. An interrupt event bit reflects the status of an interrupt event, regardless of the masking in the enable field. To determine the source of the interrupt, the software reads these bits and applies the mask in the enable field. The interrupt event bits are cleared with the software by writing a 1 to the bit positions to be cleared in the module's interrupt event register.

The interrupt level field reflects the interrupt priority level for that module. Level 7 is the highest priority, level 1 is the lowest, and level 0 indicates that no interrupt is requested. If an event occurs and causes the same level of interrupt as is currently being serviced (at the same time an interrupt of that level is being cleared) the interrupt level will become active again after two clocks. The interrupt level to the core could change if the priority level of an active interrupt is changed. To avoid potential problems, priority levels should be changed only while the corresponding interrupt is masked. Some interrupt event registers have status bits in addition to the interrupt bits. These status bits cannot cause an interrupt to the core. However, the software can read them at any time to obtain more information about the module's status.

### 5.1.2 Software Interrupts

There are seven independent software interrupts that can be set and cleared under software control; each corresponding to levels 7 through 1. Software can read the software interrupt event register for status information (or write to it to clear) or set an interrupt. Like the basic format for each module's interrupt event registers described above, the software interrupt event register contains an enable field, several software status fields, and the software interrupt event bits. Figure 5-1 illustrates the software interrupt event register.



**Figure 5-1. Software Interrupt Event Register**

Software interrupts are set by writing to the bits of the software interrupt event register's SET field. Writing a 1 to Bit 0 of the SET field generates a level 1 interrupt, Bit 1 a level 2, and so on. The EVENT field reflects the state of the pending software interrupt. Software interrupts can be set at any time and multiple interrupts can be set at the same time, if needed. However, keep in mind that setting an interrupt that has already been set will have no effect. Software interrupts are cleared by writing the EVENT field. Each bit corresponds to one of the seven software interrupts. Writing a 1 to a bit position will clear that software interrupt, while a zero in a bit position has no effect. For Bit 0, writing a 1 clears a level 1 interrupt, for Bit 6 writing a 1 clears a level 2 interrupt, and so on. The interrupt service routine must clear its interrupt to avoid another interrupt when the routine completes.

The MC68322 completely decodes the address map and generates an internal  $\overline{DTACK}$ . If the core presents an address that does not match any of the programmed addresses for the modules, none of the modules will generate  $\overline{DTACK}$ . This condition is called an illegal address. When an illegal address is detected, the core interface asserts an internal  $\overline{DTACK}$  and sets a level 7 interrupt to indicate an error condition. The software interrupt event register's CIA bit is then set, thus indicating that the core has accessed a memory location that is not mapped to a register, chip-select, or DRAM. The interrupt is cleared by writing a 1 to the CIA bit position. The core will never have a bus error condition. The EIA bit is set if the external bus master accesses a memory location that is not mapped to a register, chip-select, or DRAM. This interrupt generates a level 7 interrupt to the core. The interrupt is cleared by writing a 1 to the EIA bit position.

## 5.2 EXTERNAL INTERRUPTS

The MC68322 provides up to four pins for external interrupts (IRQ3–IRQ0). IRQ1 and IRQ0 are dedicated interrupt pins. IRQ3 and IRQ2 are multiplexed pins and must be programmed to be used as interrupt pins. Refer to **Appendix D Alternate Pin Functions** for more details. These inputs can be individually programmed for active polarity and either level or edge sensitivity. External interrupts are asynchronous to the MC68322 and are synchronized inside the MC68322 for proper operation. Two registers control each source of external interrupt—external interrupt 0/2 and 1/3 registers (EXIR0/2–EXIR1/3). Figure 5-2 illustrates these registers.

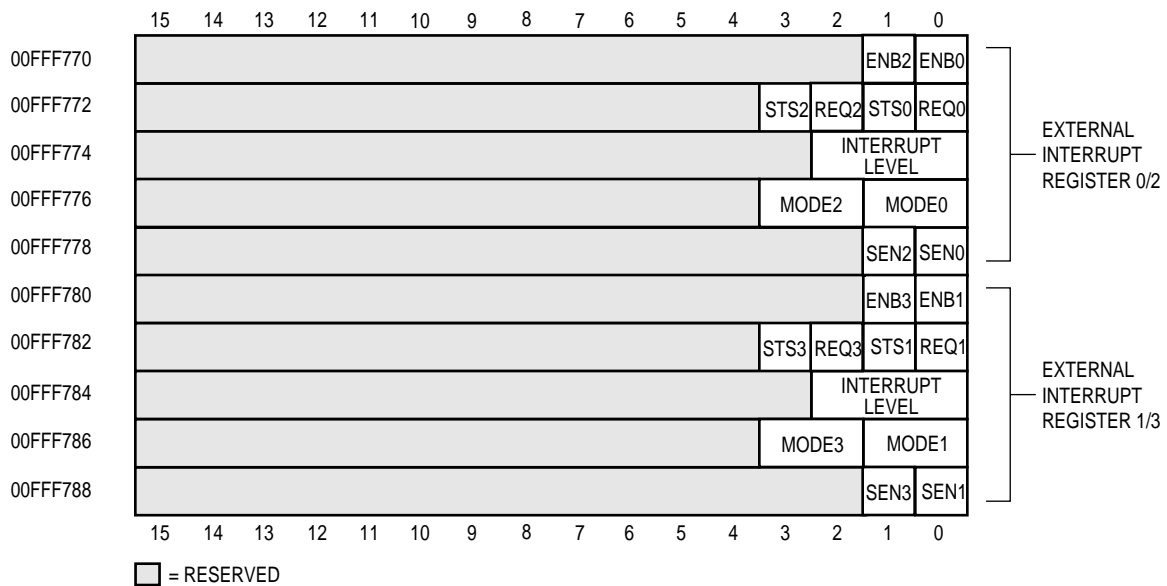


Figure 5-2. External Interrupt Registers (EXIR0/2–EXIR1/3)

**ENBx—External Interrupt Enable**

External interrupts are enabled when set and disabled when cleared. This bit affects the interrupt to the core, but not the interrupt event in the STSx or REQx fields.

**STSx—External Interrupt Status**

This bit reflects the status of the corresponding IRQ signal after it is synchronized with CLK1.

**REQx—External Interrupt Request**

This bit is set when an external interrupt event occurs. To remove the interrupt, the core should write a 1 to the REQx bit. If this bit and the ENBx bit are set, an interrupt is posted.

**Interrupt Level Register**

This field is used to set the interrupt level. A level 7 is the highest priority and level 1 is the lowest priority. Level 0 indicates that the interrupt is disabled.

**MODEx—External Interrupt Mode**

This field controls the external interrupt polarity and whether it is level sensitive or edge sensitive. Table 5-2 lists the MODEx field encodings.

**Table 5-2. External Interrupt Polarity**

ENCODING	DESCRIPTION
00	Active Low Level
01	Active High Level
10	Falling Edge Transition
11	Rising Edge Transition

**SENx—External Interrupt (Software Enable)**

This bit is only used when the external interrupt is level sensitive. SENx is only set by the software and only cleared by the hardware. It allows the software to control a level-sensitive interrupt without using the ENBx bit. The software cannot clear SENx, but it is set by the software and cleared by the hardware when a level-sensitive interrupt is received. This bit is also forced clear if the MODEx field is set for edge-sensitive interrupts (10<sub>2</sub> or 11<sub>2</sub>). When an external interrupt occurs, REQx is set and SENx is automatically cleared.

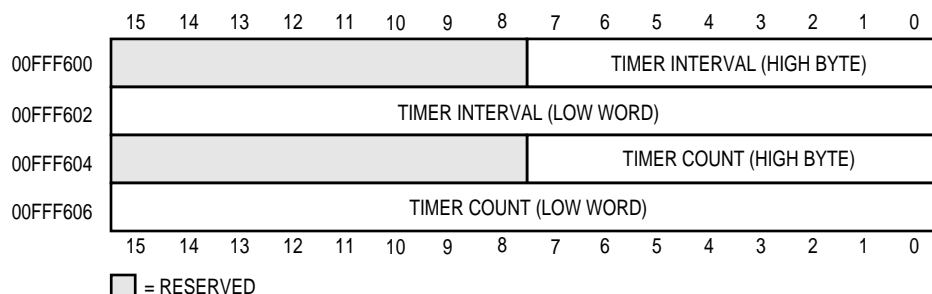
Because a level-sensitive interrupt may not change, caution must be taken so that the core is not constantly interrupted. It is recommended that an external interrupt be disabled before the SENx field in the external interrupt register is initialized by the software.



### 5.3 TIMER MODULE

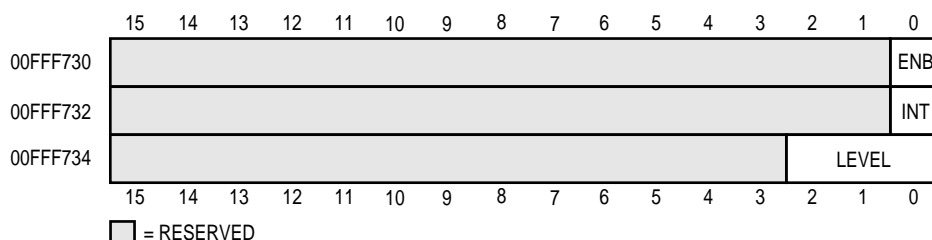
For the operating software to monitor time, the MC68322 provides a timer that continuously posts an interrupt event to the core at regular intervals. Like the other modules of the MC68322, the timer runs independent of the core (even though its interval can be changed by the core at any time).

A timer operation is initiated by a write to the interval field in the timer register. At 16MHz, the width of the timer register provides an interrupt frequency range from 125ns to more than one second. Typically, this register is only set once during software initialization to produce interrupts every 50ms. Figure 5-3 illustrates the timer register.



**Figure 5-3. Timer Register**

The timer interval field specifies the number of 1× clocks between timer interrupts. The timer count field provides the current 1× clock count value in case more precise timing is required. The interval field must be initialized before a timer interrupt event is enabled in the timer interrupt event register (TIER), which is illustrated in Figure 5-4.



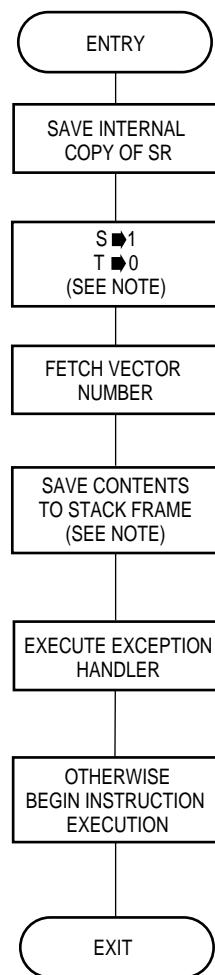
**Figure 5-4. Timer Interrupt Event Register**

The timer interval and count fields in the timer register consist of a high byte and low word. First the high byte should be loaded, followed by the low word, which is the natural order of an core long-word write access. Loading the low word causes the new interval to be loaded into the timer counter. The count field counts down to zero, generates an interrupt, and then reloads from the interval field. The value read from the count field reflects the current state of the timer counter. The count field should be read twice to determine the actual timer count. The timer count will decrement between the read of the high byte and the low word portions of the count field, which can cause the high byte to decrement. For example, if the timer counts down from  $300_{16}$  to  $2FF_{16}$  between reads, then the software may incorrectly read the upper byte as 3 and the lower word as  $FF_{16}$  for a total of  $3FF_{16}$ .

## 5.4 CORE EXCEPTION HANDLING

Exception processing is the activity performed by the core in preparation to execute a special routine for any condition that causes an exception. In particular, exception processing does not include the execution of the routine itself. It is the transition from the normal processing of a program to the processing required for any special internal or external condition that preempts normal processing. External conditions that cause exceptions are interrupts from external devices, address errors, and resets. Internal conditions that cause exceptions are instructions, address errors, and tracing. For example, the **trap**, **trapv**, **chk**, **rte**, and **div** instructions can generate exceptions as part of their normal execution. In addition, illegal instructions and privilege violations cause exceptions. Exception processing uses an exception vector table and an exception stack frame.

Exception processing occurs in four functional steps. However, all individual bus cycles associated with exception processing (vector acquisition and stacking) are not guaranteed to occur in the order in which they are described in this section. Figure 5-5 illustrates a general flowchart for the steps taken by the core during exception processing.

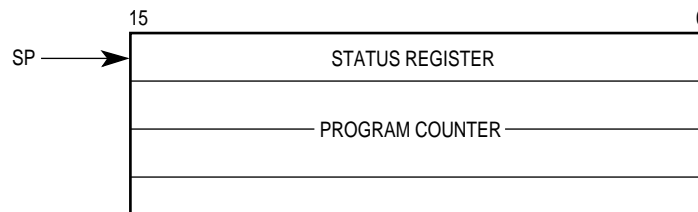


NOTE: These blocks vary for reset and interrupt exceptions.

**Figure 5-5. General Exception Processing Flowchart**

During the first step, the core makes an internal copy of the internal status register (SR), except during the reset exception, which does not make a copy of the internal status register. Then the core changes to the supervisor mode by setting the S bit and inhibits tracing of the exception routine by clearing the T bit in the SR. For the reset and interrupt exceptions, the core also updates the interrupt priority mask in the internal status register. During the second step, the core determines the vector number for the exception and internal logic provides the vector number for all exceptions. This vector number is used in the last step to calculate the address of the exception vector. Please note that throughout this section, vector numbers are given in decimal notation.

The third step is to save the current core contents for all exceptions other than the reset exception, which does not stack information. The core creates an exception stack frame on the active supervisor stack and fills it with information appropriate for the type of exception. Other information can also be stacked, depending on which exception is being processed and the state of the core prior to the exception. Figure 5-6 illustrates the general form of the exception stack frame.



**Figure 5-6. General Form of an Exception Stack Frame**

The last step initiates execution of the exception routine. The new internal program counter value is fetched from the exception vector. The core then resumes instruction execution. The instruction at the address in the exception vector is fetched and normal instruction decoding and execution is started. The memory map for exception vectors is listed in Table 5-3. The vector table is 512 words long (1,024 bytes), starting at address 0 (decimal), and proceeding through address 1,023 (decimal). The vector table provides 255 unique vectors, some of which are reserved for trap and other system function vectors. Of the 255, 192 are reserved for user interrupt vectors. However, the first 64 entries are not protected, so user interrupt vectors may overlap at the discretion of the systems designer.



**Note:** All interrupt exceptions on the MC68322 are autovectored. See **Section 5.4.2.2 Interrupt Exceptions** for more details.

**Table 5-3. Exception Vector Assignments**

VECTOR NUMBER(S)	VECTOR OFFSET (HEX)	SPACE <sup>4</sup>	ASSIGNMENT
0	000	SP	Reset Initial Interrupt Stack Pointer <sup>2</sup>
1	004	SP	Reset Initial Program Counter <sup>2</sup>
2	008	SD	Not Applicable
3	00C	SD	Address Error
4	010	SD	Illegal Instruction
5	014	SD	Integer Divide By Zero
6	018	SD	CHK Instruction
7	01C	SD	TRAPV Instruction
8	020	SD	Privilege Violation
9	024	SD	Trace
10	028	SD	Line 1010 Emulator (Unimplemented A-Line Opcode)
11	02C	SD	Line 1111 Emulator (Unimplemented F-Line Opcode)
12 <sup>1</sup>	030	—	(Unassigned, Reserved)
13 <sup>1</sup>	034	—	(Unassigned, Reserved)
14	038	SD	Format Error
15	03C	SD	Not Applicable
16-23 <sup>1</sup>	040-05C	—	(Unassigned, Reserved)
24	060	SD	Not Applicable
25	064	SD	Level 1 Interrupt Autovector
26	068	SD	Level 2 Interrupt Autovector
27	06C	SD	Level 3 Interrupt Autovector
28	070	SD	Level 4 Interrupt Autovector
29	074	SD	Level 5 Interrupt Autovector
30	078	SD	Level 6 Interrupt Autovector
31	07C	SD	Level 7 Interrupt Autovector
32-47	080-0BC	SD	TRAP #0-15 Instruction Vectors
48-63 <sup>1</sup>	0C0-0FC	—	(Unassigned, Reserved)
64-255	100-3FC	SD	User Defined Vectors

## NOTES:

1. Vector Numbers 12, 13, 16-23, And 48-63 Are Reserved For Future Enhancements By Motorola. No User-Peripheral Devices Should Be Assigned These Numbers.
2. Reset Vector (0) Requires Four Words (Unlike The Other Vectors, Which Only Require Two Words) And Is Located In The Supervisor Program Space.
3. TRAP #n Uses Vector Number 32+ n.
4. SP Denotes Supervisor Program Space And SD Denotes Supervisor Data Space.

## 5.4.1 Processing Specific Exceptions

The exceptions are classified according to their sources and each type is processed differently. There are three types of exceptions—reset, interrupt, and instruction traps.

**5.4.2.1 RESET EXCEPTION.** The reset exception corresponds to the highest exception level. The processing of the reset exception is performed for system initiation and recovery from catastrophic failure. Any processing in progress at the time of the reset is aborted and cannot be recovered. The core is forced into the supervisor state and the trace state is forced off. The interrupt priority mask is set at level 7. The vector number is internally generated to reference the reset exception vector at location 0 in the supervisor program space. Because no assumptions can be made about the validity of register contents, in particular the SSP, neither the internal program counter nor the internal status register are saved. The address in the first two words of the reset exception vector is fetched as the initial SSP, and the address in the last two words is fetched as the initial internal program counter. Finally, instruction execution is started at the address in the internal program counter. The initial internal program counter should point to the power-up/restart code.

The MC68322 does support the **reset** instruction, but the instruction will not affect any changes in the system and should be avoided due to the long execution time of the instruction. A reset exception is initiated by  $\overline{\text{RESET}}$ , not the **reset** instruction. The **reset** instruction does not assert the  $\overline{\text{RESET}}$  signal and does not modify any internal registers. The execution of the **reset** instruction does not affect the state or function of other on-chip modules.

**5.4.2.2 INTERRUPT EXCEPTIONS.** An interrupt event is posted to the core by the interrupt controller using the internal  $\overline{\text{IPL2}}\text{--}\overline{\text{IPL0}}$ . Interrupt events arriving at the core do not force immediate exception processing, but the requests are given a pending status. Pending interrupts are detected between instruction executions. If the priority of the pending interrupt is lower than or equal to the current core priority mask level, execution continues with the next instruction, and the interrupt exception processing is postponed until the current core priority mask level becomes less than the pending interrupt event.

If the priority of the pending interrupt is greater than the current core priority mask level, the exception processing sequence is started. A copy of the internal status register is saved, the privilege mode is set to supervisor mode, tracing is suppressed, and the core priority mask level is set to the level of the interrupt being acknowledged. The core internally generates a vector number corresponding to the interrupt level number. It then proceeds with the usual exception processing. The saved value of the internal program counter is the address of the instruction that would have been executed had the interrupt not been taken. The appropriate interrupt vector is fetched and loaded into the internal program counter and normal instruction execution commences in the interrupt handling routine.

**5.4.2.3 INSTRUCTION TRAPS.** Traps are exceptions caused by instructions and they occur when the core recognizes an abnormal condition during instruction execution or when an instruction is executed that normally traps during execution. Exception processing for traps is straightforward. The internal status register is copied, the supervisor mode is entered, and tracing is turned off. The vector number is internally generated, but as the **trap** instruction, part of the vector number comes from the instruction itself. The internal program counter and the copy of the internal status register are saved on the supervisor stack. The saved value of the internal program counter is the address of the instruction following the instruction that generated the trap. Finally, instruction execution commences at the address in the exception vector.

Some instructions are used specifically to generate traps. The **trap** instruction always forces an exception and is useful for implementing system calls for user programs. The **trapv** and **chk** instructions force an exception if the user program detects a run-time error, which can be an arithmetic overflow or a subscript out of bounds. A signed divide (**divs**) or unsigned divide (**divu**) instruction forces an exception if a division operation is attempted with a divisor of zero.

**5.4.2.4 ILLEGAL AND UNIMPLEMENTED INSTRUCTIONS.** An illegal instruction refers to any of the word bit patterns that do not match the bit pattern of the first word of a legal core instruction. If such an instruction is fetched, an illegal instruction exception occurs. Motorola reserves the right to define instructions using the opcodes of any of the illegal instructions. Three bit patterns always force an illegal instruction trap on all M68000 Family-compatible microprocessors. These patterns are:  $4AFA_{16}$ ,  $4AFB_{16}$ , and  $4AFC_{16}$ . Two of the patterns,  $4AFA_{16}$  and  $4AFB_{16}$ , are reserved for Motorola system products. The third pattern,  $4AFC_{16}$ , is reserved for customer use (as the take illegal instruction trap—**illegal**—instruction).

Word patterns with bits 15–12 equaling  $1010_2$  or  $1111_2$  are distinguished as unimplemented instructions and separate exception vectors are assigned to these patterns to permit efficient emulation. Opcodes beginning with bit patterns equaling  $1111_2$  (F line) are implemented in the MC68020 as coprocessor instructions. These separate vectors allow the operating system to emulate unimplemented instructions in the software. Exception processing for illegal instructions is similar to that for traps. After the instruction is fetched and decoding is attempted, the core determines that execution of an illegal instruction is being attempted and starts exception processing. The exception stack frame is then pushed on the supervisor stack and the illegal instruction vector is fetched.

**5.4.2.5 PRIVILEGE VIOLATIONS.** To provide system security, various instructions are privileged. An attempt to execute one of the privileged instructions while in the user mode causes an exception. The privileged instructions are as follows:

- **and immediate to sr**
- **move usp**
- **eor immediate to sr**
- **or immediate to sr**
- **move to sr**
- **reset**
- **move from sr**
- **rte**
- **movec**
- **stop**
- **moves**

Exception processing for privilege violations is nearly identical to that for illegal instructions. The core starts exception processing once the instruction is fetched, decoded, and the core determines that a privilege violation is being attempted. The internal status register is copied, the supervisor mode is entered, and tracing is turned off. The vector number is generated to reference the privilege violation vector and the current internal program counter and copy of the internal status register are saved on the supervisor stack. The saved value of the internal program counter is the address of the first word of the instruction causing the privilege violation. Finally, instruction execution commences at the address in the privilege violation exception vector.

**5.4.2.6 TRACING.** To aid in program development, the core includes a facility to allow tracing after each instruction. When tracing is enabled, an exception is forced after the execution of each instruction. Thus, a debugging program can monitor the execution of the program under test.

The trace facility is controlled by the T bit in the supervisor portion of the internal status register. If the T bit is cleared, tracing is disabled and instruction execution proceeds from instruction to instruction as normal. If the T bit is set (on) at the beginning of an instruction's execution, a trace exception is generated after the completion of the instruction. If the instruction is not executed because an interrupt is taken or because the instruction is illegal or privileged, the trace exception does not occur. The trace exception also does not occur if the instruction is aborted by a reset, bus error, or address error exception. If the instruction is executed and an interrupt is pending on completion, the trace exception is processed before the interrupt exception. During the execution of the instruction, if an exception is forced by that instruction, the exception processing for the instruction exception occurs before that of the trace exception.

As an extreme illustration of these rules, consider the arrival of an interrupt during the execution of a **trap** instruction while tracing is enabled. First, the trap exception is processed, then the trace exception, and finally the interrupt exception. After the execution of the instruction is complete and before the start of the next instruction, exception processing for a trace begins and a copy of the internal status register is made. The transition to supervisor mode is made and the T bit of the internal status register is turned off, thus disabling further tracing. The vector number is generated to reference the trace exception vector and the current internal program counter and the copy of the internal status register are saved on the supervisor stack. The saved value of the internal program counter is the address of the next instruction. Instruction execution commences at the address contained in the trace exception vector.

**5.4.2.7 ADDRESS ERROR.** An address error exception occurs when the core attempts to access a word or long-word operand or an instruction at an odd address. The bus cycle is aborted and the core ceases current processing and begins exception processing. Likewise, if an address error occurs during the exception processing for an address error (or reset) the core is halted. A common example of address error generation is when the stack pointer is pointing at an odd address.

## 5.4.2 Multiple Exceptions

When multiple exceptions occur simultaneously, they are processed according to a fixed priority. Table 5-4 lists the exceptions, grouped by characteristics, with group 0 having the highest priority. Within group 0, reset has the highest priority, followed by an address error. Within group 1, trace has priority over external interrupts, which takes priority over illegal instruction and privilege violation. Since only one instruction can be executed at a time, no priority relationship applies within group 2.

**Table 5-4. Exception Grouping and Priority**

GROUP	EXCEPTION	PROCESSING
0	Reset and Address Error	Exception Processing Begins Within Two Clock Cycles.
1	Trace, Interrupt, Illegal, and Privilege	Exception Processing Begins Before The Next Instruction.
2	<b>trap</b> , <b>trapv</b> , <b>chk</b> , and <b>div</b>	Exception Processing Is Started By Normal Instruction Execution.

The priority relationship between two exceptions determines which one is taken (or taken first) if the conditions for both arise simultaneously. In another example, if an interrupt event occurs during the execution of an instruction while the T bit in the internal status register is asserted, the trace exception has priority and is processed first. However, before instruction execution resumes, the interrupt exception is processed. As a general rule, the lower the priority of an exception, the sooner the handler routine for that exception executes. This rule does not apply to the reset exception. Its handler is executed first (even though it has the highest priority) because the reset operation clears all other exceptions.

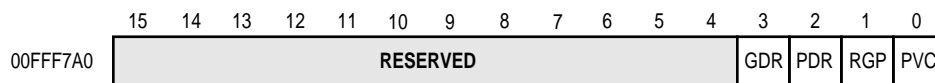


### 5.4.3 Exception Bus Cycles

Because the MC68322 has a self-contained core, all addresses generated during bus cycles should decode to an internal register, chip-select bank, or DRAM. In the event that an undefined memory location is accessed, a core illegal address interrupt event is posted and generates an interrupt if enabled. Therefore, an external BERR is not required. All vectors in the exception vector table should be initialized, including the address error exception. The core could have the stack pointer set to an odd byte boundary, which would result in an address error exception.

## 5.5 MODULE SOFT-RESET REGISTER

The module soft-reset register (MSRR) contains four write-only bit fields that are used to independently initialize a corresponding module to a known state. Figure 5-9 illustrates the MSRR.



**Figure 5-7. Module Soft-Reset Register**

The GDR and PDR bits, when set, reset the GDMA and PDMA respectively. These two bits provide for a recovery from an addressing error or any other error that causes the DMA channel to stop in an indeterminate condition. The RGP bit then returns the RGP to its idle state and the PVC bit returns the print engine video controller to its idle state. The PVC bit can be used in the event that the PVC has an address error or if an underrun condition occurs.

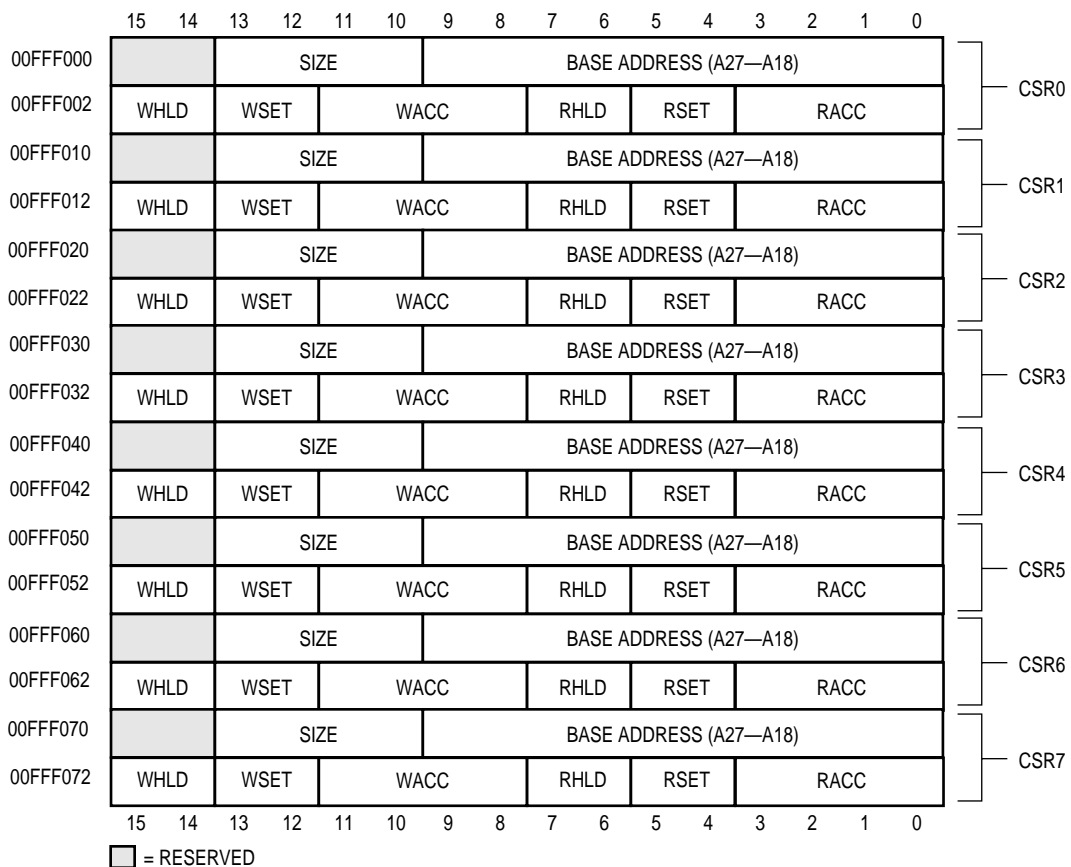
# SECTION 6

## SYSTEM INTEGRATION MODULE

The system integration module (SIM) provides the ROM, PROM, and peripheral chip-selects. It contains eight programmable chip-select banks to decode the address and supply internal  $\overline{DTACK}$  to the core after the appropriate number of wait states.

### 6.1 CHIP-SELECT REGISTERS AND BANKS

The MC68322 contains eight noncontiguous memory-mapped chip-select registers (CSR7–CSR0), each corresponding to a variably sized chip-select bank (CSB7–CSB0) in memory. Each chip-select register (CSR) indicates the corresponding chip-select bank's location, size, and timing for read and write accesses (such as data setup, hold, and recovery time). Only the core or an external bus master can access the CSRs as word or long-word sized registers. Figure 6-1 illustrates the eight chip-select registers.



**Figure 6-1. Chip-Select Register (CSR7–CSR0)**

The upper word of the memory-mapped CSR contains the bit fields that control timing characteristics for read and write cycles on the chip-select bank. By manipulating these fields, different timing are created for read and write cycles on a bank-by-bank basis.

WHLD—Write Hold Time

$\overline{CSx}$  hold time from trailing edge of  $\overline{WRU}$  and  $\overline{WRL}$  for write operations.

WSET—Write Setup Time

$\overline{CSx}$  setup time before leading edge of  $\overline{WRU}$  and  $\overline{WRL}$  for write operations.

WACC—Write Access Time

Indicates the duration of the  $\overline{WRU}$  and  $\overline{WRL}$  signals during write operations.

RHLD—Read Hold Time

$\overline{CSx}$  hold time from falling edge of  $\overline{RD}$  for read operations.

RSET—Read Setup Time

$\overline{CSx}$  setup time before leading edge of  $\overline{RD}$  for read operations.

RACC—Read Access Time

Indicates the duration of the  $\overline{RD}$  signal during read operations.

The chip-select active time (in CLK1s) for read operations is derived from these fields through the calculation:

$$\text{active read time} = \text{RSET} + \text{RACC} + \text{RHLD} + 1\frac{1}{2}$$

The chip-select active time (in CLK1s) for write operations is derived from the calculation:

$$\text{active write time} = \text{WSET} + \text{WACC} + \text{WHLD} + 1\frac{1}{2}$$

There are two guidelines that the software must follow when programming these timing parameters:

- The minimum allowed value for active read time and active write time is  $2\frac{1}{2}$ .
- The minimum allowed value for WSET = 1, which produces a write setup time of 2.

The CSR's size field contains the chip-select bank's size and should have a defined programmed value. Because each CSR has its own size field, each chip-select bank can be a different size or completely removed from the memory map. This bit field allows a maximum bank size of 64M. Table 6-1 lists the encodings for the size field.

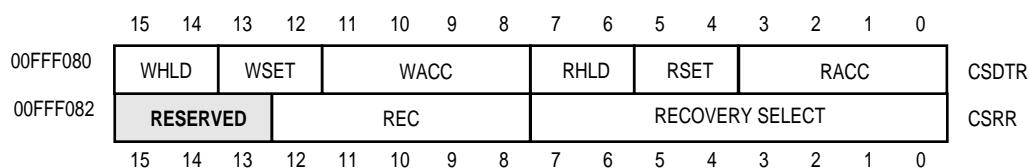
**Table 6-1. Size Field Encoding**

ENCODING	BANK SIZE	ENCODING	BANK SIZE
0	Disabled	5	4M
1	256K	6	8M
2	512K	7	16M
3	1M	8	32M
4	2M	9	64M

The CSR's base address field contains the chip-select bank's address and allows each bank to be programmed anywhere in the 256M range of the memory map. The bits in the base address field correspond to bits 27–18 of the chip-select bank's starting address.

At reset, CSR7–CSR1 are disabled. The base address of CSR0 is set to zero so the core can fetch the reset vector. CSR0 should be connected to ROMs containing the startup code and is enabled with a size of 8M on power-up. The programmable parameters should provide setup and hold times for data to the core. The read data is latched on the rising edge of the  $\overline{RD}$  signal for any cycle with RHLD different than zero. If RHLD is equal to zero, the read data must be set up by the rising edge of the  $\overline{CSx}$  signal that negates half of CLK1 before the  $\overline{RD}$  signal. On power-up, CSR0's WSET and RSET fields are also set to zero and the write and read ACC and HLD fields will be at their maximum. Be aware that the software must program the chip-select parameters for banks 7–1 before using them.

A chip-select bank can be individually located anywhere in the 256M range of the memory map and can overlap with DRAM or other chip-select banks. In case of an address overlap, all memory-mapped registers have priority over chip-select banks. Likewise, chip-select banks have priority over DRAM banks and lower numbered chip-select banks have priority over higher numbered chip-select banks. For example, CSB0 has a higher priority than CSB7. There are two additional timing registers accessed by the core—chip-select DMA timing register and chip-select recovery register. Figure 6-2 illustrates these two registers.

**Figure 6-2. Chip-Select DMA Timing and Recovery Registers**

The chip-select DMA timing register (CSDTR) is a dedicated register that provides access timing parameters to the chip-select bank accessed by the general-purpose DMA (GDMA) during DMA transfers. These timing fields function in a similar manner as those in the upper word portion of a CSR. However, the normal timing parameters for the bank are not used when the bank is accessed by the GDMA. See **Section 8 DMA Interface** for more details.

The chip-select recovery register (CSRR) contains a REC field that controls the chip-select recovery time. This recovery time applies to a chip-select only if it has been enabled in the recovery select field. The REC field provides a common programmable recovery value for all chip-select banks. Since the recovery time is common for all the chip-select banks, it should be programmed to satisfy the maximum recovery value for all chip-select banks. If the recovery time has not been satisfied between successive access to a bank (with recovery enabled) the second access is delayed.

Each bit of the recovery select field corresponds to a chip-select bank. If recovery is enabled, it is forced for that bank, regardless of when it is accessed by GDMA, the core, or an external bus master.

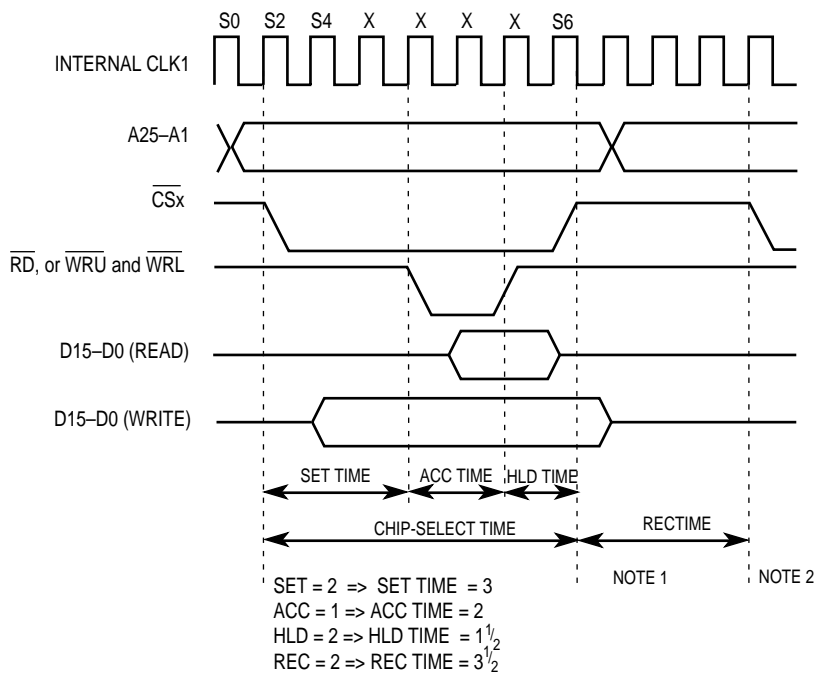
## 6.2 SYNCHRONOUS AND ASYNCHRONOUS CHIP-SELECT ACCESS TIMING

In synchronous mode, once the programmed timing values are satisfied, the system integration module generates an internal  $\overline{DTACK}$  to the core and terminates the MC68322 bus cycle. Table 6-2 lists the core's synchronous access timing values for the chip-selects and Figure 6-3 illustrates a synchronous read or write access.

**Table 6-2. Synchronous Timing Values**

PARAMETER NAME	CSR FIELD	FORMULA (CLK1)	MIN. VALUES (CLK1)	MAX. VALUE (CLK1)
Read Setup	RSET	RSET Time = RSET + 1	1	4
Write Setup	WSET	WSET Time = WSET + 1	2	4
Read Access	RACC	RACC Time = RACC + 1	1	16
Write Access	WACC	WACC Time = WACC + 1	1	16
Read Hold	RHLD	RHLD Time = RHLD - $\frac{1}{2}$	- $\frac{1}{2}$	$2\frac{1}{2}$
Write Hold	WHLD	WHLD Time = WHLD - $\frac{1}{2}$	- $\frac{1}{2}$	$2\frac{1}{2}$
Recovery	REC	REC Time = REC + $\frac{1}{2}$	$\frac{1}{2}$	$32\frac{1}{2}$

NOTE: The Value WSET = 0 Produces A WSET Time = 2.

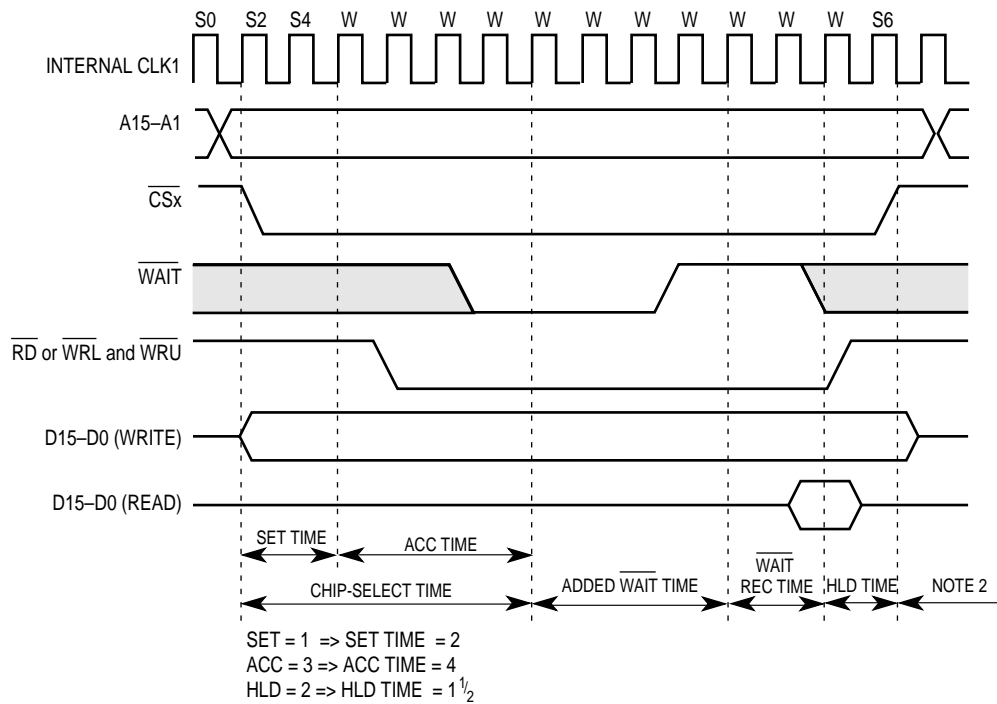


NOTES:

1. Recovery starts here; no access to  $\overline{CSx}$  will start until recovery is satisfied.
2. Delayed EC000 core access starts here.
3.  $\overline{WAIT}$  is inactive.

**Figure 6-3. Synchronous Read or Write Timing Diagram**

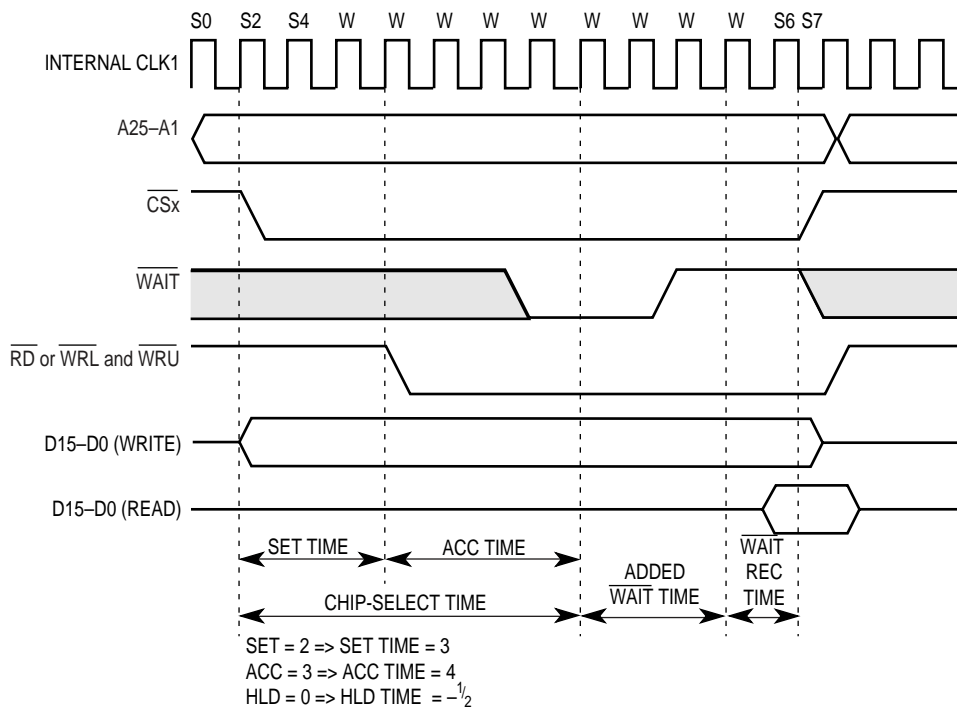
In asynchronous mode, read and write access timings (WACC and RACC) must be programmed to be at least two, so that  $\overline{WAIT}$  will be properly recognized by the MC68322. This value results in read and write access times of three CLK1s, which is the minimum required for a proper asynchronous access. Once RACC and WACC are satisfied, the MC68322 samples  $\overline{WAIT}$ . If  $\overline{WAIT}$  is active at the end of the access time, the access is stalled. After  $\overline{WAIT}$  becomes inactive and a two CLK1 recovery from  $\overline{WAIT}$ , RHLD and WHLD are satisfied. If  $\overline{WAIT}$  becomes inactive and RHLD and WHLD are zero, an internal  $\overline{DTACK}$  is generated to the core and the cycle is terminated. Figures 6-4 and 6-5 illustrate normal and special asynchronous read or write timing.



NOTE:

1. There is one CLK1 of synchronization delay on  $\overline{\text{WAIT}}$ .
2. Chip-select bank recovery starts.

**Figure 6-4. Asynchronous Read or Write Timing Diagram**



NOTE: There is one CLK1 of synchronization delay on  $\overline{\text{WAIT}}$ .

**Figure 6-5. Special Asynchronous Read or Write Timing Diagram**

# SECTION 7

## DRAM CONTROLLER

The MC68322 supports fast-page mode DRAM devices. However, nibble mode and static column DRAM devices are not supported. The MC68322 directly supports up to six banks of DRAM with bank sizes of 256Kbits × 16, 1Mbit × 16, and 4Mbits × 16. All DRAM sizes are a fixed data width of one word of 16 bits. Each of the six banks can support 512K to 8M.

The RISC graphics processor (RGP), printer video controller (PVC), DMAs, and EC000 core can all make accesses through the DRAM controller. The RGP and PVC use burst cycle accesses to maximize DRAM bus bandwidth, while the core does not burst to or from DRAM. All burst cycles from these modules occur within 256-word DRAM page boundaries. Core write accesses to the DRAM controller use data pipelining that allows for a reduction of internal bus arbitration delays.

### 7.1 DRAM REGISTERS AND BANKS

#### 7.1.1 Base Address and Size Fields

Each of the six DRAM banks correspond to an internal DRAM register. These six DRAM registers (DRAM5–DRAM0) are word-sized and indicate the DRAM bank’s size and location. Figure 7-1 illustrates these registers.

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
00FFF100					ROM MODE	SIZE	BASE ADDRESS (A27–A19)										DRAM0
00FFF110					ROM MODE	SIZE	BASE ADDRESS (A27–A19)										DRAM1
00FFF120					ROM MODE	SIZE	BASE ADDRESS (A27–A19)										DRAM2
00FFF130					ROM MODE	SIZE	BASE ADDRESS (A27–A19)										DRAM3
00FFF140					ROM MODE	SIZE	BASE ADDRESS (A27–A19)										DRAM4
00FFF150					ROM MODE	SIZE	BASE ADDRESS (A27–A19)										DRAM5
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0	
	□ = RESERVED																

**Figure 7-1. DRAM Register (DRAM5–DRAM0)**



Because each DRAM register contains a size field, each DRAM bank can be individually programmed for a different size or disabled entirely. The available sizes are either 512K, 2M, or 8M, depending on the size field's encoding. There are no limitations in the ordering of bank sizes. Table 7-1 lists the size field encodings and the equivalent DRAM bank sizes.

**Table 7-1. DRAM Size Options**

ENCODING	DRAM BANK ORGANIZATION
00	Disabled (No Size)
01	256 Kbit × 16 (512K)
10	1 Mbit × 16 (2M)
11	4 Mbit × 16 (8M)

The base address field, contained in each DRAM register, allows the six DRAM banks to be individually located at any location in the 256-byte memory map. The DRAM register base address field contains bits 27–19 of the corresponding DRAM bank's starting address. These DRAM banks can be contiguous to or disjointed from each other, as required by the operating environment. Even though the DRAM bank can be any size, the starting address must be unique, non-overlapping, and located on an address boundary equal to its size. For example, an 8M DRAM bank must be on an 8M address boundary. DRAM address space can, however, overlap with other registers, ROM, or I/O space. In case of an overlap, DRAM has the lowest priority.

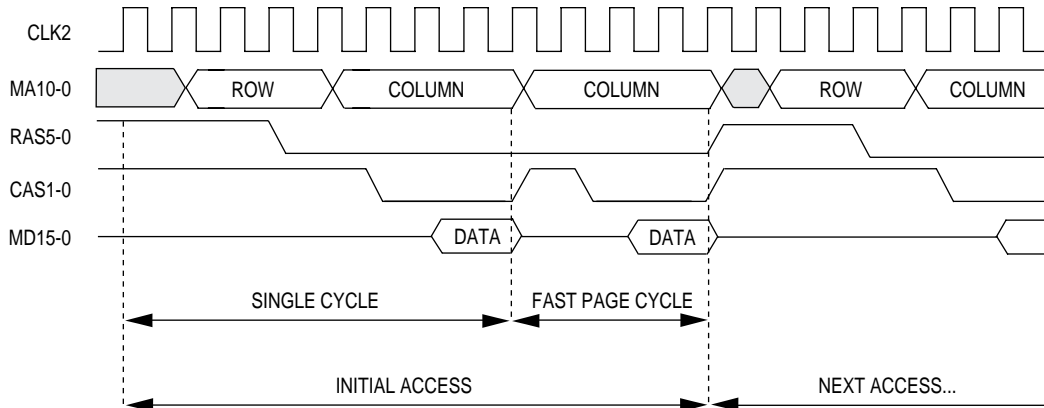
### 7.1.2 ROM Mode

The MC68322 DRAM controller "ROM mode" is available in the G59B Mask Set. The ROM mode of operation causes the selected DRAM channel to run with extended cycle times while the remainder of the channels operate at full speed. This will place font ROMs on one of the DRAM channels with only an external latch required to demultiplex the address signals. By placing the font ROMs on one of the DRAM channels, the MC68322 RGP will have direct access to font data, which eliminates the need for a font cache, thus reducing overall system DRAM requirements. ROM mode is selected for a particular DRAM channel by setting the ROM mode bit in the corresponding DRAM register (see Figure 7-1 for details).

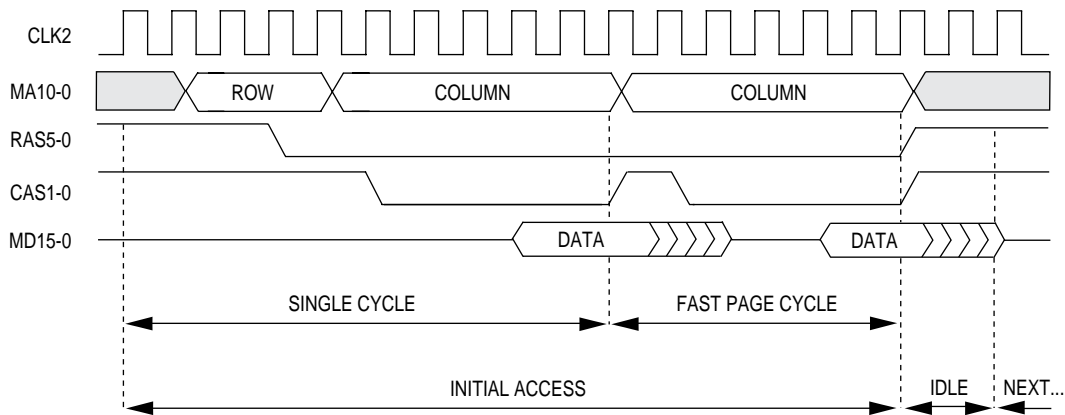
**7.1.2.1 FUNCTIONAL DESCRIPTION.** When the ROM mode is selected for one of the DRAM channels, the accesses to that channel are extended. Each initial and burst access is extended by one CLK1 period (two CLK2 periods). For example, in timing mode 1 the normal DRAM access time in CLK1s would be 4:2:2:2...; but when the ROM mode is selected, the access time becomes 5:3:3:3... for that particular channel. This timing relaxation in the ROM mode allows ROMs (which typically have longer access times than DRAM) to operate effectively on one (or more) of the DRAM channels.

At the same time, the other channels in the normal mode will run at full speed, thereby achieving maximum use of the available bandwidth in the DRAMs and maximizing system performance. In addition to extending the accesses, one CLK1 of idle time is inserted at the end of the entire access to the channel when in the ROM mode. This idle clock allows the data bus to return to a state of high impedance because ROMs can drive the data bus for an extended period of time after an access.

**7.1.2.2 TIMING EXAMPLE.** Figures 7-2 and 7-3 illustrate the difference between the DRAM accesses when in normal mode and ROM mode. This example is for DRAM timing mode 1, but similar behavior is exhibited in timing modes 0 and 2.



**Figure 7-2. DRAM Timing Mode 1 (Read Cycle, ROM Mode = 0)**

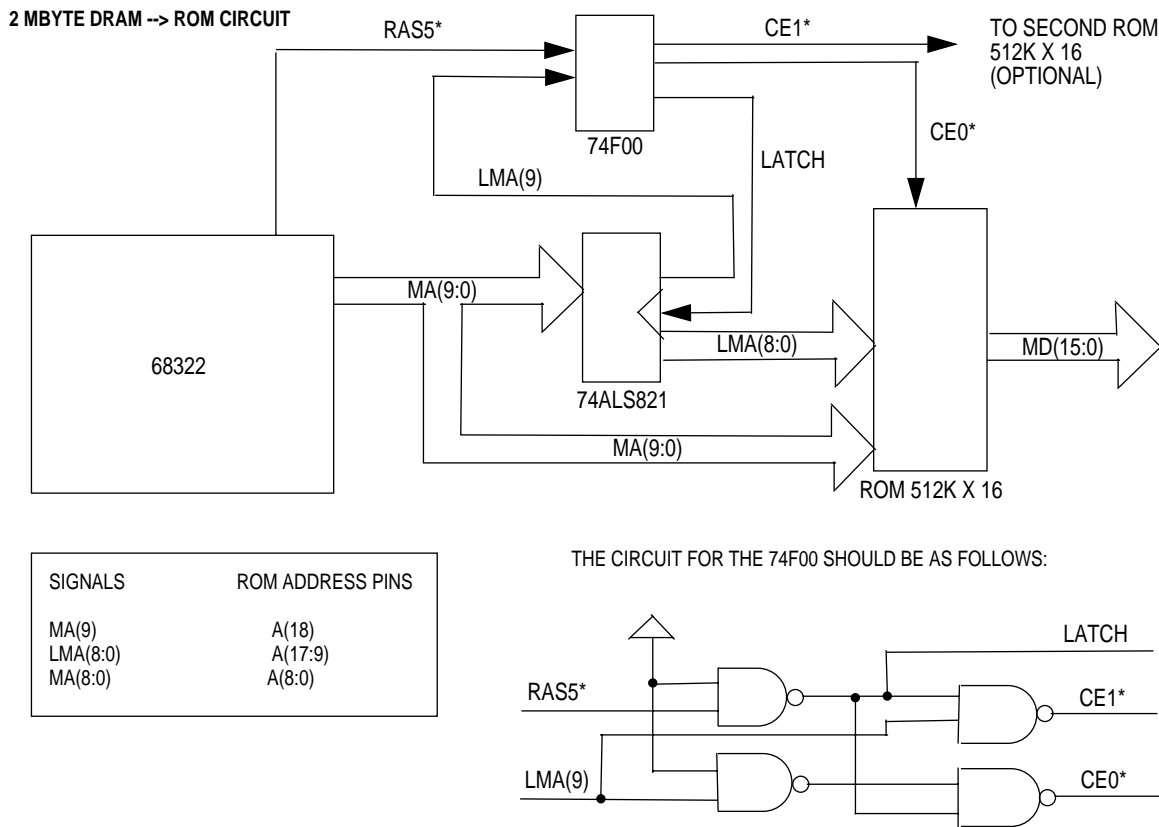


**Figure 7-3. DRAM Timing Mode 1 (Read Cycle, ROM Mode = 1)**



**Note:** The IDLE state shown here will also occur at the end of a refresh cycle whenever the ROM mode is selected for at least one of the DRAM channels.

**7.1.2.3 ADDRESS DEMULTIPLEXING CIRCUIT.** The MC68322 multiplexes the addresses on the DRAM bus as required by standard DRAM devices. To connect ROM to one of the DRAM channels, a simple address demultiplexing circuit must be used. The following circuit provides an example of how to implement this logic in a system.

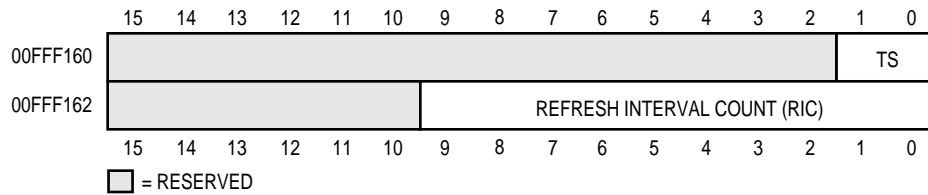


**Figure 7-4. Address Demultiplexing Example**

**7.1.2.4 OPERATIONAL EXAMPLE.** A typical MC68322-based system might be a 16-MHz design that uses 70ns DRAMs. To achieve maximum system performance in this case, select DRAM timing mode 2. This system might also include 120ns font ROMs that could be attached to one of the DRAM channels using a circuit such as the one illustrated above. Such an arrangement would minimize system cost by reducing the overall DRAM requirements. If the ROM mode was not used, the system would need to be slowed to timing mode 0 to accommodate these font ROMs. This slowdown would still produce a functional system, but overall performance would be hindered because this setup would not fully utilize the available bandwidth in the DRAMs. Instead, timing mode 2 should be selected for this system and the ROM mode should be activated for the DRAM channel supporting the font ROMs. These actions will optimize system performance as well as support the DRAM-based font ROMs that help reduce system costs.

## 7.2 DRAM CONTROL REGISTER

The DRAM control register (DRMCR) contains two fields—DRAM timing select (TS) and DRAM refresh interval (RIC). Figure 7-5 illustrates this register.



**Figure 7-5. DRAM Control Register**

The TS field is used to select one of three DRAM timing modes. The timing mode selected applies to all banks. All DRAM devices connected to the MC68322 must operate at the same speed because the DRAM controller does not support independently programmable bank speeds. The exception to this is the ability to slow down individual DRAM banks by setting the ROM mode bit in the corresponding DRAM register. See **Section 7.1.2 ROM Mode** for details.

The RIC field provides the refresh time interval in CLK1s. The  $\overline{\text{CAS}}$  and  $\overline{\text{RAS}}$  timing parameters for refresh are hardcoded and cannot be programmed. However, the RIC field can be changed at any time and the new value will take effect when the next terminal count is reached and a refresh cycle begins. Then the new RIC field value is loaded into the refresh counter.

## 7.3 DRAM TIMING MODES

Three DRAM timing modes provide wait state profiles optimized for various DRAM speeds at system frequencies of 16, 20, and 25MHz. These timing modes are not fixed to any specific system clock frequency and can therefore be used regardless of system clock frequency as long as the DRAM device timing is satisfied. The DRAM controller automatically bursts for both read and write bus cycles. Table 7-2 lists the timing modes, their associated DRAM control register TS field encodings, and the recommended system speeds.

**Table 7-2. DRAM Timing Modes**

MODE/WAIT STATE PROFILE	TS FIELD ENCODING	RECOMMENDED SYSTEM SPEED
Timing Mode 0 5:3:3:3...reads 5:2:2:2...writes	00	25MHz with 80ns
Timing Mode 1 4:2:2:2...reads 4:2:2:2...writes	01	20MHz with 100ns or 25MHz with 70ns
Timing Mode 2 3:2:1:2...reads 3:2:2:2...writes	10	16MHz with 80ns, 20MHz with 60ns, or 25MHz with 50ns
Not Used	11	Not Used

## 7.4 DRAM ACCESSES

When the RGP, PVC, one of the DMAs, or core accesses DRAM, the MC68322 begins a DRAM read or write cycle to one of the DRAM banks. The software must assure that eight or more refresh cycles (described below) have occurred before any access to the DRAM is performed. This is a requirement of the DRAM devices rather than the MC68322.

### 7.4.1 DRAM Refresh Cycles

DRAM refresh is accomplished through the use of  $\overline{\text{CAS}}$  before  $\overline{\text{RAS}}$  refresh cycles with the refresh interval fully programmable. In  $\overline{\text{CAS}}$  before  $\overline{\text{RAS}}$  refresh cycles, the DRAM provides its own refresh address, thus eliminating the need to generate an external refresh address. The refresh cycle timing is fixed and satisfies the timing requirements for the highest system clock speeds. During the refresh cycle,  $\overline{\text{WE}}$  is always driven high to prevent enabling of the test mode feature found in some DRAM devices.

All refresh cycles are transparent in that they take place between DRAM accesses without forcing the microprocessor to halt execution. If the core accesses a chip-select bank or chip-select register, a DRAM refresh cycle can occur simultaneously. However, if the core accesses DRAM during a refresh operation, it is delayed until the refresh cycle completes. When the refresh cycle occurs, the RGP and PVC will also be delayed. Figure 7-6 illustrates a DRAM refresh cycle.

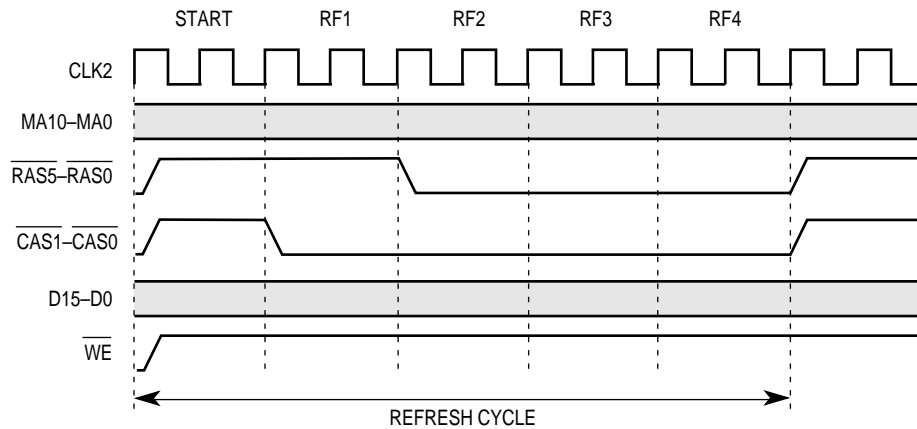


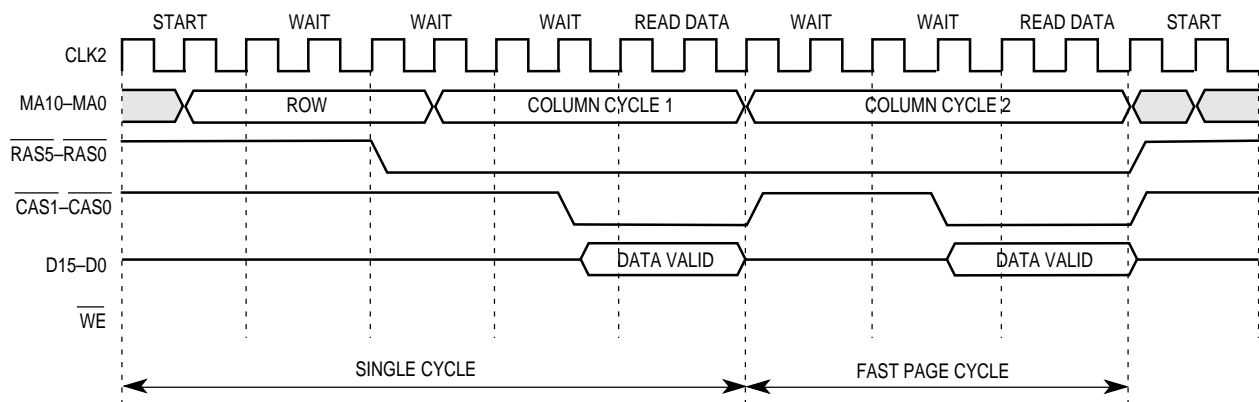
Figure 7-6. DRAM Refresh Cycle

### 7.4.2 DRAM Read Cycles

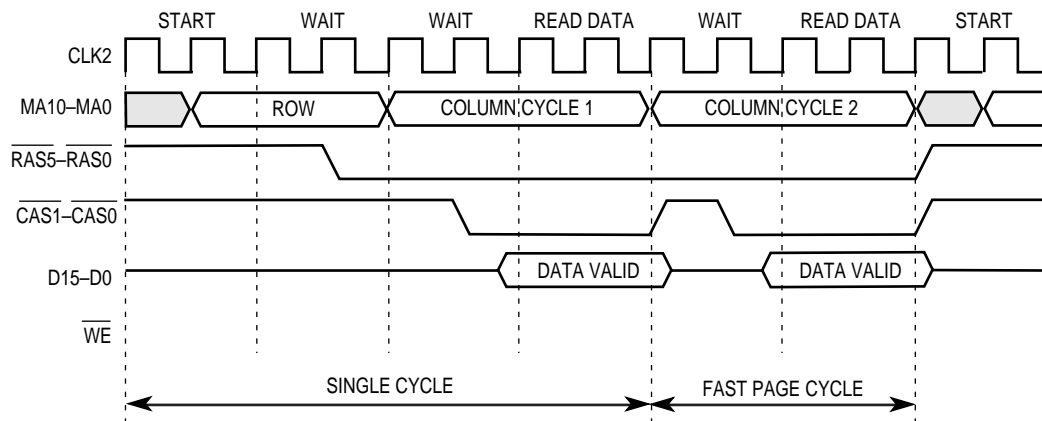
The timing mode selected during setup determines the sequence of events that join to form a DRAM read cycle.  $\overline{WE}$  remains negated during these events. The events in timing order are:

- The row address is placed on the memory address bus (MA10–MA0).
- The address selects one of the  $\overline{RAS}$  signals, which is then asserted.
- The column address is placed on the memory address bus (MA10–MA0).
- The  $\overline{CAS1}$  and  $\overline{CAS0}$  are asserted.
- Data is read.
- $\overline{RASx}$ ,  $\overline{CAS1}$ , and  $\overline{CAS0}$  are negated.

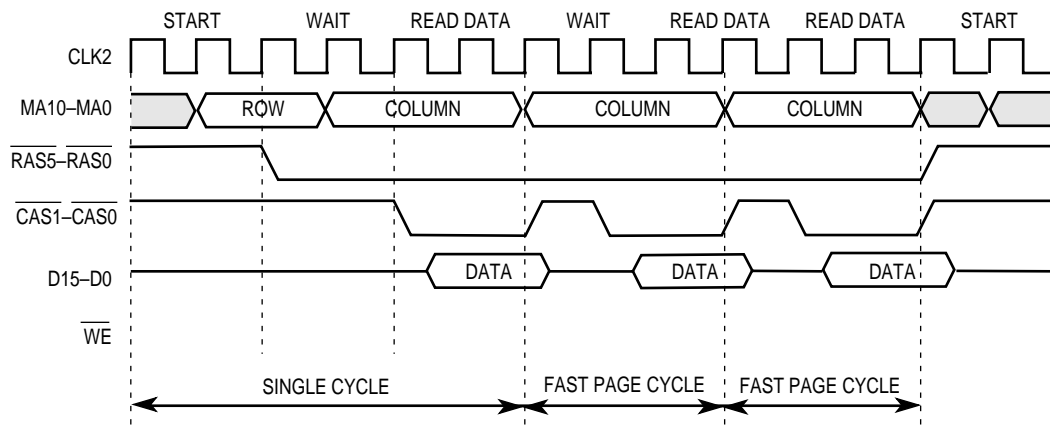
Figures 7-7 to 7-9 illustrate the DRAM read cycle timings for each of the three timing modes.



**Figure 7-7. DRAM Timing Mode 0 (Read Cycle, ROM Mode = 0)**



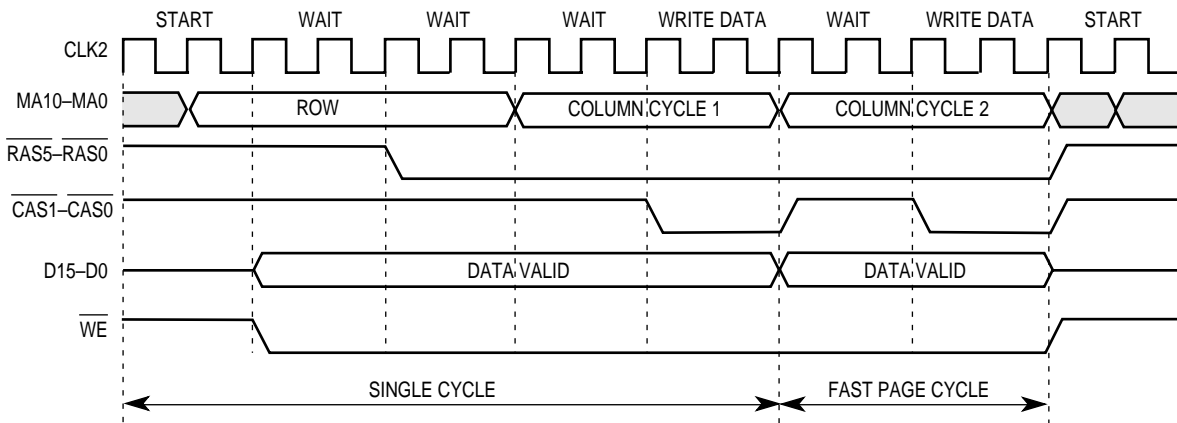
**Figure 7-8. DRAM Timing Mode 1 (Read Cycle, ROM Mode = 0)**



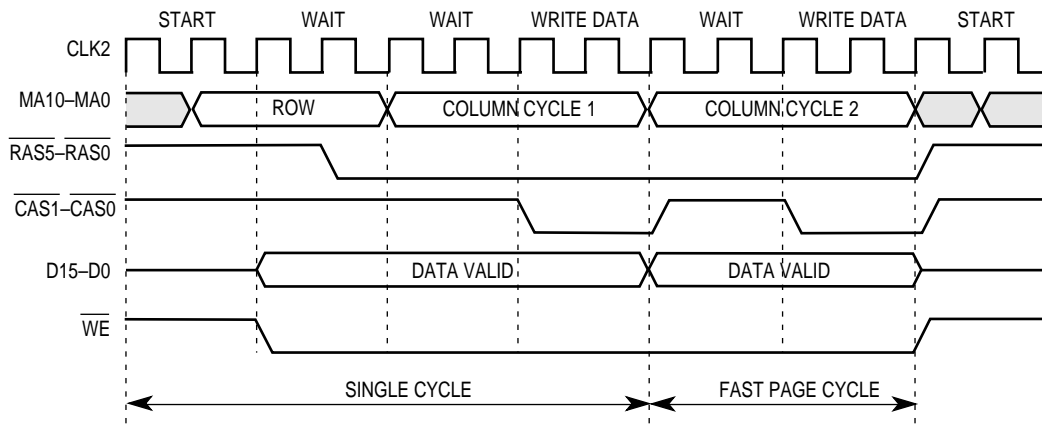
**Figure 7-9. DRAM Timing Mode 2 (Read Cycle, ROM Mode = 0)**

### 7.4.3 DRAM Write Cycles

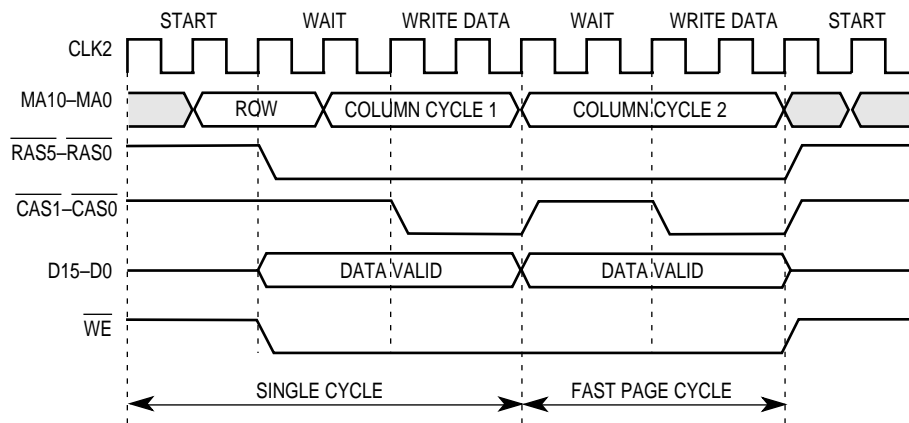
A DRAM write cycle begins in the same manner as a DRAM read cycle, except that data to be written is placed on the DRAM data bus (MD15–MD0) and  $\overline{WE}$  is asserted at the beginning of the cycle.  $\overline{CAS1}$  and  $\overline{CAS0}$  are asserted according to the internal  $\overline{LDS}$  and  $\overline{UDS}$  signals. The cycle ends by negating the  $\overline{RASx}$ ,  $\overline{CAS1}$ ,  $\overline{CAS2}$ , and  $\overline{WE}$ . It is important to note that the MC68322 implements early write cycles, which means that the assertion of  $\overline{CAS1}$  and  $\overline{CAS2}$  strobes data into DRAM. Figures 7-10 to 7-12 illustrate the DRAM write cycle timings for each of the three timing modes.



**Figure 7-10. DRAM Timing Mode 0 (Write Cycle)**



**Figure 7-11. DRAM Timing Mode 1 (Write Cycle)**



**Figure 7-12. DRAM Timing Mode 2 (Write Cycle)**

When a DRAM write cycle is initiated by the core, the data is latched into a buffer and an early internal DTACK is generated to the core. Subsequent write cycles to DRAM are delayed if the write buffer is full. The write buffer allows the core to execute the next instruction(s) while waiting for the word to be written to DRAM, thus increasing the MC68322's performance.

#### 7.4.4 DRAM Bus Arbitration

Arbitration for the DRAM bus is performed with strict priorities. If the bus is busy, the core will be delayed a maximum of 13 internal accesses before it has access to DRAM.



### 7.4.5 DRAM Burst Accesses

Burst accesses are carried out by using the fast-page mode operation of the DRAM. Fast-page mode is used for all multi-word burst cycles. In fast-page mode, the cycle is carried out as described for read and write cycles, except that only  $\overline{\text{CAS1}}$  and  $\overline{\text{CAS2}}$  are negated while  $\overline{\text{RASx}}$  remains asserted. After the defined interval (defined by the memory access timing register) the column address is incremented and the  $\overline{\text{CAS}}$  signals reassert, initiating another memory cycle. The DRAM controller interface will not burst across a 256-word address page boundary. If a burst access tries to cross a 256-word page, the cycle will terminate and a new one will begin.

## 7.5 POWER-UP SEQUENCE

Once  $\overline{\text{RESET}}$  is negated, the DRAM controller automatically performs DRAM refresh cycles. To guarantee proper DRAM operation after a power-up sequence or extended low frequency or static operation, some DRAMs require an 8-cycle precharge. To meet this requirement, the system should set the DRAM controller to the correct values and then perform eight reads from DRAM, thus disregarding the data.

On reset, the DRAM registers and the DRMCR's TS field are set to zero. Setting the TS field to zero selects the most conservative timing mode. All DRAM banks are disabled and the DRMCR refresh interval count (RIC) is set to a minimum value for the most frequent refresh rate. Software must load the RIC field with the proper value during initialization, but before it enables any DRAM bank, the DRAM register for a bank must be programmed with a valid starting address.

## **SECTION 8**

### **DMA INTERFACE**

The MC68322 DMA interface provides support for high-speed data transfers between external sources and DRAM. The DMA interface contains two channels—the parallel port DMA (PDMA) and the general-purpose DMA (GDMA). Both of these channels are single-ended and operate independently from each other with the single restriction that DRAM can only be accessed one channel at a time. Each DMA channel contains control state machines, a 16-bit data latch, data steering multiplexers, and counters for addresses and transferred data.

Each channel is interfaced and activated through a corresponding register. Once they are activated, the data steering multiplexers are configured based on direction and data width and the source controller begins requesting data. When source data is available, the source control state machine latches the data and signals the destination control state machine to start the destination operation. Destination transfer requests begin immediately after the first source data is received. DMA transfers continue until all the data has been transferred (the destination counter decrements to zero), a flush request is posted, or an error occurs.

The GDMA supports bidirectional byte- and word-sized data transfers between external DMA devices (such as I/O) and DRAM. The GDMA transfers to or from a single address on the MC68322 bus while automatically incrementing the address on the memory address bus (MA10–MA0). This provides a high-speed method of transferring data to or from a peripheral or DRAM. The PDMA operates like the GDMA, except that all transfers occur between the parallel port interface and DRAM. In addition, it is designed to support only byte-sized data transfers.

## 8.1 DMA CONFIGURATION REGISTERS

The DMA interface provides two internal memory-mapped configuration register sets called PDMA and GDMA. These registers configure each DMA channel and provide programmability for transfer address and count. Figure 8-1 illustrates the DMA interface registers.

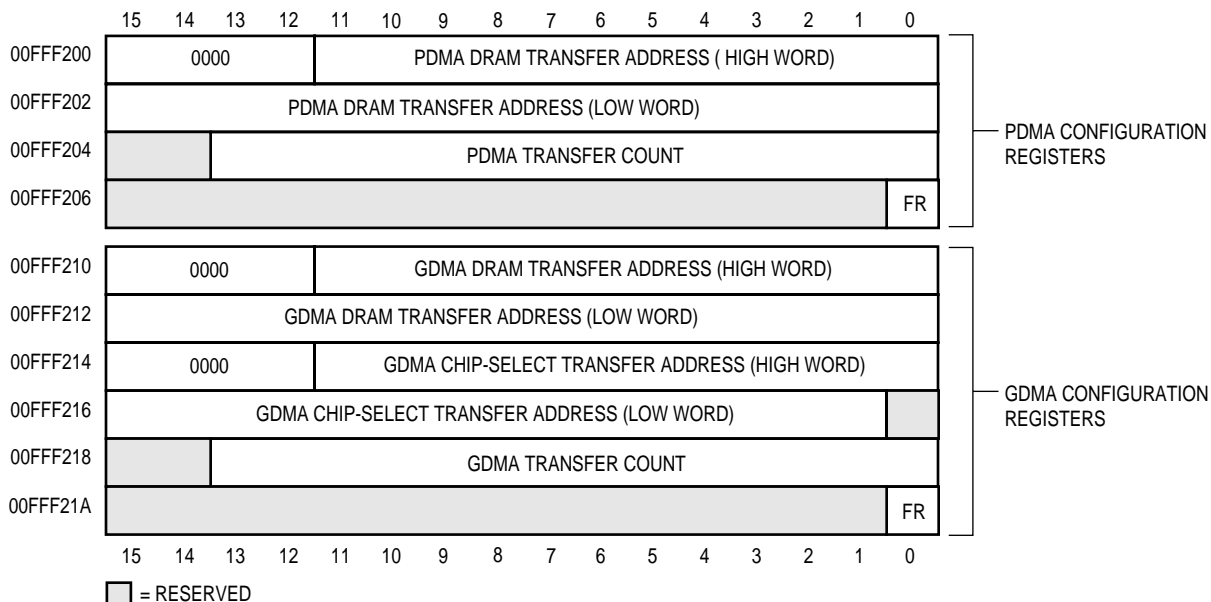


Figure 8-1. PDMA and GDMA Configuration Registers

### 8.1.1 Transfer Address Fields

The PDMA configuration register contains the PDMA DRAM transfer address field. The GDMA configuration register contains two transfer address fields—GDMA DRAM transfer address (GDTA) and GDMA chip-select transfer address (GCSTA). These fields define the base address for the beginning of the transfer and they cannot be written while the BSY bit in the active channel's interrupt event register is set. The transfer address fields support the entire 256M address range of the MC68322.

The GDTA field increments during a DMA transfer. For word-sized data transfers using the GDMA channel, the transfer address is incremented after every word is transferred. When receiving byte-sized data, the data is packed into words prior to accessing DRAM and the address is incremented after every full word is transferred to DRAM. Likewise, for sending byte-sized data, a word access to DRAM is made, the address is incremented, and the data is sent unpacked as bytes to the MC68322 bus. All byte-sized data packing is organized as big endian data.

The MC68322 bus address used for the GDMA channel is fixed and not incremented. The transfer address for the MC68322 bus is used to access one of eight possible chip-select banks. The GDMA chip-select transfer address appears on the output address pins of the MC68322 bus interface (A25–A1). It is assumed the device that is connected to the banks on the MC68322 bus used for the DMA handles any addressing issues internally.

## 8.1.2 Transfer Count Fields

Each DMA configuration register provides a 14-bit programmable transfer count field, thus allowing for a maximum 16-Kbyte or 16-Kword transfers. The transfer count is in words if the selected DMA channel is programmed to perform word-sized transfers. Similarly, the transfer count is in bytes if the DMA channel is programmed to perform byte-sized transfers. Writing this register activates the DMA channel, so that all other configuration register fields must be initialized before writing the transfer count field.

When activated, the transfer count value is loaded into an internal counter and decremented after each destination transfer. The transfer count field is not double buffered, so writing a new value during an active transfer will not start the next DMA operation at the conclusion of the current operation. However, if a new value is written to the transfer count field during an active transfer, the new value is ignored.

During an active transfer, reading the transfer count field will reflect the current value of the destination transfer count. This value is required to determine the amount of data remaining to be transferred when a DMA channel is shut down using flush request. To assure an accurate value after issuing a flush request, the transfer count should be read only after receiving a DMA complete interrupt event. This ensures that all data was transferred and is reflected in the count value.

## 8.1.3 Flush Request (FR) Fields

Each DMA channel contains a write-only control field (FR) that allows the core real-time control over an active DMA transfer. A read by the core results in a value of zero. The FR bit (when set during an active transfer) shuts down the transfer and then returns the DMA channel to a condition ready for a new operation. For transfers to DRAM, the FR bit instructs the channel to disable reading source data and to finish transferring any data left in the internal data latch to DRAM. For transfers from DRAM, the FR bit instructs the DMA channel to disable reading source data and to discard any data left in the internal data latch that was read from DRAM. When completed, a DMA complete interrupt is posted and the DMA channel controllers return to the idle state.

## 8.2 GDMA CONTROL REGISTER

The GDMA control register (GDMCR) is used to configure the transfer direction, transfer data width, and DREQ input mode, as well as enable  $\overline{CS}_x$  during MC68322 bus cycles. This register is not double buffered and writing a new value during an active transfer will change the current operational mode of the GDMA channel. This is not recommended. When read by the core, the register reflects the current programmed bit fields. Figure 8-2 illustrates the GDMA control register.

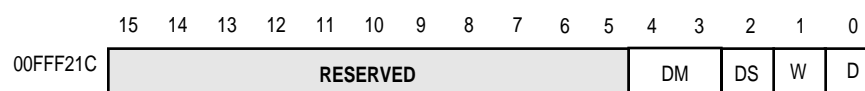


Figure 8-2. GDMA Control Register

The DM field is used to select how the GDMA channel recognizes an external DMA device request. Table 8-1 lists the field encodings and available options.

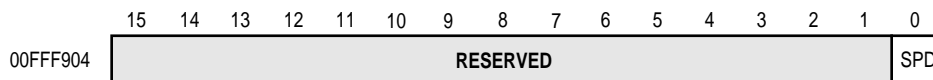
**Table 8-1. DM Field Encoding**

ENCODING	DREQ REQUEST OPTION
00	Active Low Level
01	Active High Level
10	Falling Edge Transition
11	Rising Edge Transition

Setting the DS bit causes the MC68322 bus interface signal ( $\overline{CS}_x$ ) to be asserted during a DMA-generated MC68322 bus cycle. If cleared,  $\overline{CS}_x$  will not be asserted during the cycle. When the W bit is set, the GDMA channel performs byte-sized transfers, but word-sized transfers when cleared. When the D bit is set, the GDMA channel transfers data from DRAM to an external DMA device and when cleared the GDMA channel transfers the data from an external DMA device to DRAM.

### 8.3 DMA SPEED REGISTER

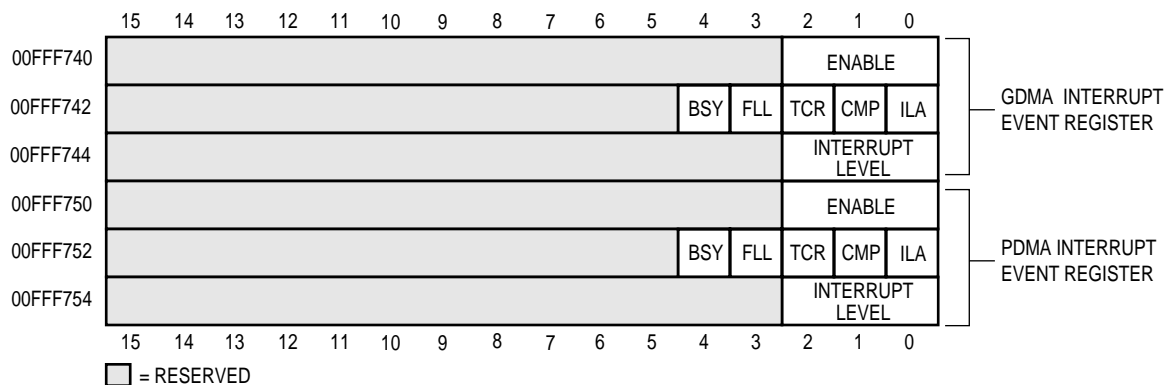
The SPD bit in the DMA speed (DMASP) register controls the maximum time that the GDMA waits between accesses to DRAM. When set, GDMA waits a maximum of 29 DRAM bus cycles plus six CLK1 periods for DRAM refresh. This is used for low-memory bandwidth devices such as serial ports. When DMASP is clear, GDMA waits a maximum of 17 DRAM bus cycles plus six CLK1 periods for DRAM refresh. This setting is used for high-memory bandwidth applications. GDMA latency occurs when the core, RISC graphics processor, print engine video controller, or an external bus master uses the DRAM bus. The minimum latency is zero. If the DRAM bus is idle when the GDMA makes a request, its DRAM bus cycle starts immediately. Figure 8-3 illustrates DMASP, which indicates the speed of the DMA throughput.



**Figure 8-3. DMA Speed Register**

## 8.4 DMA INTERRUPT EVENT REGISTERS

The DMA interface provides two memory-mapped interrupt event registers for each DMA channel. These registers provide real-time and interrupt status to the core. Figure 8-4 illustrates the two DMA interrupt registers.



**Figure 8-4. DMA Interrupt Event Registers**



**Note:** The DMA interrupt registers are located in the interrupt register portion of the memory map and are therefore not necessarily located contiguously with the other DMA registers.

Interrupt events for either channel can be individually enabled or disabled by programming each bit of the enable field. Setting each bit enables its respective event and clearing it masks the event. If an event is enabled, an interrupt is sent to the core. The following are GDMA and PDMA interrupt events and their bit field descriptions.

### BSY—Busy

This bit is set when the DMA channel's transfer count field is written, indicating that the channel is active and that the core should not write to the channel's configuration registers. This bit is cleared when the final transfer has completed and the channel remains idle and is ready to accept new register values.

### FLL—Full

This bit is set when byte- or word-sized data is held in the DMA data latch and is cleared when the transfer operation is complete. The combination of the BSY and FLL bits indicate real-time status.

### TCR and CMP—Terminal Count Reached and Complete

The TCR bit indicates when a DMA transfer is completed under normal termination (the number of transfer operations equals the value programmed in the channel's configuration register transfer count field). When the TCR bit is set, the DMA channel is finished with its transfer. The CMP bit is then set, indicating that the DMA registers are ready to be written for the next DMA transfer.

### ILA—Illegal Address

An illegal address interrupt is generated if a DMA channel attempts to access a DRAM or chip-select bank using an out-of-range address. This bit is set if the channel's configuration register GDTA, GCSTA, or PDTA fields do not access a valid address. If this error occurs, the DMA channel will halt all transfer operations and park in an error condition. The DMA channel must then be reset using the soft-reset register's GDR or PDR bits before starting a new operation.

The interrupt level field indicates the interrupt level. If any of the events in the channel's interrupt event register occur, an interrupt level indicated by this field is sent to the EC000 core. The interrupt levels range from seven to zero (zero disables interrupts).

## 8.5 INITIATING A DMA OPERATION

The DMA channel is activated after programming the channel's configuration register and writing the transfer count field. The software must ensure that a DMA channel is idle before writing the transfer count field. Writing a zero value to the transfer count field is allowed, but not recommended. New values should not be written into the DMA registers during active transfers because undefined results will occur.

For transfers to the MC68322 bus from DRAM, the GDMA immediately requests data from DRAM. After receiving data from DRAM, the GDMA begins monitoring the external DREQ input. When the external DMA device requests data, it is read out of the internal data latch and presented through the MC68322 bus to complete the data transfer. This process continues until all data is transferred, an address error occurs, or the transfer is terminated by the core through a flush request.

For data transfers to DRAM from the MC68322 bus, the GDMA begins monitoring the external DREQ input. When data is available from the external DMA device, the MC68322 bus cycle starts and the data is loaded into the internal data latch. Once the internal data latch is loaded with source data, the GDMA requests a DRAM cycle and, when granted, writes the data to memory. This process continues until all data is transferred, an address error occurs, or the transfer is terminated by the core through a flush request.

During an active DMA transfer, the system software can decide to reallocate the DMA resource to another device. By issuing a flush request, the current DMA operation will be terminated. When the CMP bit in the interrupt event register is set, new values for the DMA channel registers can be written and the channel restarted.

## 8.6 DMA TRANSFERS

Because both DMA channels operate on a demand basis, all DMA-initiated transfers (through either the PDMA or GDMA channels) use a data request and acknowledge type handshake.

The GDMA supports both word- and byte-sized transfers. For word-sized transfers, the external DMA device must connect to D15–D0 and for byte-sized transfers the device must connect to D7–D0.  $\overline{WRU}$  and  $\overline{WRL}$  signals are both asserted during an MC68322 word-sized bus write cycle, but only  $\overline{WRL}$  is asserted during byte-sized writes.

Support for byte-sized transfers is handled automatically by the DMA interface through data packing and it requires no external logic. Byte-sized data received from the core or PPI are packed into words prior to requesting a DRAM access. Data sent to the core as bytes are read in as words from DRAM and unpacked to bytes for transfer. All data transfers to and from memory are in big endian format, thus assuring compatibility with the processor's data organization in memory.

### 8.6.1 PDMA Transfers

Handshaking for the PDMA channel is transparent and handled internally between the PPI and the PDMA channel. A typical transfer cycle for the PPI is described in **Section 9 Parallel Port Interface**.

### 8.6.2 GDMA MC68322 Bus Read and Write Cycles

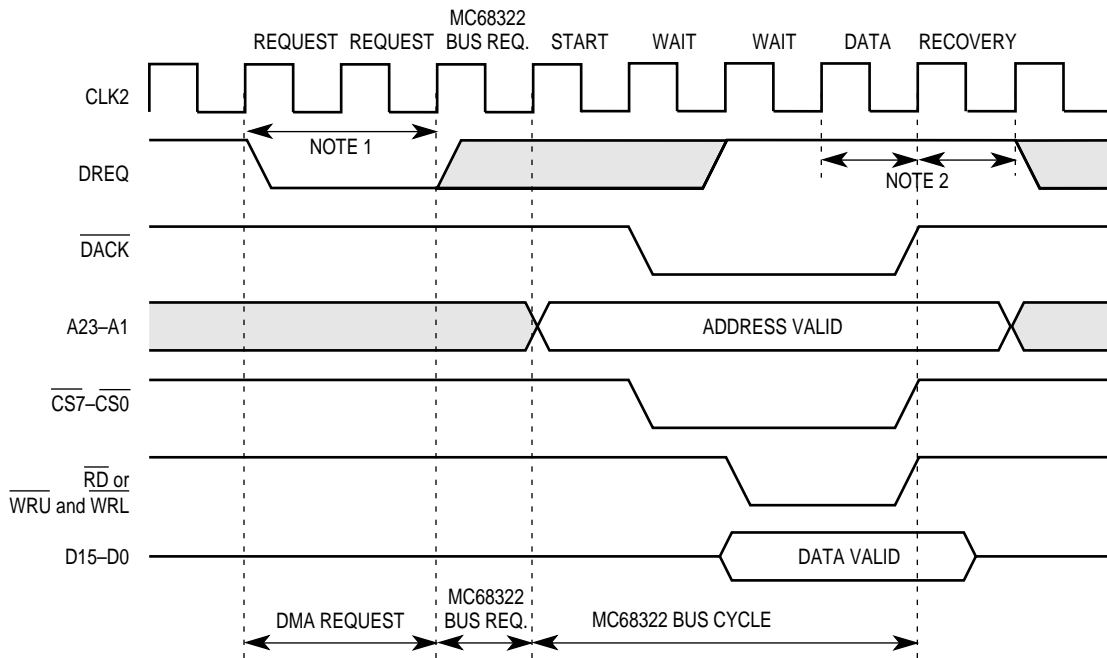
GDMA handshaking uses the DREQ and  $\overline{DACK}$  signals. DREQ is an input signal that has a programmable level or edge sensitivity with polarity control. The default configuration is active low level.  $\overline{DACK}$  is an output signal that is used by the external DMA device to acknowledge that a GDMA cycle is in progress. The polarity for the  $\overline{DACK}$  output is fixed to provide an active low output.

A GDMA cycle is requested when DREQ is asserted by the external DMA device, such as PROM or I/O. After the DMA interface internally synchronizes DREQ, it arbitrates for the MC68322 bus and, when granted, asserts  $\overline{DACK}$ . After  $\overline{DACK}$  is asserted, the internal bus interface unit (BIU) performs the GDMA cycle defined by the GDMA configuration register. After the cycle completes,  $\overline{DACK}$  is negated and the DMA cycle terminates.

During a GDMA bus cycle,  $\overline{CS}_x$  can be disabled. This option supports external DMA devices that require  $\overline{CS}_x$  to be inactive during a DMA operation. The DS bit in the GDMA configuration register controls the operation of  $\overline{CS}_x$  during a DMA-generated MC68322 bus cycle.

A GDMA MC68322 bus write cycle occurs when the GDMA channel is configured to transfer from DRAM to an external DMA device. A word-sized GDMA write cycle asserts the  $\overline{WRU}$  and  $\overline{WRL}$  signals and a byte-sized GDMA write cycle asserts  $\overline{WRL}$  and negates  $\overline{WRU}$ . Figure 8-5 illustrates the typical timing for a fast DMA read or write cycle. The bus cycle timing shown is obtained using minimum values for all timing parameters.





NOTES:

- 1) This diagram illustrates DREQ programmed as an active low input. DREQ is an asynchronous input and is synchronized internally by the GDMA interface; it requires no setup or hold time to be recognized for proper operation. However, to guarantee recognition of the input at a certain edge of CLK, DREQ must satisfy a setup requirement that it remain active for at least two consecutive CLK rising edges to be detected by the GDMA interface.
- 2) Setup and hold requirements must be met to prevent the start of the next GDMA cycle. If back to back GDMA cycles are preferred, DREQ must stay active and detected as a low at this time.

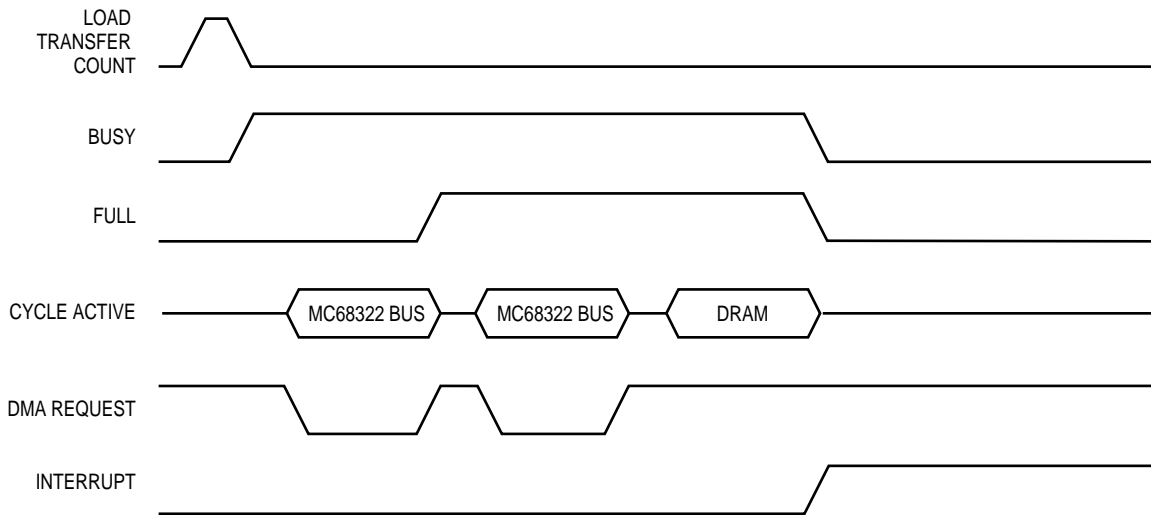
**Figure 8-5. GDMA MC68322 Bus Read Or Write Cycle**

The GDMA can be mapped to any chip-select bank. To optimize DMA access timing, the chip-select DMA timing register is provided. See **Section 6 System Integration Module** for more information. This register provides a separate chip-select bank timing that is specific to a DMA access. During a GDMA access to a chip-select bank, the internal  $\overline{DACK}$  signal has a timing that is identical to the  $\overline{CS}_x$ . The DS bit in the GDMA configuration register controls the assertion of the  $\overline{CS}_x$  with  $\overline{DACK}$  during a GDMA access.

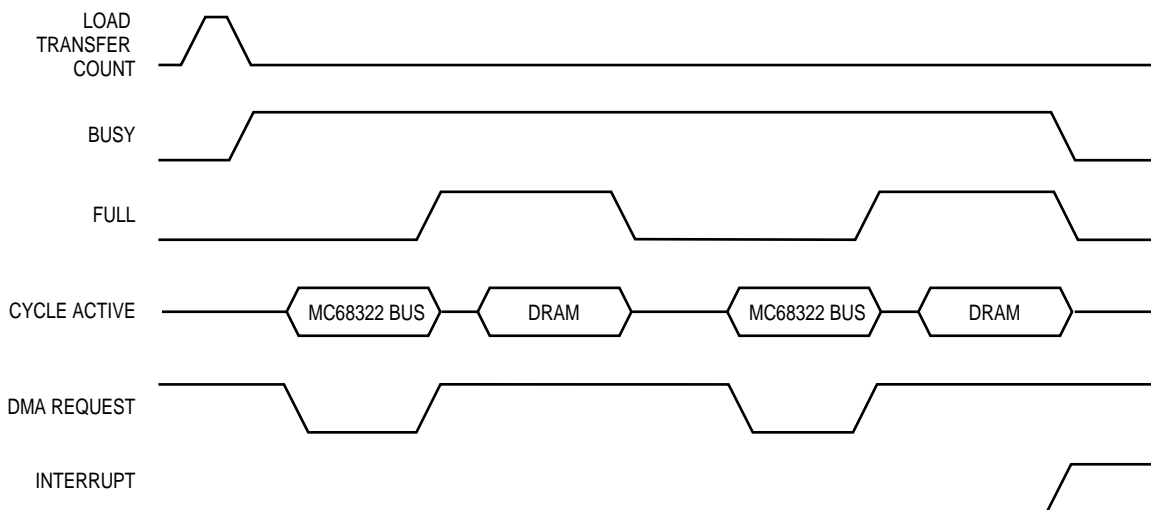
### 8.6.3 GDMA DRAM Bus Read and Write Cycles

The DMA interface is one of five interfaces that internally arbitrates for control of the DRAM bus. A GDMA DRAM bus read cycle request is generated when the internal data latch can accept new data. This can occur either when the DMA interface is first started or after data is transferred to a destination. A GDMA DRAM bus write cycle request is generated as soon as data is available from the DMA source.

Figures 8-6 and 8-7 illustrate byte- and word-sized DMA write transfers across both MC68322 and DRAM buses. In these figures, the transfer count register is set to two.



**Figure 8-6. Byte-Sized DMA DRAM Write Transfer**



**Figure 8-7. Word-Sized DMA DRAM Write Transfer**

## 8.7 DMA TRANSFER TERMINATION

A DMA transfer can terminate when one of the three following conditions occur—normal termination, bad address, or an core-forced termination.

### 8.7.1 Normal Termination

Normal termination is the normal conclusion of a DMA transfer and it occurs when the source and destination transfer counters have decremented to zero and no address value errors have occurred. When the DMA transfer completes, the **CMP** and **TCR** bits in the channel's interrupt event register are set and an interrupt is posted to the core if enabled.

## 8.7.2 Bad Address Termination

If at some point during the DMA transfer an address supplied by the DMA to the DRAM or MC68322 bus interfaces is incorrect (not mapped to a valid bank), an address fault occurs. The DMA transfer terminates, the ILA bit in the channel's interrupt event register is set, and an interrupt is posted to the core if enabled. As a result of this bad address error, the DMA channel will park in an error condition. The DMA channel must then be reset using one of soft-reset register's DMA bits before a new operation can be initiated. A write to either the GDR or PDR bits in the soft-reset register will initiate a DMA soft-reset operation. See **Section 5 Interrupt and Exception Handling** for more information. During the reset condition, only the DMA channel control logic is affected. Initial register values are not cleared.

## 8.7.3 Core-Forced Termination

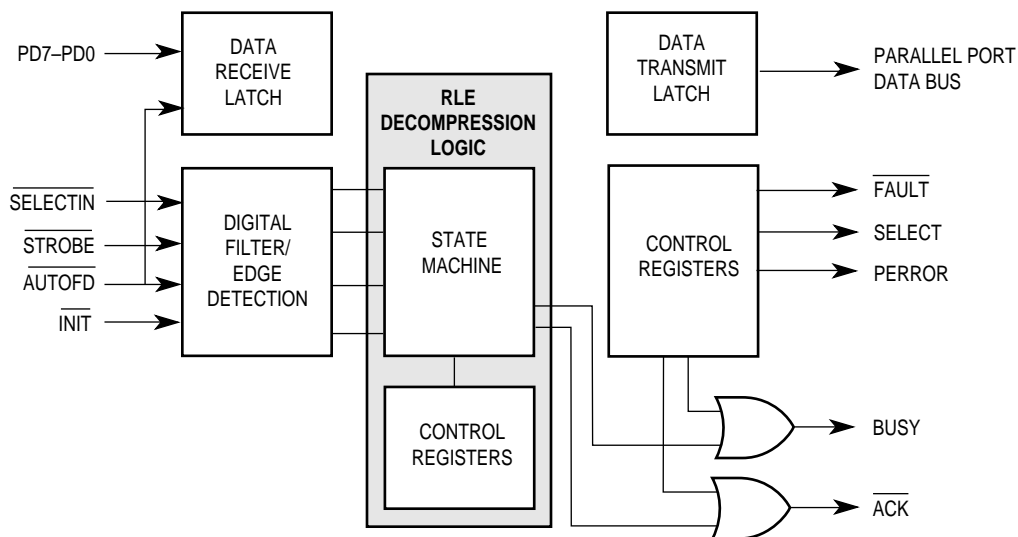
If, during a DMA transfer, the core sets the FR bit in the channel's configuration register, the DMA controller will terminate any ongoing transfer for that channel. If the DMA channel is receiving data from the MC68322 bus or the PPI, any data remaining in the internal data latch is written to DRAM. If the DMA channel is sending data to the MC68322 bus, any data remaining in the data latch is discarded. When the data transfers are complete, the channel shuts down, sets the CMP bit in the channel's interrupt event register, and returns to the idle state. Depending on the value remaining in the transfer counters, it is possible to set the FR bit in the channel's configuration register, and still have the DMA transfer terminate normally due to the transfer counters decrementing to zero. If this condition occurs, both the CMP and TCR bits will be set in the channel's interrupt event register.

## SECTION 9 PARALLEL PORT INTERFACE

The MC68322 contains a direct-connect, fully IEEE 1284 Level 2 compliant, bidirectional parallel port interface (PPI). The PPI supports four IEEE 1284 communications modes—compatibility (Centronics™), nibble, byte, and enhanced capabilities port (ECP). It also supports all variants of these modes, including device ID requests and run-length encoded (RLE) data compression.

The PPI contains specific hardware to support automatic handshaking during host to peripheral (forward) data transfers in compatibility and ECP modes, and run-length detection and decompression of host to peripheral data during ECP transfers. This can substantially improve data rates when operating the parallel port in compatibility or ECP mode. When hardware handshaking is used in combination with the PPI's dedicated DMA controller, data rates as high as 2M per second can be achieved in ECP forward mode. The software can disable and enable hardware handshaking to allow direct control of PPI signals as well as to support future protocols. The remainder of IEEE 1284 operations (negotiation, reverse data transfers, and termination cycles) must be carried out by the software. Please note that IEEE 1284 EPP communications mode is not supported.

The PPI contains an interface controller that consists of a data receive and transmit latch, run-length encoding decompression logic, input conditioning logic, and edge detector logic. The RLE decompression is accomplished through a smart state machine and additional control logic. The PPI has input conditioning logic to filter the incoming control signals. Figure 9-1 illustrates the PPI controller block diagram.



**Figure 9-1. Parallel Port Interface Controller Block Diagram**

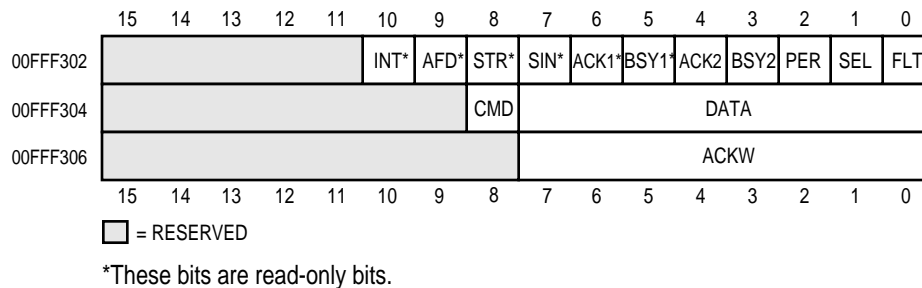
## 9.1 PPI REGISTERS

There are three memory-mapped registers that control the PPI:

- PPI interface register
- PPI control register
- PPI interrupt event register

### 9.1.1 Parallel Port Interface Register

The parallel port interface register (PPIR) is a read/write register that contains two 8-bit fields—one that controls the  $\overline{ACK}$  pulse width and another that contains the latched parallel data transmitted from the host to the printer. The PPIR also contains eleven bits that control the parallel port interface signals. These eleven bits consist of four read-only bits that are used to read the logic level of the host input pins, two read-only bits to read the logic level on the BUSY and  $\overline{ACK}$  printer output pins, and five read/write bits control the logic levels on the printer output pins. Figure 9-2 illustrates the parallel port interface register.



**Figure 9-2. Parallel Port Interface Register**

#### ACKW— $\overline{ACK}$ Width

This field defines the  $\overline{ACK}$  pulse width when compatibility mode is enabled (PPCR MODE field = 01). The  $\overline{ACK}$  pulse width is selectable from 0 to 255 CLK1 periods wide. At 16 MHz, the software can set pulse widths anywhere in the range of 0 to 16  $\mu$ s. If the field is clear, no  $\overline{ACK}$  pulse is issued. Otherwise, the cycle proceeds as normal.

ACKW can be changed at any time and with any PPCR MODE encoding selected, but it can only be used during a compatibility mode handshaking cycle. If ACKW is changed near the end of a data transfer (when an  $\overline{ACK}$  is already low), then the new pulse width value does not affect the current cycle. The new value of ACKW is used when the next cycle occurs.

#### CMD—Command

When read, this bit provides the logic level of  $\overline{AUTOFD}$  when  $\overline{STROBE}$  transitioned from high to low with the PPCR's PDE bit clear. If set,  $\overline{AUTOFD}$  was latched high and if clear,  $\overline{AUTOFD}$  was latched low. This is a read-only bit, so writing CMD has no effect.

## DATA

This field is an 8-bit read/write field. When read, DATA provides the latched logic levels on the parallel port data bus when  $\overline{\text{STROBE}}$  last transitioned from high to low with the PPCR's PDE clear. When written, the value defines the logic levels to be driven by the MC68322 when PD7–PD0 is enabled by setting the PDE bit. The most-significant bit of the DATA field corresponds to PD7 and the least-significant bit to PD0.

The CMD and DATA fields latch the logic levels on the parallel port data bus (PD7–PD0) and  $\overline{\text{AUTOFD}}$  pins, which is used to indicate command bytes during ECP mode forward data transfers. The CMD and DATA fields should not be read while the PDMA channel is enabled; doing so clears the PDMA request.

### INT— $\overline{\text{INIT}}$ Status (Read-only)

Indicates the level read on  $\overline{\text{INIT}}$  after synchronization and optional digital filtering.

### AFD— $\overline{\text{AUTOFD}}$ Status (Read-only)

Indicates the level read on  $\overline{\text{AUTOFD}}$  after synchronization and optional digital filtering.

### STR— $\overline{\text{STROBE}}$ Status (Read-only)

Indicates the level read on  $\overline{\text{STROBE}}$  after synchronization and optional digital filtering.

### SIN— $\overline{\text{SELECTIN}}$ Status (Read-only)

Indicates the level read on  $\overline{\text{SELECTIN}}$  after synchronization and optional digital filtering.

### ACK1— $\overline{\text{ACK}}$ Status (Read-only)

Indicates the level driven on  $\overline{\text{ACK}}$ . This bit is the ACK2 bit NOR'ed with  $\overline{\text{ACK}}$  from the PPI state machine. This bit is set on reset.

### BSY1—BUSY Status (Read-only)

Indicates the level driven on BUSY. This bit is the BSY2 bit OR'ed with BUSY from the PPI state machine. This bit is set on reset.

### ACK2— $\overline{\text{ACK}}$ Control

ACK2 forces a low level to be driven on  $\overline{\text{ACK}}$ . This is generally done when hardware handshaking is disabled and the PPI state machine is idle. The ACK2 bit is NOR'ed with  $\overline{\text{ACK}}$  from the PPI state machine before driving  $\overline{\text{ACK}}$ . If ACK2 is set, then the  $\overline{\text{ACK}}$  pin is forced low and the ACK1 bit is cleared.

### BSY2—BUSY Control

BSY2 forces a high level to be driven on BUSY. This is generally done when hardware handshaking is disabled and the PPI state machine is idle. The BSY2 bit is OR'ed with BUSY from the PPI state machine before driving BUSY. If BSY2 is set, then BUSY is forced high and the BSY1 bit is set. BSY2 is set on reset.

### PER—PERROR Control

Setting this bit drives a high level and clearing it drives a low level on PERROR.

**SEL**—SELECT Control Bit

Setting this bit drives a high level and clearing it drives a low level on SELECT.

**FLT**— $\overline{\text{FAULT}}$  Control Bit

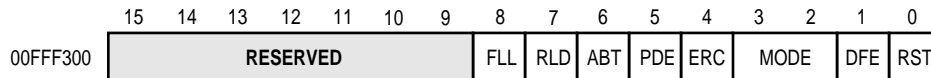
Setting this bit drives a high level and clearing it drives a low level on  $\overline{\text{FAULT}}$ .



**Note:** The FLT, SEL, PER, and BSY2 bits are arranged to correspond to their use as parallel port data lines during nibble mode reverse data transfers. When hardware handshaking is enabled, BUSY and  $\overline{\text{ACK}}$  are controlled by the PPI state machine. When hardware handshaking is disabled and the PPI state machine is idle, BUSY and  $\overline{\text{ACK}}$  can be controlled by the software using the BSY2 and ACK2 bits.

### 9.1.2 Parallel Port Control Register

The parallel port control register (PPCR) is a read/write register containing nine bits that configure the operation of the PPI. Figure 9-3 illustrates the PPCR.



**Figure 9-3. Parallel Port Control Register**

**FLL**—Full

This bit is a read-only status bit that indicates when parallel port data from the host is latched in the DATA field of the PPIR. FLL is cleared when the PPIR’s DATA field is read. When handshaking and DMA are enabled, FLL is set and then cleared as data is latched and read during forward data transfers. The FLL bit is also cleared when the RST bit is set.

**RLD**—Run-Length Decompression

This bit is a read-only status bit that indicates when run-length decompression is taking place during ECP forward data transfers. RLD is set when a run-length count is received and loaded into the internal counter and cleared when the last of the decompressed data is read from the PPIR’s DATA field. This bit can only be set when ECP with RLE (MODE = 11<sub>2</sub>) is enabled. If the MODE field is reprogrammed during decompression, decompression continues and RLD remains set until the operation is complete. RLD is also cleared when RST is issued.

**ABT**—Abort

This bit causes the PPI to use  $\overline{\text{SELECTIN}}$  to detect when the host suddenly aborts a reverse transfer and returns to compatibility mode. If ABT is set, a low level on  $\overline{\text{SELECTIN}}$  causes the PDE bit to be cleared and the PD7–PD0 output drivers to be three-stated. In fact, if ABT is set and  $\overline{\text{SELECTIN}}$  is low, setting the PDE bit has no effect. This protection logic, as with all internal PPI logic, operates on a synchronized and optionally digitally filtered  $\overline{\text{SELECTIN}}$  that is latched into the PPIR.

### PDE—Parallel Port Data Bus Output Enable

This bit performs two functions—controls the state of the three-state output buffers and qualifies the latching of data from the output drivers into the PPIR's DATA field. When clear, PDE disables the PD7–PD0 output buffers and allows data to be latched into the DATA field on every high to low transition of  $\overline{\text{STROBE}}$ . When set, PDE enables the PD7–PD0 output buffers, preventing data from being latched into the DATA field. In this state, the DATA field is unaffected by transitions on  $\overline{\text{STROBE}}$ . Setting the ABT bit affects the operation of PDE. If the ABT bit is set,  $\overline{\text{SELECTIN}}$  must remain high to allow PDE to be set or remain set. If the ABT bit is set and  $\overline{\text{SELECTIN}}$  goes low, PDE is cleared, and setting PDE will have no effect.

### ERC—Error Cycle

The ERC bit is used to execute an error cycle when in compatibility mode ( $\text{MODE} = 01_2$ ). When set, ERC sets the BSY1 bit in the PPIR, which immediately causes the MC68322 to drive BUSY high. If ERC is set when a compatibility mode handshake sequence is in progress, BSY1 remains set beyond the end of the cycle. The ERC bit does not affect an  $\overline{\text{ACK}}$  pulse that is already active, but does prevent an  $\overline{\text{ACK}}$  pulse if it is about to be generated. While ERC is set, the software can set or clear the PPIR's SEL, PER, and FLT bits. When ERC is cleared, the PPI generates an  $\overline{\text{ACK}}$  pulse and negates BUSY to automatically conclude the error cycle.

When the MODE bit is set to any value except  $01_2$ , setting ERC has no effect. Setting  $\text{MODE} = 01_2$  when ERC is already set, causes the handshake controller to immediately begin an error cycle as described above.

### MODE

This 2-bit field selects and enables a hardware handshaking mode for forward data transfers. The following paragraphs describe the functions for encoding the MODE field.

- 00 = Disable all hardware handshaking so that handshaking can be performed by the software.
- 01 = Enable compatibility mode hardware handshaking during forward data transfers. In this mode, the PPI responds to a high to low transition on  $\overline{\text{STROBE}}$  and automatically sets and clears the BSY1 and ACK1 bits in the PPIR to handshake with the host. MODE can be reprogrammed at any time, but if a compatibility mode cycle is currently in progress, it completes as normal. However, MODE should only be changed from compatibility mode handshaking when BUSY is high. This ensures that no parallel port activity is taking place when reconfiguring the PPI.
- 10 = Enable ECP mode hardware handshaking without RLE support during forward data transfers. In this mode, the PPI responds to a high to low transition on  $\overline{\text{STROBE}}$  and automatically sets and clears the BSY1 bit in the PPIR to handshake with the host. Reception of both run-length counts and channel addresses causes the PPIR's CRD bit to be set. The software is responsible for responding to channel addresses and performing data decompression. MODE



can be reprogrammed at any time, but if an ECP cycle is currently in progress, it completes as normal.

- 11 = Enable ECP mode hardware handshaking with RLE support during forward data transfers. The PPI performs the same ECP mode handshaking as above, except RLE decompression is enabled. RLE decompression enables the PPI to detect and intercept run-length counts and to automatically perform data decompression. However, during this mode, only channel addresses cause the PIER's CRD bit to be set. The software is responsible for responding to channel addresses. If MODE is reprogrammed when decompression is occurring (when the RLD bit is set), the decompression continues unhindered to completion. The RST bit can be set to immediately abort decompression.

**DFE—Digital Filtering Enable**

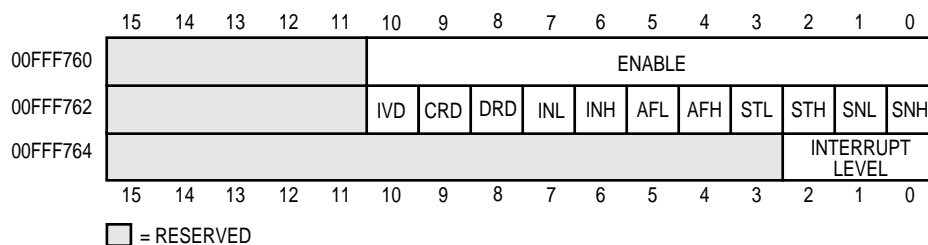
Setting this bit enables digital filtering on all four host control signal inputs— $\overline{\text{SELECTIN}}$ ,  $\overline{\text{STROBE}}$ ,  $\overline{\text{AUTOFD}}$ , and  $\overline{\text{INIT}}$ .

**RST—Reset**

Setting this bit causes the PPI's handshake control and decompression logic to immediately terminate the current operation and return to idle. RST clears the RLD and FLL bits. The PPI state machine BUSY and  $\overline{\text{ACK}}$  are negated. If the PPIR's BSY2 and ACK2 bits are clear, then BSY1 = 0 and ACK1 = 1. The software should set the MODE field to 00 to disable handshaking when setting RST to prevent the PPI state machine from starting again. RST is a write-only bit and setting it causes the reset. Clearing RST has no effect. This bit always reads as zero.

**9.1.3 PPI Interrupt Event Register**

The PPI interrupt event register (PIER) contains 11 bits that can be enabled and used to drive the parallel port using a software driver. Eight of the eleven bits indicate when a rising or falling edge is seen on any of  $\overline{\text{SELECTIN}}$ ,  $\overline{\text{INIT}}$ ,  $\overline{\text{AUTOFD}}$ , or  $\overline{\text{STROBE}}$  host inputs. The remaining three bits indicate when a data or command byte is received or when an invalid termination event is detected. Figure 9-4 illustrates the PIER.



**Figure 9-4. PPI Interrupt Event Register**

The following bits describe each of the parallel port interrupt events that can be posted by the PPI. The first eight interrupt events signal level changes that occur at the host control signal input pins. Note that these events are detected after the host inputs are synchronized, optionally digitally filtered, and recorded in the PPIR.

**IVD—Invalid Transition**

This bit is set when  $\overline{\text{SELECTIN}}$  transitions from high to low in the middle of an ECP forward data transfer handshake sequence. The invalid transition interrupt is posted if  $\overline{\text{SELECTIN}}$  is low when  $\overline{\text{STROBE}}$  is low or  $\text{BUSY}$  is high. This event can be posted only when ECP is enabled.

**CRD—Command Received**

This bit is set when a command byte is latched into the PPIR's DATA field. If ECP without RLE ( $\text{MODE} = 10_2$ ) is enabled, then a command received interrupt is posted when a run-length or channel address is received. If ECP with RLE ( $\text{MODE} = 11_2$ ) is enabled, then a command received interrupt is posted when a channel address is received. This event can only be posted when ECP is enabled.

**DRD—Data Received**

This bit is set when data is latched into the PPIR's DATA field, which occurs on every high to low transition of  $\overline{\text{STROBE}}$  when the PPCR's PDE bit is clear. This interrupt is also set if ECP with RLE ( $\text{MODE} = 11_2$ ) is enabled and data decompression is in progress.

**INL— $\overline{\text{INIT}}$  Low**

This bit is set when a high-to-low transition on  $\overline{\text{INIT}}$  is reported in the PPIR.

**INH— $\overline{\text{INIT}}$  High**

This bit is set when a low-to-high transition on  $\overline{\text{INIT}}$  is reported in the PPIR.

**AFL— $\overline{\text{AUTOFD}}$  Low**

This bit is set when a high-to-low transition on  $\overline{\text{AUTOFD}}$  is reported in the PPIR.

**AFH— $\overline{\text{AUTOFD}}$  High**

This bit is set when a low-to-high transition on  $\overline{\text{AUTOFD}}$  is reported in the PPIR.

**STL— $\overline{\text{STROBE}}$  Low**

This bit is set when a high-to-low transition on  $\overline{\text{STROBE}}$  is reported in the PPIR.

**STH— $\overline{\text{STROBE}}$  High**

This bit is set when a low-to-high transition on  $\overline{\text{STROBE}}$  is reported in the PPIR.

**SNL— $\overline{\text{SELECTIN}}$  Low**

This bit is set when a high-to-low transition on  $\overline{\text{SELECTIN}}$  is reported in the PPIR.

**SNH— $\overline{\text{SELECTIN}}$  High**

This bit is set when a low-to-high transition on  $\overline{\text{SELECTIN}}$  is reported in the PPIR.

## 9.2 HARDWARE HANDSHAKING

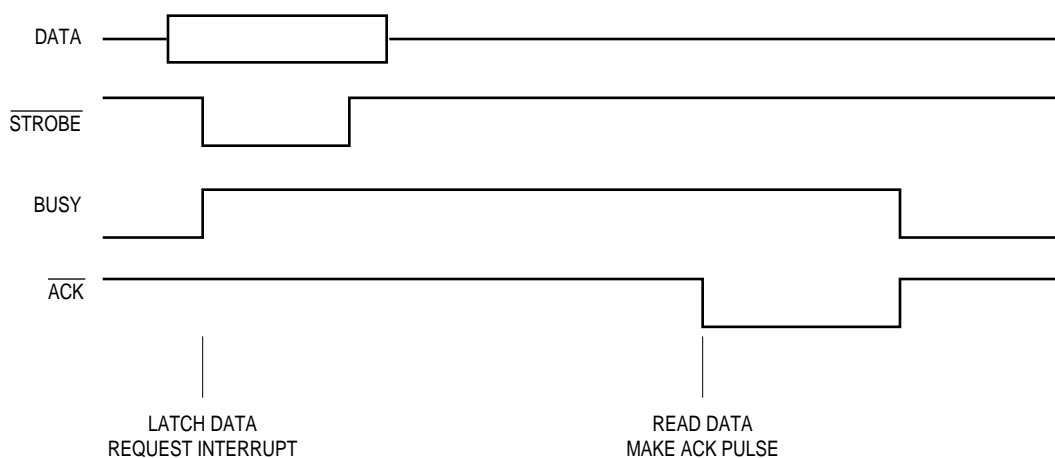
The PPI supports three hardware handshaking modes for forward data transfers that are enabled and disabled by the software. One mode supports forward data transfers during compatibility mode and the other two modes support forward data transfers during ECP mode. Only one of the three modes can be enabled at a time or all modes can be disabled. When disabled, the software must assume full responsibility for handshaking and use the PIER and PPIR to read and control the logic levels on all parallel port pins.

### 9.2.1 Compatibility Handshaking

Compatibility mode hardware handshaking is enabled by setting the PPCR's  $MODE = 01_2$ . When this mode of handshaking is enabled, the PPI automatically generates  $BUSY$  when the leading edge of  $\overline{STROBE}$  from the host is received and latches the logic levels on  $PD7-PD0$  and  $\overline{AUTOFD}$  into the PPIR. The PPI then waits for  $\overline{STROBE}$  to negate and the PPIR's  $DATA$  field to be read. After both of these events occur, the PPI asserts  $\overline{ACK}$  for the duration specified in the PPIR's  $ACKW$  field and then negates  $\overline{ACK}$  and  $BUSY$  to conclude the data transfer.

When data is latched into the PPIR's  $DATA$  field, the PPI generates two interrupt events—a parallel port DMA (PDMA) request and a core interrupt request. The software can alternate between PDMA and interrupt-based data transfers. The PDMA request remains active until the data is read by either the PDMA channel or core in response to the interrupt event. If the PDMA has been enabled, it responds to the request by reading the PPIR's  $DATA$  field and writing the data to DRAM. When the PDMA channel reads the  $DATA$  field,  $\overline{ACK}$  is pulsed,  $BUSY$  is negated, and the PDMA request is cleared.

The PIER's  $ENABLE$  field can be programmed to interrupt the core when parallel port data is received. If enabled, the PIER's  $DRD$  bit, when set, will generate a data received interrupt event. When the core reads the PPIR's  $DATA$  field,  $\overline{ACK}$  is pulsed,  $BUSY$  is negated, and the PDMA request is cleared. The PPI functions without using the PDMA, but the system throughput will be dramatically impacted. Therefore, using the PDMA is strongly recommended. Figure 9-5 illustrates the compatibility mode timing.



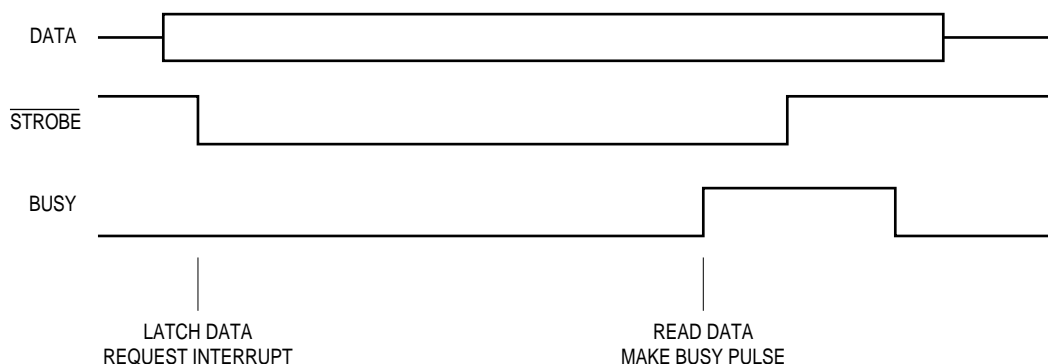
**Figure 9-5. Compatibility Mode Timing Diagram**

## 9.2.2 ECP Handshaking

The PPI supports two ECP hardware handshaking modes for forward data transfers—one with run-length encoding and one without. ECP hardware handshaking is enabled by setting the MODE field in the PPCR to 102, while ECP hardware handshaking with RLE is enabled by setting MODE = 112. The basic operation of these two handshake modes is identical.

When either mode is enabled, the PPI automatically responds to  $\overline{\text{STROBE}}$  by latching the logic levels on PD7–PD0 and  $\overline{\text{AUTOFD}}$  in the PPIR's DATA field. When the PPIR's DATA field is read, the PPI drives BUSY high, waits for  $\overline{\text{STROBE}}$  to go high, and then drives BUSY low to conclude the cycle. Since no  $\overline{\text{ACK}}$  pulse is generated, the pulse width duration specified in the PPIR's ACKW field is not used. Like compatibility handshake mode, data is latched at the leading edge of  $\overline{\text{STROBE}}$ , thus causing a PDMA request and the posting of a data received interrupt event (setting the DRD bit in the PIER). DMA and interrupt operation is the same as described for compatibility handshaking mode.

Two additional interrupts events can be posted by the PPI when an ECP handshake mode is programmed—command received and invalid termination (setting the CRD and/or IVD bits in the PIER). A CRD is posted when a command byte is received from the host and an IVD is posted when  $\overline{\text{SELECTIN}}$  transitions illegally in the middle of a handshake sequence; that is, an invalid termination interrupt is posted if  $\overline{\text{SELECTIN}}$  is low when  $\overline{\text{STROBE}}$  is low or BUSY is high. Such an event can be caused by the user changing a switch box or a parallel cable coming loose. An invalid termination interrupt event should be treated as an immediate termination and the PPI should be returned to compatibility mode operation. Figure 9-6 illustrates the ECP mode timing.



**Figure 9-6. ECP Mode Timing Diagram**

**9.2.2.1 COMMAND BYTE DETECTION.** When ECP is enabled (MODE = 102 or 112), the PPI monitors the  $\overline{\text{AUTOFD}}$  level that is latched into the PPIR's CMD bit, to detect and intercept command bytes. If CMD is clear, the PPI interprets the data as a command byte and examines the most-significant bit to interpret the byte. If PD7 = 0, the command byte is a run-length count and if it is 1 the command byte is a channel address. The action taken with a channel address or run-length count depends on which ECP handshaking mode is selected.

If ECP without RLE is enabled ( $MODE = 10_2$ ), a channel address and run-length count always generates a command received interrupt event. The PPI does not perform any decompression.  $BUSY$  remains low until the PPIR's DATA field is read by the core. No data received interrupt event is posted and no PDMA request is generated for the command byte. If ECP with RLE is enabled ( $MODE = 11_2$ ) and a channel address is received, a command received interrupt event is posted.  $BUSY$  remains low until the PPIR's DATA field is read by the core. If the PPI detects a run-length count, RLE decompression will occur on the next data byte that is received. No command or data received interrupt events are posted and no PDMA request is generated for the run-length command byte.

**9.2.2.2 RLE DECOMPRESSION.** When ECP with RLE handshaking is enabled, run-length counts are detected and automatically loaded into an internal counter. The PPI then sets the PPCR's RLD bit. A run-length count of zero is interpreted as a replication factor of one and a run-length count of 127 is interpreted as a factor of 128.  $BUSY$  is driven high when the run-length value is loaded into the counter and lowered when  $\overline{STROBE}$  returns high. The PPI then waits for the next data byte.

When the next data byte arrives,  $BUSY$  remains low for the entire decompression period. In the MC68322, depending on internal bus utilization, a replication factor of 128 could result in  $BUSY$  remaining low for 1 to 2 ms while  $\overline{STROBE}$  remains low. As decompression occurs, the PPI generates PDMA requests and data received interrupts events in exactly the same manner as when it receives uncompressed data. When the run-length counter decrements to zero, the last PPIR DATA field read results in the  $BUSY$  being driven high and then low following the rising edge of  $\overline{STROBE}$ . The RLD bit is also cleared at this time. After a run-length command is received the next byte usually is a data byte. If the next byte is also a run-length, then this new run-length is used. If the next byte is a channel address, then a command received interrupt event is posted and the PPI state machine continues waiting for the next data byte.

If the PPCR's MODE field is changed after the run-length command was received, but before the data byte was received, then RLD remains set. When ECP with RLE is enabled again, decompression will begin when the data byte is received. If MODE is changed while decompression is occurring (RLD is set and the data byte was received), the decompression continues until completion. If decompression must be immediately aborted, RST should be issued in the PPCR.

The PPCR's FLL bit is set when the run-length is received and immediately cleared when the run-length is loaded into the counter. FLL is set again when the data byte is received and cleared when the run-length counter reaches zero and the last data byte is decompressed.

### 9.2.3 Disabling Hardware Handshaking

When hardware handshaking is disabled ( $MODE = 00_2$ ), the software is responsible for controlling the PPI. However, the PPIR's DATA field continues to latch parallel port data on the leading edge of  $\overline{STROBE}$ , even with all hardware handshaking disabled. Also, host control signal inputs are always synchronized to CLK1 to prevent metastable events from reaching the internal logic of the MC68322. This does not include PD7–PD0 since they are guaranteed to be stable when  $\overline{STROBE}$  is low.

### 9.3 SOFTWARE-CONTROLLED HANDSHAKING

By clearing the PPCR's MODE field, the software controls all parallel port operations, including negotiation and termination phases, as well as reverse data transfers. The software can also respond to parallel port inputs by way of polling or interrupts. This gives the software the flexibility to adapt to new and revised protocols. The software controls BUSY and  $\overline{\text{ACK}}$  with the BSY2 and ACK2 bits in the PPIR. Normally, the PPI state machine should be idle when using BSY2 and ACK2. The software can issue a reset (by setting the RST bit in the PPCR) to force the PPI state machine to idle.

When polling, status bits in the PIER report the logic level at each parallel port input pin and control bits allow separate and direct control of each of the output pins. The host inputs are always synchronized to the internally generated CLK1 signal to prevent metastable events from reaching the microprocessor. This results in a maximum of one CLK1 period of delay before an external event appears in the PIER. The  $\overline{\text{SELECTIN}}$ ,  $\overline{\text{INIT}}$ ,  $\overline{\text{AUTOFD}}$ , and  $\overline{\text{STROBE}}$  signals can also be digitally filtered to improve noise immunity. Digital filtering adds another CLK1 period of delay before level changes on these signals are indicated in the PIER.

### 9.4 DIGITAL FILTERING

The MC68322 contains digital filter circuitry on host control signal inputs ( $\overline{\text{SELECTIN}}$ ,  $\overline{\text{STROBE}}$ ,  $\overline{\text{AUTOFD}}$ , and  $\overline{\text{INIT}}$ ) to improve noise immunity and make the PPI more impervious to inductive switching noise. Digital filtering can be enabled, regardless of whether hardware handshaking is enabled or disabled. When digital filtering is disabled, the host control signals are synchronized to the internally generated CLK1 signal to prevent metastable events from reaching the internal logic of the MC68322. However, the synchronization logic does not prevent glitches on the host control signals from reaching the PPI's internal logic and causing spurious events.

When digital filtering is enabled, the host control signals are first synchronized and then passed through individual digital filters. The digital filter samples the host input on the rising edge of CLK1 and passes a logic level change through, but only if the host signal is sampled at the same logic level for a second consecutive clock. Digital filtering protects internal logic from glitches as wide as one CLK1 period. Such internal logic includes the hardware handshake control logic, the PPCR, and the parallel port interrupt controller.

Synchronization plus digital filtering adds two CLK1 periods of delay before a level change on one of the host signals appears in the PPCR, and three CLK1 periods of delay before an output responds to an input (before BUSY responds to  $\overline{\text{STROBE}}$ ). Likewise, synchronization and digital filtering of  $\overline{\text{STROBE}}$  affect the point at which PD7–PD0 and  $\overline{\text{AUTOFD}}$  are latched into the PPIR. Without digital filtering, PD7–PD0 and  $\overline{\text{AUTOFD}}$  are sampled on the second rising edge of CLK1 after  $\overline{\text{STROBE}}$  is first sampled low. With digital filtering, PD7–PD0 and  $\overline{\text{AUTOFD}}$  are sampled on the third rising edge of CLK1 after  $\overline{\text{STROBE}}$  is first sampled low. Digital filtering can be disabled to avoid the one clock penalty that it adds to recognizing input signals. This is an option in specialized applications that have a high bandwidth requirement and the ability to guarantee signal integrity between the host and the printer. Otherwise, it is highly recommended that digital filtering be enabled.

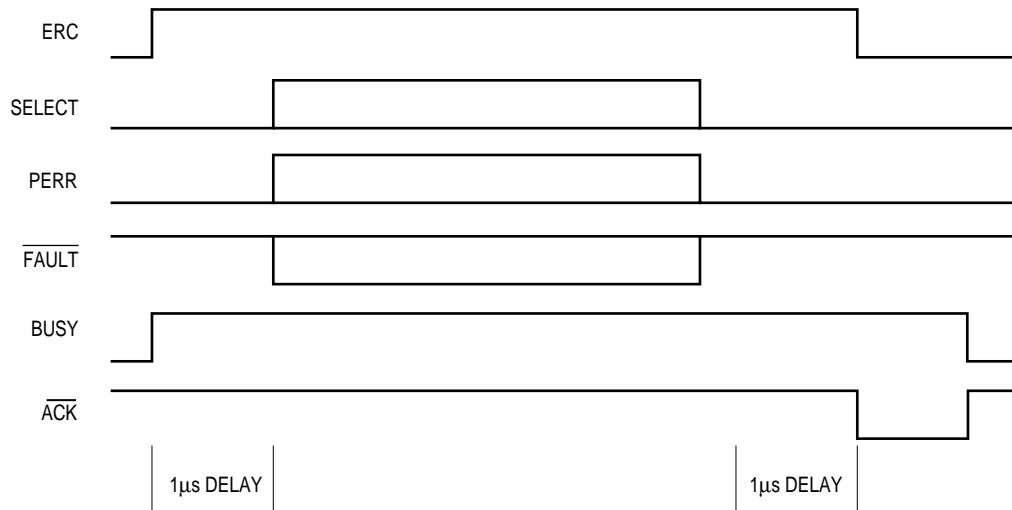
## 9.5 ERROR CYCLES

An example of an error cycle is when the user takes the printer off-line, a paper jam occurs, or the printer runs out of paper. When any of these or other events occur, the printer runs an error cycle to alert the host of a change in the operational status of the printer. An error cycle consists of asserting the BUSY signal and changing the states of SELECT, PERROR, and/or  $\overline{\text{FAULT}}$  to reflect the error condition. This is done by manipulating the ERC bit in the PPCR, which in turn triggers the PPI to set or clear bits in the PIER.

In compatibility mode, the software sets ERC to notify the PPI when an error cycle occurs. This causes the PPI to immediately set the BSY1 bit in the PIER. After 1  $\mu\text{s}$ , the software can set or clear the SEL, PER, or FLT bits in the PPIR, to indicate the error condition. After the error condition is cleared, the software returns the SEL, PRR, and FLT to their normal negated state. After 1  $\mu\text{s}$ , the software clears ERC, and the handshake logic concludes the cycle by generating an  $\overline{\text{ACK}}$  pulse and clearing BSY1.

If ERC is set and then  $\overline{\text{STROBE}}$  is received from the host, then the handshake logic performs the data transfer and data is still latched, but no acknowledge is generated until ERC is cleared. In other words, the ERC bit prevents the handshake logic from generating an  $\overline{\text{ACK}}$  pulse and clearing BSY1. Instead, ACK1 and BSY1 remain set in the PPIR and the transfer cycle is extended. As long as ERC remains set, BUSY remains high, and the software can manipulate status lines to indicate the error condition. After the error condition is cleared, the software returns the SEL, PER, and FLT bits in the PPIR to their normal negated state. After 1  $\mu\text{s}$  the software clears ERC, and the handshake logic concludes the cycle by generating an  $\overline{\text{ACK}}$  pulse and clearing BSY1.

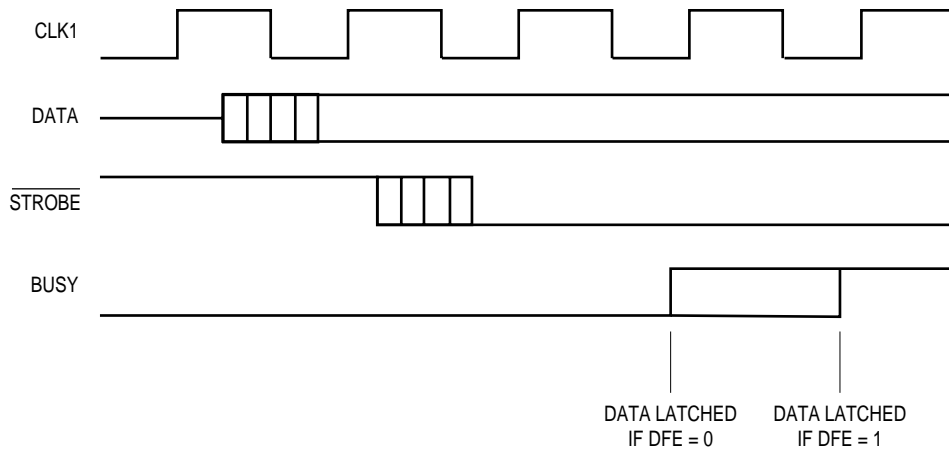
If ERC is set when the handshake logic is in the middle of a transfer, data is still latched, but no acknowledge is generated until ERC is cleared. In other words, ERC prevents the handshake logic from generating an  $\overline{\text{ACK}}$  pulse and clearing BSY1. If ERC happens to be set near the end of the cycle after the handshake logic has begun to generate an  $\overline{\text{ACK}}$  pulse, the hardware continues to produce the  $\overline{\text{ACK}}$  pulse as normal. Setting ERC does not affect an  $\overline{\text{ACK}}$  pulse that is already active, but does prevent an  $\overline{\text{ACK}}$  pulse that hasn't yet started (as described above). This behavior can result in two  $\overline{\text{ACK}}$  pulses being generated while BUSY is high. The first  $\overline{\text{ACK}}$  is generated in response to  $\overline{\text{STROBE}}$  and the second after ERC is cleared. Figure 9-7 illustrates the timing diagram for an error cycle.



**Figure 9-7. Error Cycle Timing Diagram**

### 9.6 PARALLEL PORT DATA BUS LATCHING

Latching is enabled only when the PPCR's PDE bit is clear. Latching occurs based on every high to low transition of  $\overline{\text{STROBE}}$ , regardless of the handshaking mode being enabled or disabled.  $\overline{\text{STROBE}}$ , like all PPI control inputs, is synchronized and optionally digitally filtered before it is used internally by the PPI. This affects the point at which parallel port data is latched. The 9-bit data input to the latch is not synchronized or digitally filtered. Figure 9-8 illustrates the timing diagram for parallel port data bus latching.



**Figure 9-8. Parallel Port Data Latch Timing Diagram**



## 9.7 PPI ON RESET

The assertion of  $\overline{\text{RESET}}$  causes all PPI registers and register fields to be cleared to zero (with one exception). The BSY2 bit in the PPIR is set and remains that way until the software explicitly clears it. This is intended to delay the host long enough for the software to initialize the PPI.

The RST bit in the PPCR resets the PPI state machine, which includes the hardware handshake controller and run-length decompression logic. Normally, when RST is set, MODE is set to 002 to prevent the state machine from starting again. RST immediately causes the handshake controller and the run-length decompression logic to return to idle. The handshake controller immediately discontinues handshaking with the host, terminates decompression, and any pending DMA request is removed. When the PPI state machine is in the idle state, BUSY and  $\overline{\text{ACK}}$  are negated. If BSY2 and ACK2 are clear, then BSY1 will be cleared and ACK1 will be set. RST also clears RLD and FLL bits in the PPCR. The software should issue RST under error conditions, such as when an invalid transition interrupt occurs or when the software detects a time-out error.

## 9.8 PPI DATA TRANSFER RATE

The expected data transfer rate in a system depends on several factors—capability of the host computer, memory speed, and bandwidth used. The block size of the data transfer affects the data rate because of the overhead involved. Forward data transfers are DMA based and reverse data transfers are interrupt based. Overhead cycles such as the negotiation phases are also interrupt based. The following are expected data transfer rates.

- Compatibility mode (forward) data rate = 400K/sec. Actually, the PPI is capable of a compatibility mode forward data rate of 2M/sec. The 400K/sec is determined by the IEEE 1284 specification.
- ECP mode (forward) data rate = 2M/sec.
- ECP decompression (forward) data rate = 4M/sec.
- Nibble mode (reverse) data rate = 5K/sec.
- Byte mode (reverse) data rate = 10K/sec.

## SECTION 10

# PRINT ENGINE INTERFACE

The MC68322 uses an 8-bit, half-duplex, synchronous serial protocol to communicate with the print engine. This print engine interface contains an integrated print engine video controller (PVC) as well as control logic for the 8-bit serial communication channel compatible with many print engines. The print engine interface is one of the two components of the graphics unit; the other being the RISC graphics processor.

The PVC can drive virtually all laser printers currently on the market. It automatically retrieves video data from DRAM using burst cycles and manages all parameters associated with the video image. This type of print engine video interface eliminates all the software overhead associated with the transfer of the bitmap image to the print engine. The software starts the transfer and waits for the end of page or band interrupt event, which allows the core to start processing the next page. The PVC also saves a significant amount of hardware cost. Static RAMs and FIFOs are unnecessary since the bitmap image is transferred directly from DRAM, serialized, and then transmitted to the print engine. The PVC uses an efficient page mode access to DRAM to fill its internal eight word queue (FIFO).

The PVC is divided into two subsystems:

- A memory subsystem that performs direct memory accesses to DRAM to obtain video data from a page or band bitmap.
- A video subsystem that serializes the data and handshakes with the printer to transmit the video data.

The memory subsystem contains a data fetch controller, an 8-word deep FIFO, and many counters. It interfaces to the core through a set of memory-mapped registers called the printer control block. The video subsystem's task is to interface with the print engine and serialize and transmit video data. It contains a video interface controller, a 16-bit shift register, a phase-locked loop (PLL) clock circuit, and some counters.

The PLL video clock divisor divides the VCLK input by 32, 24, 16, 12, 8, or 4 to produce the dot clock for the PVC. This is accomplished internally by using a prescaler and by using either the rising edge or both edges of the clock for the PLL. This feature allows for direct support of multiple resolutions using a single external oscillator. For example, the PVC can transmit  $300 \times 300$ ,  $600 \times 300$ ,  $600 \times 600$ , or  $1200 \times 600$  resolution page images using the same VCLK input frequency.

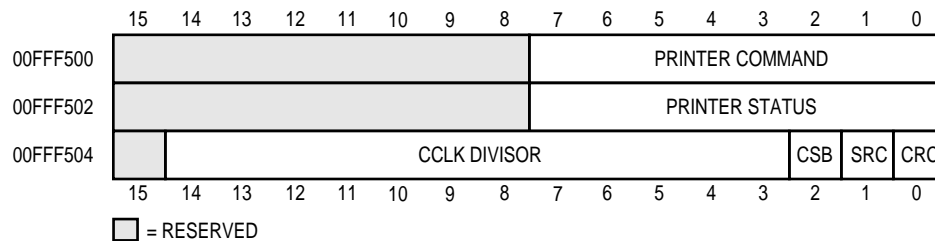
The print engine interface is capable of sending and receiving information to and from a print engine, which is done at the same time as all other operations within the MC68322. This transmission and reception of information is synchronous either to the clock supplied by the interface or to the clock supplied by the print engine. For transmitting video data, the PVC can run synchronously from a 1× video rate from the print engine. If the print engine does not supply a video clock, the PVC can phase lock to a print engine's synchronization signal and generate an internal video rate to use during transmission. The phase-locked loop circuitry runs off of a clock from an external oscillator.

## 10.1 PRINT ENGINE INTERFACE REGISTERS

The print engine interface contains several registers that control the engine interface protocol, signals, and interrupts. These registers include the printer communication register, the PVC control register, the printer control block register set, the PVC interrupt event register, and the printer communication interrupt event register.

### 10.1.1 Printer Communication Register

The printer communication register (PCOMR) contains several bits and fields that control the status of data transfers to or from the print engine.



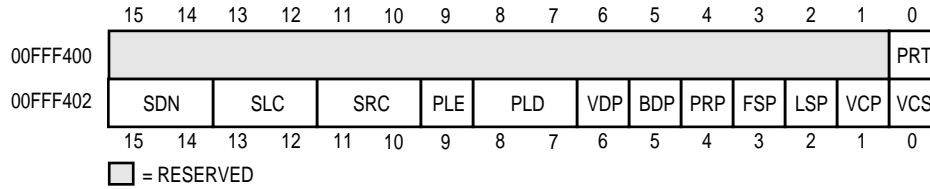
**Figure 10-1. Printer Communication Register**

The core uses the printer command field (write-only) to send an 8-bit command to the print engine and the printer status field (read-only) receives an 8-bit serial command. The 12-bit CCLK divisor field determines the period of the clock supplied to the print engine.  $CCLK \text{ period} = CLK1 \text{ period} \times (CCLK \text{ Divisor} + 1) \times 16$ . At 16MHz, this provides a range of 1µs to 4.1ms per CCLK period.

The PCOMR has three bits that determine the various modes of operation—the  $\overline{CMD/STS}$  bidirectional (CSB) bit, the  $\overline{SCLK}$  source (SRC) bit, and the  $\overline{CCLK}$  source (CRC) bit. The CSB bit indicates whether  $\overline{CMD/STS}$  is bidirectional (CSB=1) or output only (CSB=0). The values programmed in the SRC and CRC bits determine whether the MC68322 (SRC, CRC=1) or the print engine (SRC, CRC=0) have supplied status or command clocks, respectively.

## 10.1.2 PVC Control Register

The PVC control register (PVCCR) contains bits that control the polarity of all MC68322 inputs and outputs used by the PVC. The PVCCR should be written after an MC68322 reset and before a print operation begins.



**Figure 10-2. PVC Control Register**

While the PVC is active, the polarity of input signals should not be altered. The PVC responds to a polarity change if it occurs while the PVC is waiting for a transition, but this should be avoided. The PVCCR also contains the video clock source control bits, the PLL video clock divisor control, and the SmartToner control bits.

### SDN—SmartToner Density

This field selects one of four halftone patterns, where  $00_2 = 1/2$  density,  $01_2 = 1/4$  density,  $10_2 = 1/6$  density, and  $11_2 = 1/8$  density. This will affect the amount of toner used and the quality of the printed page when the SmartToner feature is enabled.

### SLC and SRC—SmartToner Left and Right Edge Control

These fields are used to select which dot the halftone pattern is applied to on the left and right edges of a character. If the halftone pattern starts and ends on the first dot, then the character will appear more jagged than if it starts and ends on the second or third dot. For example, if the halftone pattern starts and ends on the second dot, then the character will have a one dot wide border on the left and right sides, with the halftone pattern applied in between.

**Table 10-1. SLC and SRC Encodings**

ENCODING	HALFTONE PATTERN
00	1st Dot
01	2nd Dot
10	3rd Dot
11	Not Used

### PLE—PLL Edge

This bit is set to clock on both edges and clear to clock on rising edge only.

### PLD—PLL Divisor

This field controls the PLL video clock prescaler. A PLD value of  $00_2$ ,  $01_2$ ,  $10_2$ , or  $11_2$  causes the VCLK input to prescale by one, two, three, or four before clocking the  $8\times$  PLL.

VDP— $\overline{\text{VIDEO}}$  Polarity Control

Set if  $\overline{\text{VIDEO}}$  is active high; clear if active low.

BDP—Border Polarity Control

Set to drive low logic level in a nonprinting area; clear to drive high.

PRP— $\overline{\text{PRINT}}$  Polarity Control

Set if  $\overline{\text{PRINT}}$  is active high; clear if active low.

FSP— $\overline{\text{FSYNC}}$  Polarity Control

Set if  $\overline{\text{FSYNC}}$  is active high; clear if active low.

LSP— $\overline{\text{LSYNC}}$  Polarity Control

Set if  $\overline{\text{LSYNC}}$  is active high; clear if active low.

VCP—VCLK Polarity Control

Set if rising edge is to be used; clear if falling edge is to be used.

VCS—VCLK Source

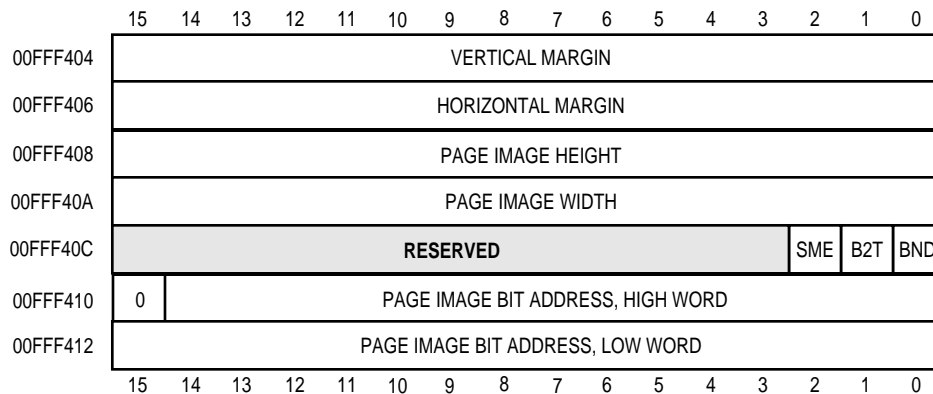
Clear if VCLK input is 1× clock; set if VCLK input is an external oscillator driving the 8× PLL.

PRT— $\overline{\text{PRINT}}$

This bit controls the level of the MC68322's  $\overline{\text{PRINT}}$  output pin. When the PRT bit is set,  $\overline{\text{PRINT}}$  is asserted, but when it is clear,  $\overline{\text{PRINT}}$  is negated. The active level of  $\overline{\text{PRINT}}$  is determined by the PRP bit. In nonprinting applications or for applications that do not require a  $\overline{\text{PRINT}}$  interface signal, the  $\overline{\text{PRINT}}$  output pin can be used as a general-purpose output.

### 10.1.3 Printer Control Block Register Set

The printer control block (PCB) register set is a group of six registers that define a print operation for the PVC to execute. The PCB values define the dimensions and location of the band image, page margins, and band control data. Loading the page image bit address portion of the PCB is the stimulus that starts the PVC and begins a print operation. The entire PCB register set is doubled buffered, which allows two print operations to be loaded at a time. Basically, when the first operation finishes, the second operation immediately begins executing. This is especially useful in banding applications.



**Figure 10-3. Printer Control Block Register Set**

The PCB's vertical and horizontal margin registers position the page image on the printed page. The vertical margin value specifies the number of scanlines that are skipped before the first scanline of page image data and the horizontal margin value defines the number of internal video clocks that are skipped before the first bit is transmitted at the start of each scanline.

The PCB's page image height and width registers describe the limits of the page image as it is to be read from memory and transmitted to the print engine. The page image height is measured in scanlines, but the page image width is in bits. The PCB also contains three 1-bit fields that control the execution of a print operation on a band-by-band basis.

#### BND—Band

This bit is used during banding applications to indicate when the current band image is to be followed by another band on a page. It is set for all bands, but cleared for the last band in a sequence or when only one band appears on a page.

#### B2T—Bottom to Top

This bit is set to indicate the render direction. When B2T is clear, it indicates a top to bottom and left to right direction. When set, it indicates a bottom to top and right to left direction.

#### SME—SmartToner Enable

This bit is used to reduce the amount of toner the printer uses. When it is enabled, a halftone pattern is applied to the video data sent to the print engine. The print image in memory is not affected. The SME bit is double buffered and can be set or cleared for a specific page.

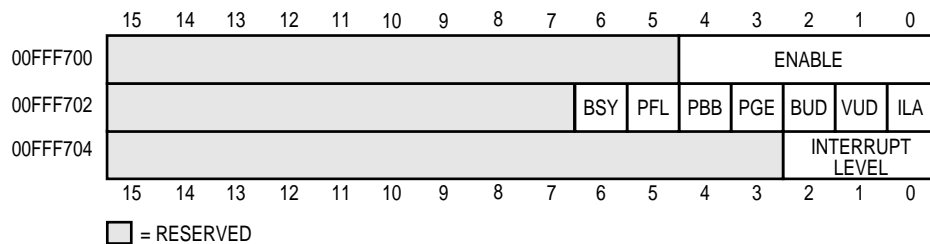


**Note:** It is not recommended that you change the SmartToner options in the PVCCR (SLC, SRC, and SDN bits) while printing a page.

The page image bit address fields hold the starting address of the band image to be printed. This address is a bit address and points to the first bit of data to be transmitted to the print engine. Loading the page image bit address fields is also the stimulus that starts the PVC and initiates a print operation. The values of all other PCB registers must be set before the page image bit address register is loaded.

### 10.1.4 PVC Interrupt Event Register

The PVC interrupt event register (PVCIR) contains five bits that indicate PVC interrupts events, two that report normal events (band begin and page end), and three that report errors (video under, band under, and PVC error). A PVC error interrupt event results from an illegal address or out-of-range address. If an error is detected by the PVC during the processing of the PCB register set, it sets one of these five interrupt sources in the PVCIR. After any of these interrupts occur, the PVC's operation is undefined and must be immediately reset by the module soft reset register's (MSRR) PVC bit to return it to normal operation.



**Figure 10-4. PVC Interrupt Event Register**



**Note:** The PVCIR is located in the interrupt register portion of the memory map and, therefore, is not necessarily located contiguously with the other print engine interface registers.

#### BSY—PVC Busy

This bit indicates when the PVC is executing a print operation.

#### PFL—PCB Full

This bit indicates when the PCB register set is available for a new print operation. The PCB registers should only be loaded when the PFL bit is clear. When the PFL bit is set, the PCB register set should not be altered.

**PBB—Page/Band Begin**

This bit is posted at the start of every page or band. At the start of a page, PBB is set after it receives  $\overline{\text{FSYNC}}$  from the print engine and provides timing information to the software for controlling the PVCCR's PRT bit (typically, the timing of a print engine's  $\overline{\text{PRINT}}$  signal is referenced to  $\overline{\text{FSYNC}}$ ). When starting a band in the middle of a page, a page/band begin interrupt event is posted when the first page image data is read from memory.

**PGE—Page End**

This bit is issued at the bottom of every page (end-of-page is determined by the BND bit in the PCB control field) immediately after the last word of the last scanline is read from memory. In response to a page end interrupt event, the page or band buffer can be reclaimed and reused for the next page.

**BUD—Band Underrun**

This bit indicates a banding error, which is posted when the last word of one band is read, but the PCB register set (specifically, the page image bit address) for the next band is not yet loaded. Band underruns usually point to a page description that is too complicated for the given banding environment. This bit is only set when the BND bit in the PCB control field is set, thus indicating that the image data is a band.

**VUD—Video Underrun**

This bit indicates that a video underrun has occurred. It is posted if the print engine requests video data faster than the PVC can access memory. A video underrun is usually the result of a band underrun or PVC error, but can also mean the memory configuration is too slow for the video rate of the print engine.

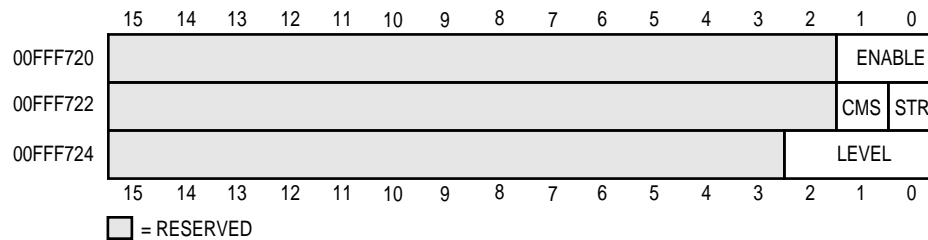
**ILA—Illegal Address**

This bit indicates the PVC's attempt to read from DRAM memory that is nonexistent because of an out-of-range address. The out-of-range address occurs when reading page image data and can be the result of loading an illegal memory address in the page image bit address register or reading beyond the end of memory space. The PVC shuts down in response to an out-of-range address. In this state, the PVC only responds to a PVC soft-reset that is required to return the PVC to normal operation.



## 10.1.5 Printer Communication Interrupt Event Register

The printer communication interrupt event register (PCIER) indicates when a serial command has been sent or a serial status has been received.



**Figure 10-5. Printer Communication Interrupt Event Register**



**Note:** The PCIER is located in the interrupt register portion of the memory map and, therefore, is not necessarily located contiguously with the other print engine interface registers.

The command sent (CMS) bit indicates that the PCOMR's printer command field has been written and that the 8 bits of command data have been sent to the print engine. This interrupt event bit notifies the core that the command has been sent.

The status receive (STR) bit indicates that a status received interrupt event is active. This occurs when the interface receives one byte of status data and the  $\overline{\text{SBSY}}$  pin is deactivated. STR is set when the  $\overline{\text{SBSY}}$  pin is inactive to prevent a command operation from occurring before the print engine is ready to receive it.

## 10.2 PRINTER COMMUNICATION PROTOCOL

The print engine interface uses the  $\overline{\text{CBSY}}$  and  $\overline{\text{SBSY}}$  pins to indicate the direction of data transfer and it uses the CCLK pin to pace data transmissions. It does not employ handshaking, but it asserts the  $\overline{\text{CBSY}}$  and  $\overline{\text{SBSY}}$  pins before the actual data transmission so there is sufficient time for the logic to prepare for the subsequent data. CCLK remains inactive until either the  $\overline{\text{CBSY}}$  or  $\overline{\text{SBSY}}$  pin is asserted and then goes through eight periods, one per data bit. During a command transfer to the print engine, the MC68322 shifts a bit on each CCLK falling edge and expects the print engine to sample the datastream on each rising edge. Similarly, for a status transfer from the print engine, the MC68322 samples the datastream on each rising edge and expects the print engine to shift a bit on each falling edge.

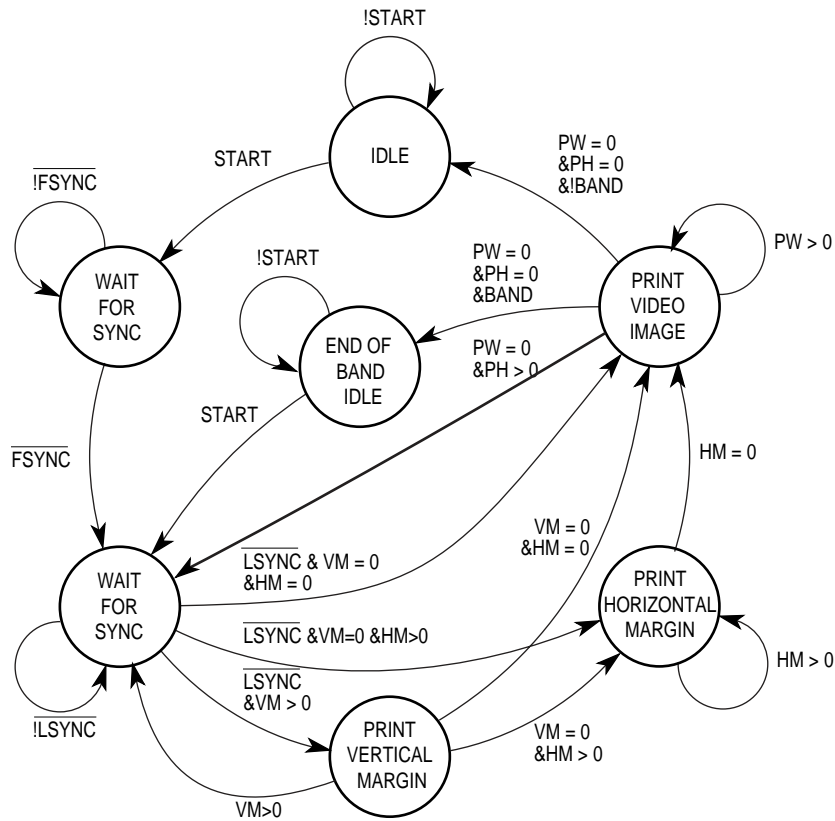
### 10.3 PRINT ENGINE INTERFACE OPERATION

Page images are generally rendered by the RISC graphics processor (RGP) into an area of memory known as the band buffer. After a page image is rendered, the PVC is programmed to transmit the contents of the band buffer to the print engine. The PVC starts by loading the PCB register set with the dimension and location of the page image. Once the page image bit address register is written, the memory subsystem sets the PVCCR's PFL and BSY bits and begins to fetch data from memory to fill its FIFO. After the video subsystem is started, which is also started when the page image bit address register is written, PFL is cleared.

Next, the video interface controller begins waiting for  $\overline{\text{FSYNC}}$  and once it arrives, a page/band begin interrupt event is posted and the controller waits for the leading active edge of  $\overline{\text{LSYNC}}$ . Then after  $\overline{\text{LSYNC}}$  arrives, the vertical margin count is decremented and the controller again waits for  $\overline{\text{LSYNC}}$ . When the vertical margin decrements to zero, the next  $\overline{\text{LSYNC}}$  causes the horizontal margin count to be decremented for each internal video clock until it reaches zero. At this time, the first bit of video data is transmitted.

Video data is loaded from the memory subsystem's FIFO into a 16-bit shift register. On each active edge VCLK, data is shifted out. When the last data bit is shifted out, the shift register is reloaded. As the video subsystem shifts out video data and empties the FIFO, the memory subsystem makes additional memory fetches. The FIFO is filled until the page image height and width values decrement to zero, at which time BSY is cleared and the memory subsystem returns to idle. Every internal video clock decrements the page width counter. When the page width counter reaches zero, the page height counter is decremented and, if it is not zero, the video interface controller returns again to wait for  $\overline{\text{LSYNC}}$ .

When the page width and page height reach zero, the transmission completes and the video interface controller posts a page end interrupt event and returns to idle. If the BND bit was set in the PCB's band control register, the controller returns instead to end-of-band idle and waits for another page address to load before resuming execution. A state diagram for the video interface controller machine is illustrated in Figure 10-6.



## NOTES:

$\overline{\text{FSYNC}}$  =  $\overline{\text{FSYNC}}$  Asserted  
 HM = Horizontal Margin Register Value  
 LSYNC = Leading Edge of LSYNC Detected  
 PH = Page Image Height Register Value  
 PM = Page Image Width Register Value  
 START = Load of Page Image Bit Address Register  
 VM = Vertical Margin Register Value

**Figure 10-6. PVC Video Interface State Diagram**

### 10.3.1 Synchronous/Asynchronous PVC Operation

The PVC interfaces with print engines that provide either an asynchronous or synchronous interface. A print engine interface is considered synchronous if it supplies a video clock and a synchronous control signal. If a print engine supplies no clock, the interface is considered asynchronous. The PVC is programmed for synchronous operation by clearing the PVCCR's VCS bit. This causes the PVC to use VCLK as a 1× clock source to clock video data. The VCLK period defines the width of a video dot. The active edge of VCLK is programmed by setting the PVCCR's VCP bit.

A synchronous print engine interface must source VCLK at a frequency that equals the expected video dot rate and supply an  $\overline{\text{LSYNC}}$  signal that meets the setup and hold requirements of the MC68322 relative to the VCLK source. However, VCLK must be a free-running clock.

The PVC is programmed for asynchronous operation by setting the PVCCR's VCS bit. This causes the PVC to use the VCLK as a clock source and to enable the internal PLL circuitry. The VCLK input is either 4, 8, 12, 16, 24, or 32 times the frequency of the video data rate the print engine requires. The PLL circuit continuously generates an internal 1× clock to drive the video subsystem. When a print operation is active and  $\overline{\text{LSYNC}}$  arrives, the PLL adjusts the frequency of the internal 1× clock in synchronization with  $\overline{\text{LSYNC}}$ 's arrival. The PLL guarantees that each scanline starts at the same point on the page with a maximum offset of an eighth of a dot.

The PLL always takes a 1-dot clock delay to synchronize with  $\overline{\text{LSYNC}}$ . Depending on the PLL prescaler value selected, this will be a fixed time in the range of 4 to 32 VCLK periods. A horizontal margin of zero dots is allowed. Asynchronous interfaces need only provide an  $\overline{\text{LSYNC}}$  signal that accurately identifies the start of each scanline. And, again, the VCLK source must be a free-running clock.

On reset, the print engine interface is programmed for synchronous operation. If you want asynchronous operation with the PLL, the PVCCR's VCS bit must be set. After changing the VCS bit, a PVC reset interrupt event must be posted by setting the PVC bit in the MSRR. This will reset the video state machines to a known state. The VCS bit should be changed only between pages when the video state machines are inactive.

### 10.3.2 Command Operation

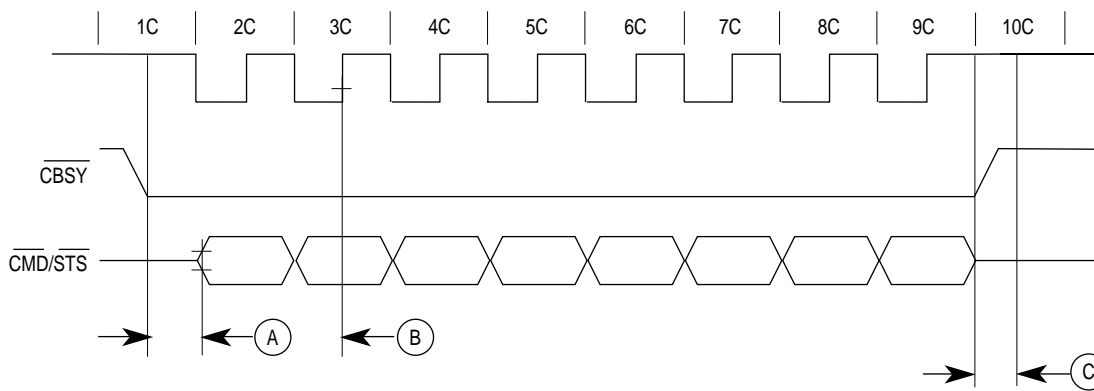
Commands are sent to the print engine based on CCLK. Writing another command to the PCOMR printer command field before one is finished should be avoided because it will corrupt the data. The PCOMR should be properly programmed to choose between the command and status modes.  $\overline{\text{SBSY}}$  and  $\overline{\text{STS}}$ , which are used during status operations, should not be asserted during a command operation.

**10.3.2.1 CCLK SUPPLIED BY MC68322.** In this mode, a write to the PCOMR's printer command field makes  $\overline{\text{CSY}}$  active.  $\overline{\text{CSY}}$  completely brackets a command transmission by providing a setup and hold of one half CCLK period each. This allows sufficient time for the print engine to detect the impending command byte and prepare its internal logic. Setting the PCOMR's CRC bit allows the MC68322 to supply CCLK. The value of the PCOMR's CCLK divisor field should be programmed to provide sufficient setup and hold time for the command data, with respect to the rising edge of the CCLK.

Command data is transferred on  $\overline{\text{CMD/STS}}$ , which remains in high impedance until the  $\overline{\text{CSY}}$  setup time is satisfied.  $\overline{\text{CMD/STS}}$  is then brought active (for one CLK1 period) and each bit is driven on the falling edge of CCLK. The print engine should sample the command data on the rising edge of CCLK. At the end of the transmission,  $\overline{\text{CMD/STS}}$  is brought high for one CLK1 period and then returns to the high impedance state.

When  $\overline{\text{CSY}}$  transitions to an inactive state (indicating the end of the command operation), the PCIER's CMS bit is set to notify the core that the command has been sent, thus causing a command sent interrupt event to occur. The software should not try to write another command until this interrupt is received.

A recovery cycle is then initiated, during which no command or status operations are performed. If a subsequent command operation is initiated during this time, it will be latched and started after the recovery. If  $\overline{\text{SBSY}}$  becomes active, indicating a status operation, then it must remain active until the end of the recovery, which has the duration of one half CCLK period. Figure 10-7 illustrates the timing diagram for a command operation when the core supplies CCLK.



## NOTES:

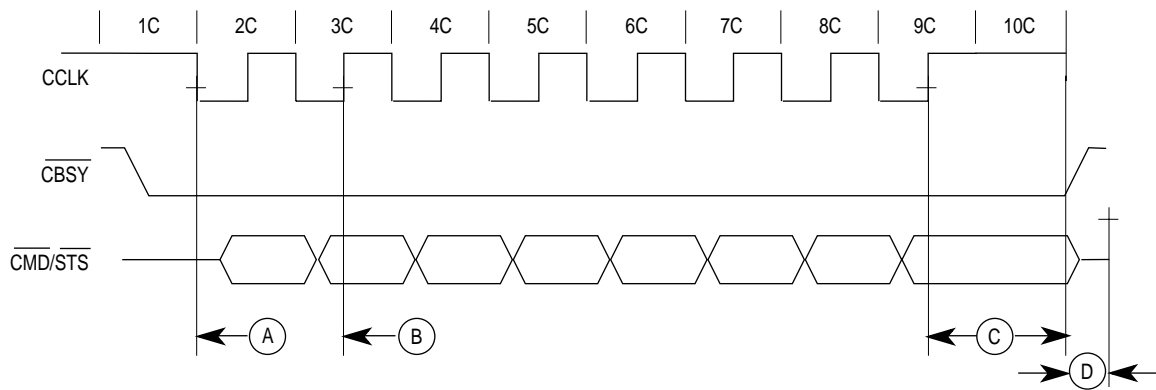
- $\overline{\text{SBSY}}$  and  $\overline{\text{STS}}$  are not sampled during command operations.
- A = CBSY setup; half CCLK period.
- B = CMD sampled by the print engine.
- C = Recovery cycle; half CCLK period.

**Figure 10-7. Command Operation—MC68322 Supplies CCLK**

**10.3.2.2 CCLK SUPPLIED BY PRINT ENGINE.** In this mode, the print engine interface expects eight rising edge transitions of CCLK from the print engine. Because CCLK is asynchronous to CLK1, it is synchronized inside the MC68322. As a result, the period of this signal can be no less than four CLK1 periods. When a command operation is initiated,  $\overline{\text{CBSY}}$  transitions to an active state and stays active until all command bits are transferred and hold time (the value in the PCOMR's CCLK divisor field) for the last command bit is satisfied.

Command data is transferred on  $\overline{\text{CMD/STS}}$ , which remains in high impedance until the first falling edge of CCLK. The command bit changes three CLK1 periods after every falling edge of the CCLK. At the end of the transmission,  $\overline{\text{CMD/STS}}$  is brought high for one CLK1 period and then returns to the high impedance state. When  $\overline{\text{CBSY}}$  transitions to an inactive state (indicating the end of the command operation), the PCIER's CMS bit is set to notify the core that the command has been sent, thus causing a command sent interrupt event to occur. The software should not try to write another command until this interrupt is received.

$\overline{\text{SBSY}}$  and  $\overline{\text{STS}}$ , which are used during status operations, should not be asserted during a command operation. Figure 10-8 illustrates the timing diagram for a command operation when the print engine supplies CCLK.



## NOTES:

$\overline{\text{SBSY}}$  and  $\overline{\text{STS}}$  are not sampled during command operations.

A =  $\overline{\text{CMD}}$  transmitted with the first falling edge of CCLK.

B =  $\overline{\text{CMD}}$  sampled by the print engine.

C =  $\overline{\text{CMD/STS}}$  hold; controlled by the CCLK divisor field.

D = Recovery cycle; half CCLK period. Any print engine response is acknowledged after recovery cycle.

**Figure 10-8. Command Operation—Print Engine Supplies CCLK**

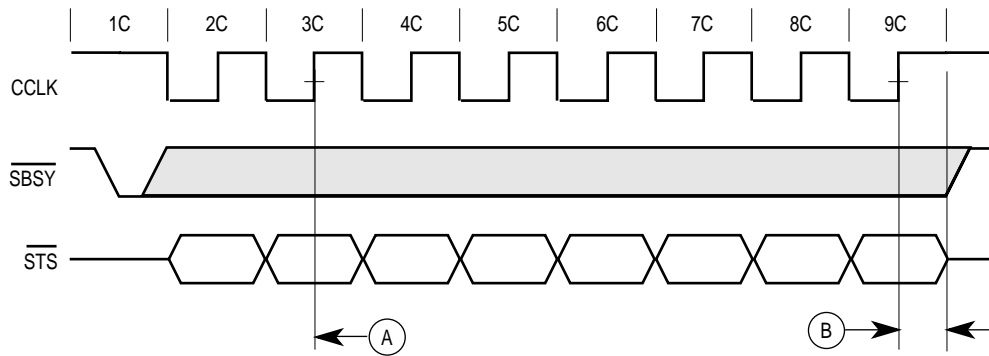
### 10.3.3 Status Operation

CCLK status information is sent to the MC68322 from the print engine. During the status operation, serial data is assembled into the PCOMR's printer status field. Status information can be supplied to the MC68322 on either the  $\overline{\text{STS}}$  or  $\overline{\text{CMD/STS}}$  signals. If status information is driven on  $\overline{\text{STS}}$ , the  $\overline{\text{CMD/STS}}$  signal will be in a high impedance state.  $\overline{\text{CBSY}}$  is in a high impedance state during the status operation.

**10.3.3.1 CCLK SUPPLIED BY MC68322.** In this mode, the MC68322 print engine interface generates eight high to low transitions of CCLK so that the status data can be sampled at every rising edge. The PICR's CCLK divisor field must be programmed to provide sufficient setup and hold time for the status data to be sampled.

After all eight bits of the status are sampled, the print engine interface enters a recovery cycle. The recovery time is one half CCLK period as programmed in the PCOMR's CCLK divisor field. During the recovery time,  $\overline{\text{SBSY}}$  is not sampled and if a command operation is initiated it will be latched and executed after the recovery. When the status operation is complete, the PCIER's STR bit is set, thus indicating that the PCOMR's printer status field is full. If enabled, this bit can and will cause an interrupt event.

$\overline{\text{SBSY}}$  is an asynchronous signal to the print engine interface and is internally synchronized to CLK1. The print engine should not assert the signal at any time during a command operation or recovery cycle. Once  $\overline{\text{SBSY}}$  has been asserted, it must remain that way until the status operation is initiated. Figure 10-9 illustrates the timing diagram for a status operation when the core supplies CCLK.



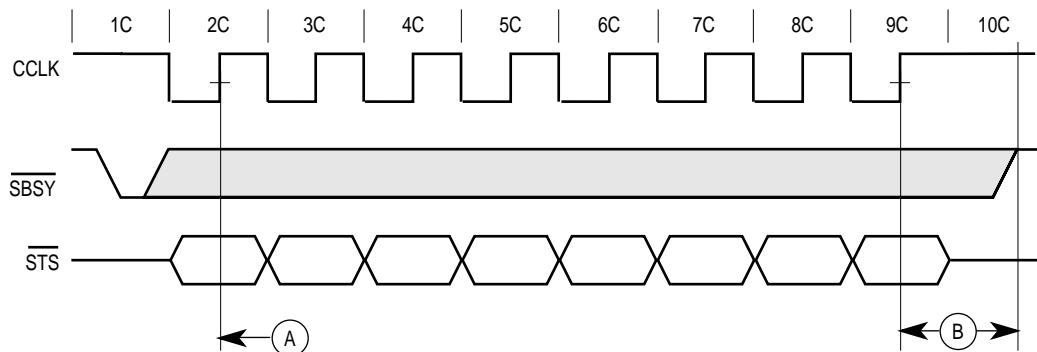
NOTES:

- CBSY and CMD/STS are in high impedance.
- A = STS is sampled by the EC000 core.
- B = Recovery cycle; half CCLK period. Any pending command is acknowledged after the recovery.

**Figure 10-9. Status Operation—MC68322 Supplied CCLK**

**10.3.3.2 CCLK SUPPLIED BY PRINT ENGINE.** In this mode, the print engine is expected to supply eight high to low transitions of CCLK so that the status can be sampled on every rising edge. The CCLK period must be at least four CLK1s for proper sampling of each status bit. After all eight bits of the status are sampled, the print engine interface enters a recovery cycle. The recovery time is one half CCLK period as programmed in the PCOMR's CCLK divisor field. When the status operation is complete, the PCIER's STR bit is set, thus indicating that the PCOMR's printer status field is full. If enabled, this bit causes an interrupt event.

SSBY is an asynchronous signal to the print engine interface and is internally synchronized to CLK1. The print engine should not assert the signal at any time during a command operation or recovery cycle. Once SSBY has been asserted, it must remain that way until the status operation is initiated. Figure 10-10 illustrates the timing diagram for a status operation when the print engine supplies CCLK.



NOTE: CBSY and CMD/STS are in high impedance.

- A = STS is sampled by the EC000 core.
- B = Recovery cycle half CCLK period. Any pending command is acknowledged after the recovery.

**Figure 10-10. Status Operation—Print Engine Supplied CCLK**

### 10.3.4 PLL Video Clock Divisor

The 8× PLL circuitry can clock on either the rising edge or both edges of its clock. This is controlled by the PVCCR's PLE bit. If the PLL clocks on the rising edge, it divides its input clock by eight to produce a dot clock. This dot clock will have no more than 1/8 dot jitter. If the PLL clocks on both edges, it divides its input clock by four to produce a dot clock. When PLD = 00 or 10, and PLE = 01, the dot clock has no more than 1/7 dot jitter. When the duty cycle of the VCLK is less than 50%, the amount of jitter will increase when using both clock edges. The duty cycle does not adversely affect the jitter of the dot clock when using the rising edge only mode. PLD and PLE have an effect only when the PVCCR's VCS bit selects the PLL clock. After changing the PLD or PLE bits, a PVC soft-reset must be posted. The PLD and PLE bits should only be changed between pages when the video state machine is inactive. Table 10-2 lists all values of PLD, PLE, and the resulting dot clock.

**Table 10-2. PLL Video Clock Divisor**

PLD ENCODING	PLE ENCODING	PRESCALE VALUE	CLOCK EDGE	DOT CLOCK	CLOCK JITTER
0	0 1	+1	Rising Both	VCLK ÷ 8 VCLK ÷ 4	1/8 Dot 1/7 Dot
1	0 1	+2	Rising Both	VCLK ÷ 16 VCLK ÷ 8	1/8 Dot 1/8 Dot
2	0 1	+3	Rising Both	VCLK ÷ 24 VCLK ÷ 12	1/8 Dot 1/7 Dot
3	0 1	+4	Rising Both	VCLK ÷ 32 VCLK ÷ 16	1/8 Dot 1/8 Dot

NOTE: ClockJitter When PLLDIV = 0 And PLL EDGE = 1 Is Dependent On VCLK Input Clock Symmetry.

In some cases there are multiple configurations that produce the same effective dot clock. It is preferable to use the rising edge of the clock when there is a choice because it will produce slightly less jitter.

The prescaler can be set to divide by three. This is useful in a situation where only one resolution is required because this would allow a common input clock for both VCLK and CLK2. For example, if an 8 ppm engine at 300 × 300 dpi requires a dot clock of 2.5 MHz, then VCLK could be 30 MHz. This 30-MHz clock could also be used for the processor clock (CLK2) so that an oscillator could be eliminated from the board.



## 10.4 PVC ON RESET

There are two sources for a PVC reset interrupt event—the MC68322  $\overline{\text{RESET}}$  pin and the MSRR's PVC bit. The  $\overline{\text{RESET}}$  pin resets the entire device while the PVC soft-reset only resets the PVC and PLL logic. The state of the PVC after each type of reset is similar in that:

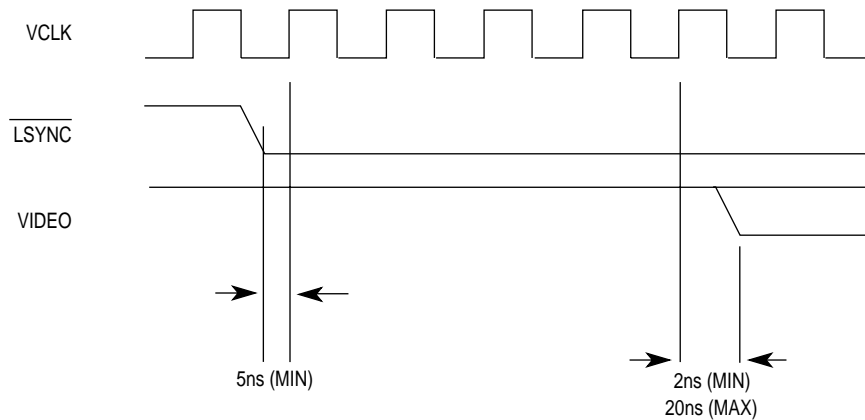
- The PLL is reset
- All PVC controllers return to idle
- The PVCIR's PFL and BSY bits are cleared

There are two important differences during a MC68322 reset—all PVC registers are cleared and any pending PVC interrupts are cleared. Following the assertion of the RESET input, all PVC registers are cleared to zeros. This causes the PVC to assume a default interface. The default interface uses a 1× clock, interprets all inputs, and drives all outputs as active low signals. If this interface is not correct for the specific print engine, the PVCCR must be written to define a different interface. If the PVCCR's VCS bit changes, then the PVC will need to be soft-reset.

During a PVC soft-reset interrupt event, no PVC registers are affected. Any pending print operation is cleared and the next print operation is assumed to start at the beginning of a page. Any pending interrupts remain pending. A PVC reset interrupt event only occurs after the conclusion of the PVC's on-going memory cycle.

## 10.5 PVC VIDEO DATA TIMING

The following timing diagram illustrates the relationship between the video data and the assertion of the  $\overline{\text{LSYNC}}$  signal when the horizontal margin is set to zero.



There are many other variables associated with the behavior of the PVC. The above timing diagram corresponds to a system configuration in the following manner.

### 10.5.1 1X Video Clock(PVCCR Bit 0 = 0)

If the PLL is used, then video data will be clocked out by an “effective video clock” that is generated internal to the MC68322. Recognition of the  $\overline{\text{LSYNC}}$  signal and the subsequent output of video data is relative to this internal clock since it is located outside the chip. For applications that use an external ASIC to post-process the video data, use the 1X clock mode.

### 10.5.2 VCLK Rising Edge (PVCCR Bit 1 = 11)

If VCP = 0, the specifications shown above are relative to the falling edge of VCLK.

### 10.5.3 Border Polarity High (PVCCR Bit 5 = 0)

This bit controls the state of the video out signal in the “margin” area of the page. If this bit is set, the video signal is driven low when the video data is not being driven out of the part.

# SECTION 11

## RISC GRAPHICS PROCESSOR

The RISC graphics processor (RGP) is one of the two components of the graphics unit; the other being the print engine interface. The RGP executes a display list of graphic orders (a special list of instructions) to render a page image. This section describes the following RGP functional blocks and how they operate.

- Graphic order parser
- Graphic order execution unit
- Band registers
- Arithmetic logic unit
- Pixel data files
- Boolean logic unit

The graphic order parser is responsible for reading the display list and determining opcode type and, if a banded opcode, the appropriate band to process the graphic order. It is also responsible for aligning and transferring the operands into the band registers, interpreting, aligning, and transferring scanline table bit string specifiers into the band registers, as well as writing back any modified operands in the event of a band fault.

The graphic order execution unit is a collection of state machines that control all aspects of the graphic operations, including calculation of effective addresses, rectangular pixel array widths in words, barrel shifter rotation offsets, left/right mask values, and halftone positioning. The band registers hold the graphic order operands and values that are required for internal use. The arithmetic logic unit performs all internal calculations including addition, subtraction, increment by one, decrement by one, and comparison. It is also used to calculate and modify effective addresses for source, destination, and halftone bit maps.

The pixel data files (along with internal barrel shifters) hold source, halftone, and destination data, and it constantly adjusts so that all the data is uniformly aligned. These data files operate independently from the remainder of the RGP, requesting new data at the same time existing data is processed through the barrel shifters or Boolean logic unit for subsequent writing to memory.

The Boolean logic unit carries out all combinations of 1-, 2-, and 3-operand Boolean operations, combining source, halftone, and destination data together in one of 256 possible combinations. Left and right masking is used to preserve the original destination data at the left and right edges of the destination bitmap.

## 11.1 RGP REGISTERS

There are three registers that provide control of and receive status from the RGP—the RGP start register, the RGP diagnostic register, and the RGP interrupt event register.

### 11.1.1 RGP Start Register

The RGP starts by writing the address of a display list into the RGP start register (RSR). When the RSR is loaded, the first graphic order is fetched. The RSR in the MC68322 is double buffered, thus allowing a second display list address to be loaded into the RSR prior to the completion of the first display list. The second display list is automatically started as soon as the first one finishes.

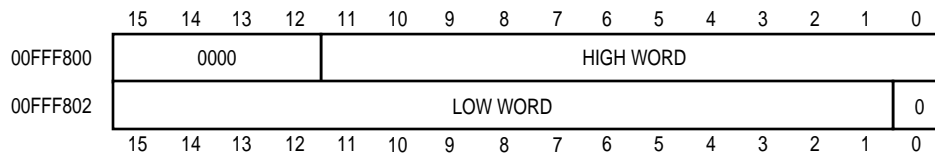


Figure 11-1. RGP Start Register

### 11.1.2 RGP Diagnostic Register

The RGP diagnostic register (RDR) contains the byte address of the graphic order opcode or operand being interpreted from the display list. It is frozen when an error occurs and the RGP interrupt event register’s RER bit is set. Depending on the nature of the error, the RDR can be frozen beyond the exact point in the display list where an RGP error occurs because of the prefetch characteristics of the RGP. The RDR resumes functioning when a soft-reset of the RGP occurs.

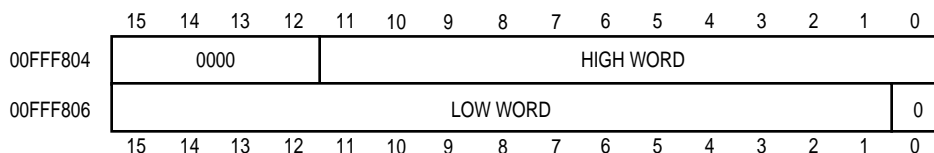


Figure 11-2. RGP Diagnostic Register

### 11.1.3 RGP Interrupt Event Register

The RGP interrupt event register (RIER) processes and controls interrupts from the RGP to the core.

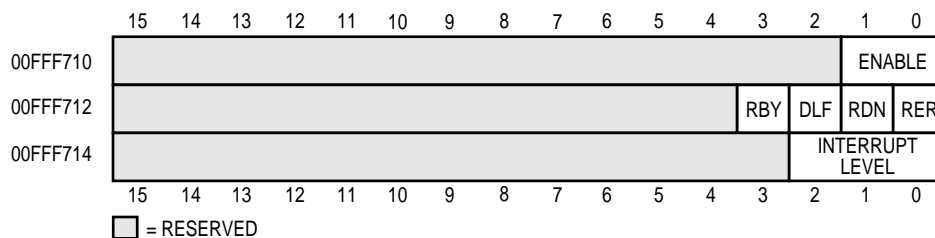


Figure 11-3. RGP Interrupt Event Register

The RDN and RER bits are set by the RGP to indicate one interrupt event to the core and each event bit has a corresponding bit in the enable field. The RBY bit indicates that the RGP is executing a display list and the DLF bit indicates that a second display list is queued. The RGP sets the RBY bit when the RSR is loaded and clears the RBY bit when it encounters a STOP graphic order. If a second display list start address is loaded before a STOP graphic order is encountered, the RGP keeps RBY set as the second display list starts.

Since the RSR is double buffered, the DLF bit is set only after a second display list address is loaded. If a second display list address was loaded and RGP reaches the end of the current display list, it clears the DLF bit, keeps RBY set, and starts reading the second display list. DLF is cleared as soon as a STOP graphic order opcode for the current display list is encountered. If the RSR is loaded when DLF is set, the new address overwrites the previously queued address. Also, when an error occurs, RBY and DLF retain their state until the RGP is soft-reset.

## 11.2 RGP BASIC OPERATION

The RGP can render either an entire page from a display list or multiple bands from a single banded display list. The RGP is activated by writing the starting address of a display list to the RSR. The RGP then executes the display list and renders a page or band image and when the end of the display list is reached, the RGP generates an RGP done interrupt event to the core and waits for another display list address. Be aware that a second display list address can be loaded while the RGP is working on the first display list. In this case, an interrupt is generated upon completion of the first display list and the second display begins executing immediately. Also, a second interrupt is generated upon completion of the second display list.

The ability of the RGP to render multiple bands from a single banded display list allows for complete freedom in the design of a banded memory system. A banded display list contains band information near the beginning of the list. This band information includes the address and size of the band, as well as the band number that is to be processed. The RGP uses this information to determine which orders from the display list to process and where in memory to create the bitmap image for the specific band. After one band is fully rendered and the RGP generates an RGP done interrupt event, the software should adjust the display list to reflect the next band information and then restart the RGP. The quantity, size, and location of band buffers is determined by the core's software. These band parameters can be dynamically altered to suit a particular application or page complexity. The band numbers are under software control to allow for nonsequential band processing applications like duplex printing.

Depending on the length of the display lists and interrupt latency, one or both display lists can complete and generate an RGP done interrupt event before any interrupt is even serviced. To determine whether one or both display lists has executed, the software must first clear the RIER's RDN bit, then reread it to examine the current setting. If the RIER's RBY bit is set and a RGP done interrupt is being serviced, then one display list has finished and the second display list is currently being executed. If the RBY bit is clear, both display lists have finished and the RGP is sitting idle.

The RGP can detect certain errors during the execution of a display list, including an out-of-range display list start address, out-of-range graphic order operand addresses, and illegal graphic order opcodes. When the RGP detects one of these errors, it immediately stops processing the current display list and sets the RIER's RER bit. When RER is set, the RDR determines the graphic order that generated the error. However, the RGP must be reset by an RGP soft-reset to return it to normal operation. A write to the RGP bit in the soft-reset register initiates an RGP soft-reset operation. See **Section 5 Interrupt and Exception Handling** for more information. On an RGP soft-reset, the RGP registers are initialized as follows:

1. RSR is purged.
2. RIER's DLF and RBY bits are cleared.
3. RDR is cleared to all zeros.
4. Internal Boolean code registers that hold the Boolean values defined by the SET\_BOOL graphic orders are cleared to all zeros.

If the RSR is loaded between the time an RGP error is detected and the time the soft-reset is issued, the address stored in the RSR is lost. Additionally, if RDR information is needed, the RDR must be read before an RGP soft-reset is executed. Due to the prefetching characteristics of the RGP, and depending on the type of error, the address returned by the RDR may be slightly beyond the source of the error.

When a graphic order is interrupted due to a band fault, all parameters necessary to resume the data transfer at the point where the band fault occurred are written back into the display list. In addition, the graphic order's band number is incremented (or decremented for 180° pages) and written back into the display list to ensure that the graphic order is executed for the next band. Since the hardware automatically updates graphic orders that generate band faults, graphic orders that span multiple bands execute without any software intervention. Once a graphic order has been updated after a band fault, the execution of the display list continues with the next graphic order in the list.

## SECTION 12

# GRAPHIC OPERATIONS

The MC68322's graphics unit (comprised of the print engine interface and the RISC graphics processor) operates on a display list, which is a collection of graphic orders. A graphic order is a special instruction or command that directs the MC68322 to perform a discrete function, such as setting internal environment registers or executing graphic operand transfers (source to destination bit maps). The MC68322 supports two types of transfers—bit block and scanline.

### 12.1 TYPES OF BITMAPS

The basic element of a graphic operand is the bitmap. As described in **Section 1 Introduction**, a bitmap is a two dimensional array composed of scanlines (each row in the array) and pixels (the junction of a scanline and column in the array). The width of the bitmap (X dimension) is the number of pixels in the scanline and the height of the bitmap (Y dimension) is the number of scanlines. Graphic orders use six types of bit maps:

- **Banded**—A banded bitmap has a warp associated with it that defines the X dimension and a logical coordinate that defines the Y dimension. When specifying a banded bitmap in a transfer of data, the transfer of data is checked against the Y boundary of the bitmap. If the data falls beyond the Y boundary, a band fault occurs and the transfer of data for the current graphic order is prematurely terminated.
- **Unbanded**—An unbanded bitmap also has a warp associated with it that only defines the X dimension. However, there is no logical coordinate associated with its Y dimension. The significance of this is that no boundary checks are made for unbanded bit maps. When specifying an unbanded bitmap in a transfer of data, it is assumed that the bitmap is of sufficient size to complete the transfer.
- **Unexpanded**—An unexpanded bitmap is a two dimensional array of pixels that represents a low resolution image. Typically, an unexpanded bitmap describes a bitmap image that is created at 75, 100, 150, 200, or 300 dpi. In contrast, all other bit maps have an implied resolution that matches the print engine resolution, which can be 300, 600, or 1200 dpi. Unexpanded bit maps can be positioned anywhere in memory, packed or unpacked, and they do not need to be aligned to word boundaries. Also, an unexpanded bitmap is always used as a source bitmap by expanded graphic orders.
- **Expanded**—An expanded bitmap is a conceptual term that describes the outcome of processing an unexpanded source bitmap during the execution of an expanded graphic order. The expanded graphic order increases the resolution of the unexpanded bitmap to match the resolution of the destination bitmap being used in a transfer. Since the RGP expands the unexpanded bit maps during the transfer, expanded bit maps never actually exist in memory.

- **Frame**—A frame bitmap is a two dimensional array of pixels that is generally a subset of some larger bitmap array (the width and warp are identical). For example, a frame bitmap could describe the bounding box of a character. To better illustrate this difference, suppose a rectangle is transferred to a bitmap, both the warp of the bitmap and the width of the rectangle being transferred must be specified. When transferring a rectangle to a frame bitmap, the warp is the width of the rectangle being transferred. Operations that involve transferring pixels to frame bit maps are particularly useful when a small rectangle of a larger image must be saved for future reference. The frame bitmap is the smallest possible storage means for the rectangle because the scanlines are packed in memory. The end of one scanline and the beginning of the next have no unused bits between them.
- **Halftone**—A halftone bitmap is a special type of bitmap because it is automatically replicated in both the horizontal and vertical directions. It is intended to hold a pattern that is repetitively applied to data being transferred from a source bitmap to a destination bitmap, thus causing a change in the appearance of the data.

## 12.2 GRAPHIC OPERANDS

Up to three operands are used when composing the print image—source, destination, and halftone. The source operand is typically located in a frame or unexpanded bitmap, the destination operand in a banded or unbanded bitmap, and the halftone operand in a halftone bitmap. These operands are represented by a specific 1-byte constant value, which is listed in Table 12-1.

**Table 12-1. Graphic Operation Data Operand Constant Values**

CONSTANTS	BINARY CODING
Zero	00000000
One	11111111
Destination	10101010
Source	11001100
Halftone	11110000

These five constant values can be combined using Boolean arithmetic computations to yield a 1-byte Boolean code that corresponds to a specific transfer effect. See **Section 12.4 Boolean Codes** for more information.



## 12.3 TYPES OF GRAPHIC OPERANDS

Graphic orders specify which graphic operands to transfer and there are three types:

- All graphic orders of the form XXXX\_D (such as SET\_BOOL\_D) define 1-operand transfers that specify only the destination bitmap.
- All graphic orders of the form XXXX\_SD or XXXX\_HD (such as BLT2F\_SD or SL2UB\_HD) define 2-operand transfers that specify the destination bitmap and either the source or halftone bitmap (the halftone applied to the destination).
- All graphic orders of the form XXXX\_SHD (such as BLT2BB\_SHD) define 3-operand transfers that specify the destination bitmap, source bitmap, and halftone bitmap (the halftone applied to the source and destination).

## 12.4 BOOLEAN CODES

The preferred result of a graphic operand transfer must be specified by means of the 1-byte Boolean code that directs the MC68322 to logically combine the given source, destination, and/or halftone operands during the graphic operation. Figure 12-1 illustrates eight common Boolean-coded graphic operation transfers and all 256 Boolean-coded graphic operation transfer combinations are represented in Figure 12-2. They involve all possible logical combinations between the true and inverted versions of the three operands.

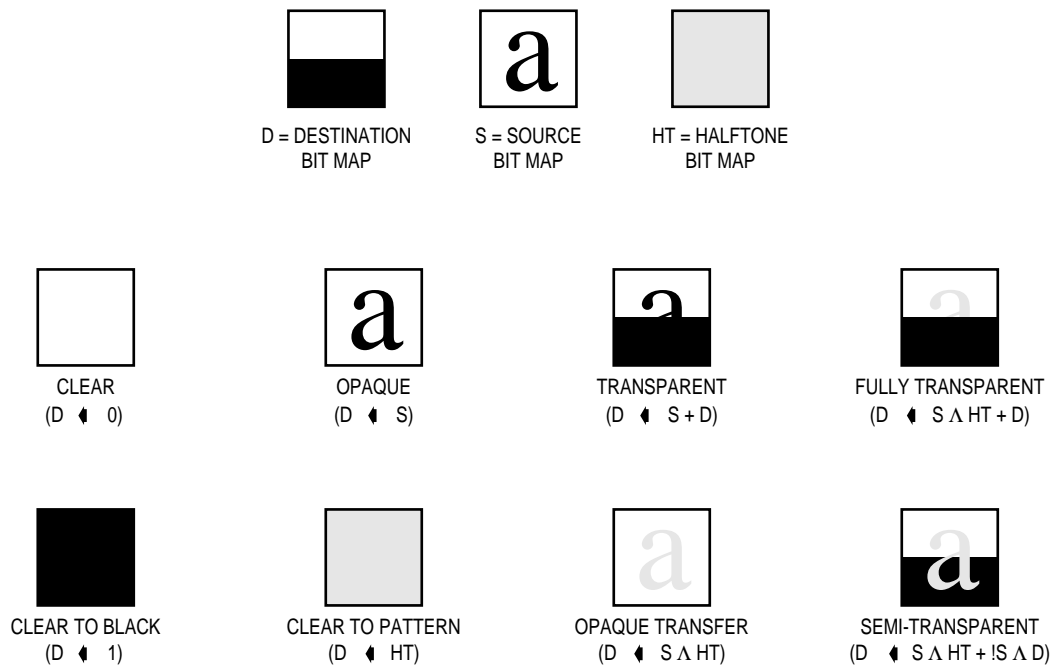
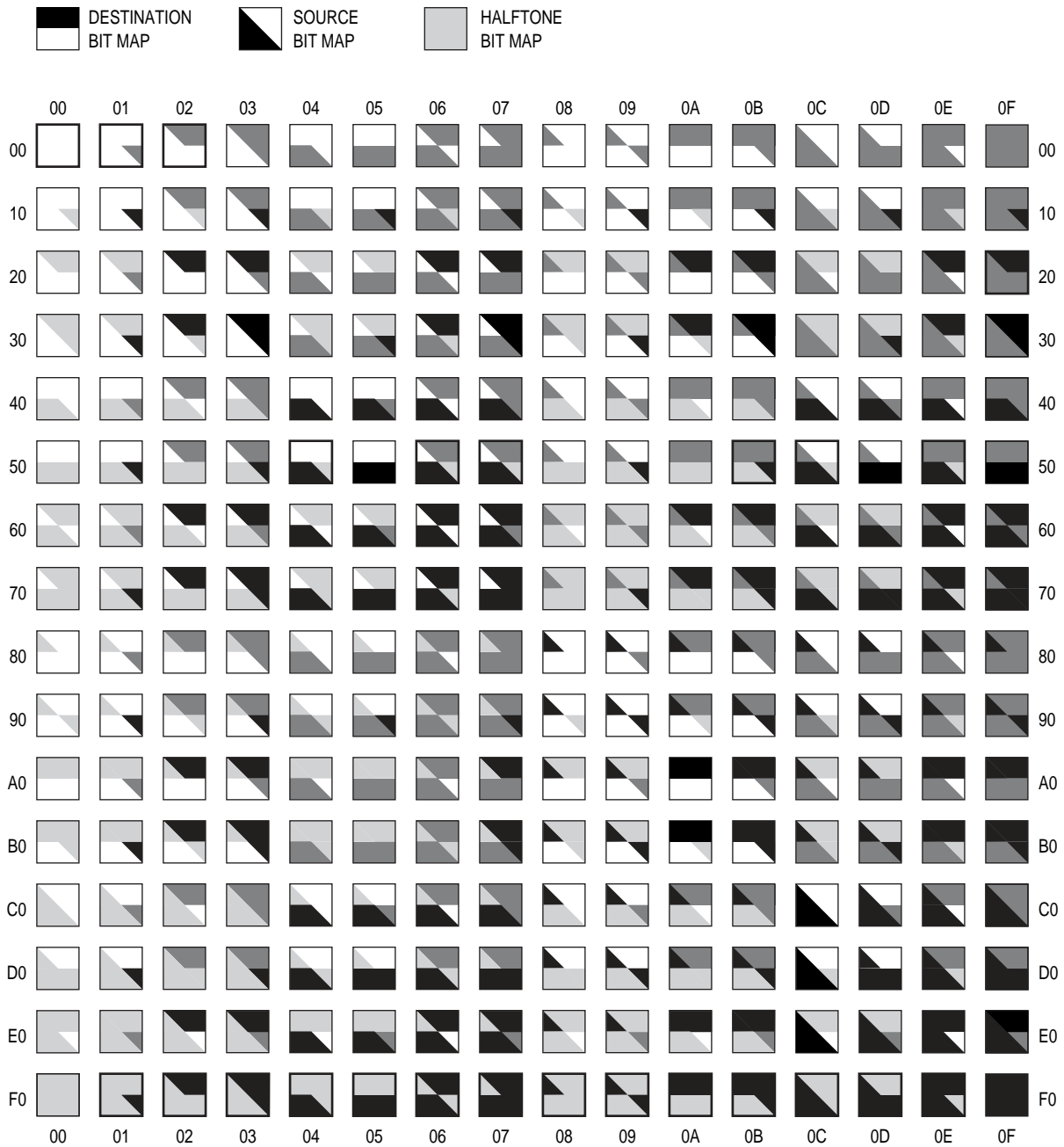


Figure 12-1. Eight Common Graphic Operation Transfers



**Figure 12-2. 256 Possible Boolean Coded Graphic Operation Transfers**

The Boolean codes shown in Figure 12-2 are derived from the constants listed in Table 12-1. For example, setting the Boolean code to  $EE_{16}$  causes the MC68322 to OR the source and destination operands, thus producing a transparent combination of the two. This combination, illustrated in Figure 12-2 (row E0, column 0E), is calculated by logically OR'ing the destination and source constants.

DESTINATION	V	SOURCE	=	BOOLEAN CODE
10101010	V	11001100	=	11101110 = $EE_{16}$

A more complicated graphic operation transfer is calculated in the same way. For example, the Boolean code for a semi-transparent source transfer onto the destination bitmap involving halftoning (sometimes used by PostScript) is determined as follows:

$$\begin{array}{rclclcl} \text{[(NOT SOURCE)} & \wedge & \text{DESTINATION]} & \vee & \text{(SOURCE } \wedge \text{ HALFTONE)} & = & \text{BOOLEAN CODE} \\ \text{[(NOT 11001100)} & \wedge & \text{10101010]} & \vee & \text{(11001100 } \wedge \text{ 11110000)} & = & \text{11100010} = E2_{16} \end{array}$$

Take caution when calculating a Boolean code to ensure only operand values that are contained in the graphic operation transfer are used. For example, the halftone value should not be used when specifying the Boolean code for a transfer involving a source/destination graphic order. If this constraint is violated, the MC68322 treats the extra parameter(s) as zero. Thus, if the Boolean code for  $\text{dest} = (\text{source} \wedge \text{halftone}) \vee \text{dest}$  is used to specify a transfer involving a source/destination graphic order, the MC68322 treats the halftone parameter as zero. The resulting operation becomes  $\text{dest} = (\text{source} \wedge 0) \vee \text{dest}$  or simply,  $\text{dest} = \text{dest}$ .

## 12.5 BIT BLOCK TRANSFERS

A bit block transfer (bitBLT) is an operation that combines up to three rectangular bit maps (source, halftone, and destination) in some Boolean combination and places the result in the original destination bitmap. An expanded bitBLT operation is similar to a standard bitBLT operation, except that each pixel of the source bitmap is replicated in one or both dimensions by scaling factors from 1 to 16. An expanded bitBLT graphic order reads an unexpanded source bitmap directly from memory and expands it to match the resolution of the destination bitmap. Then it is combined with the destination and/or halftone bitmap to complete the transfer.

For normal or expanded bitBLT operations, if the destination bitmap is a banded bitmap, the transfer terminates prematurely if the transfer frame extends below the bottom of the band. This is considered a band fault. The remainder of the display list is processed in the event of a band fault, but the transfer operation that caused the band fault will resume when the display list is rerun to render the next band of the page image.

## 12.6 SCANLINE TRANSFERS

Scanline transfers are used to operate on nonrectangular regions of bit maps. In their simplest form, they are used to fill arbitrary polygons and draw vectors. A scanline transfer involves operating on a specified set of scanline runs on one or more bit maps. The set of scanline runs is defined by a scanline table, which contains a series of bit string specifiers. Each bit string specifier is a compressed run-length encoding of a scanline run. The compressed format of scanline tables not only saves memory, but also improves performance since fewer memory fetches must be performed.

Scanline operations arise quite often from two sources. The first occurs from outline fonts, which describe the outline of a set of characters via splines, lines, and arcs. The outlines are scaled for the preferred point size through software algorithms. The result is a set of scanline endpoints that must be filled to create a solid character. The second occurs from vector images such as wire-frame diagrams, which are entered as either a series of line drawing graphic orders or as a previously generated bitmap. Since these vector images contain a high percentage of white space, they typically require less storage space when described as a series of scanline operations. Scanline transfers operate on a destination bitmap and specify a halftone bitmap to render grayed or patterned images. No source bitmap is involved in a scanline transfer.

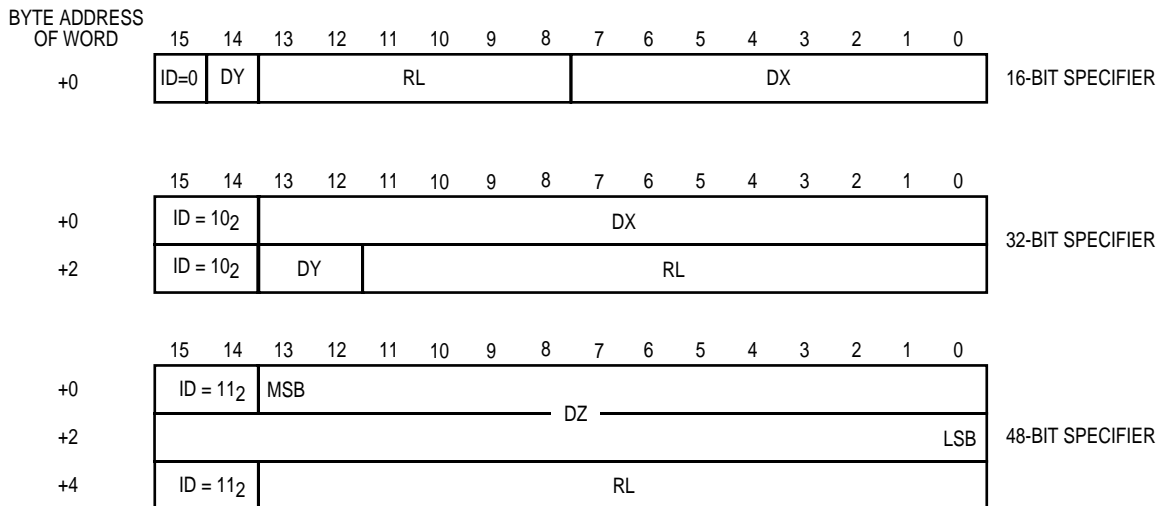
### 12.6.1 Scanline Tables and Bit String Specifiers

A scanline table consists of a series of bit string specifiers representing an image that has been compressed using run-length encoding. For example, bitmap fonts can be converted into this format to reduce memory requirements for font storage. A scanline table can also be used to efficiently represent line art and filled polygon shapes.

Each bit string specifier describes a displacement along with a run-length encoded definition of a graphics operation for the RGP to perform on a single scanline. The MC68322 supports three different sizes of bit string specifier formats: 16-, 32-, and 48-bit. The smaller formats help to reduce memory requirements when short displacements or scanline runs are required. The larger formats allow any pixel in a bitmap to be reached with a single specifier. Three bit string specifiers are supported:

- 16-Bit—Conditionally moves to the next scanline, goes a short distance left or right from there, and then draws a line that is a maximum of 63 bits long.
- 32-Bit—Moves vertically up to three scanlines, goes a large distance left or right from there, and then draws a line that is a maximum of 4,095 bits long.
- 48-Bit—Moves a very large distance in both X and Y dimensions and then draws a line that is a maximum of 16,383 bits long.

All bit string specifiers are multiples of 16 bits and must always be located on word memory boundaries (0 mod 2 byte addresses). Each bit string specifier consists of an ID field, an unsigned run length (RL) field, and a signed displacement (DX, DY, or DZ) field. Figure 12-3 illustrates the three types of bit string specifier formats.



NOTES:

- RL = Run Length (unsigned)
- DX = X Dimension Displacement—Horizontal (signed)
- DY = Y Dimension Displacement—Vertical (unsigned)
- DZ = X and Y Dimension Displacement—Horizontal and Vertical (signed)

**Figure 12-3. Bit String Specifier Formats**

The ID field identifies the type of bit string specifier. The bit string specifiers are designed to be recognizable no matter what direction the scanline table is read from memory. This is required to properly handle scanline graphic orders in a duplex banding environment where 0° and 180° pages are possible. Thus, the ID fields in the 32- and 48-bit bit string specifiers are duplicated to allow parsing of the scanline table in either direction. The RL field indicates the number of pixels to be drawn horizontally during the scanline run.

The displacement fields indicate the number of horizontal and vertical pixels to skip before drawing a scanline. The DX field indicates the signed horizontal movement along the X dimension of the bitmap. The DY field indicates the unsigned vertical movement along the Y dimension of the bitmap. The DZ field is the signed aggregate displacement based on a calculation of the preferred X and Y movements along with the destination warp. Thus, the warp of the target bitmap must be known before building a scanline table containing 48-bit bit string specifiers. This calculation is  $DZ = (DY \times DW) + DX$ , where DW is the unsigned warp of the target destination bitmap. Note that DY is signed for this calculation. Table 12-2 lists the definitions for each field of the three types of bit string specifiers.

**Table 12-2. Bit String Specifier Field Definitions**

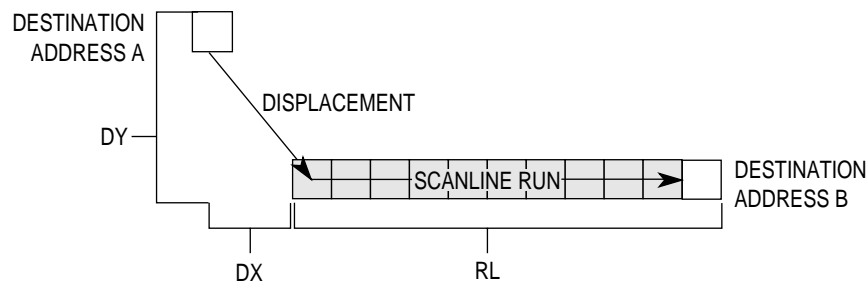
BIT STRING TYPE	FIELD	DEFINITION
16-bit	DY	Zero to one scanline.
	RL	0 to 63 pixels.
	DX	+127 to -128 pixels.
32-bit	DY	Zero to three scanlines.
	RL	0 to 4,095 pixels.
	DX	+8,191 to -8,192 pixels.
48-bit	RL	0 to 16,383 pixels.
	DZ	+536,870,991 to -536,870,992 pixels.

A 16-bit null termination bit string specifier of  $0000_{16}$  is placed before the beginning and after the end of every scanline table. This is a real specifier that specifies no movement of the base pointer and a run of zero pixels. Note that the null terminators can be shared between scanline tables placed adjacent in memory. In other words, the ending point  $0000_{16}$  of one scanline table serves as the starting point  $0000_{16}$  for the next scanline table. The dual termination of the scanline tables is required for duplex banding applications where the MC68322 reads the scanline table in both forward and reverse directions to render  $0^\circ$  and  $180^\circ$  pages.

### 12.6.2 Scanline Run Operation

When a scanline graphic order begins, the current destination address is established by the initial address specified in the graphic order. The MC68322 maintains a current destination address value and updates it after each bit string specifier is executed. For  $0^\circ$  pages, after a scanline run occurs, the current destination address points to the pixel immediately following the last pixel in the run. For  $180^\circ$  pages, the current destination address points where the last specifier's X and Y displacements put it.

The displacements specified by DX, DY (for 16- and 32-bit bit string specifiers), and DZ (for 48-bit bit string specifiers) are applied before executing the scanline run for a  $0^\circ$  page and after executing it for a  $180^\circ$  page. Figure 12-4 illustrates a scanline run using the values in the bit string specifier fields. Notice that destination address A is the initial address for  $0^\circ$  pages as well as the final address for  $180^\circ$  pages and, in turn, address B is the final address for  $0^\circ$  pages and the initial address for  $180^\circ$  pages.



**Figure 12-4. Scanline Run Operation**

When rendering to banded bit maps of 0° pages or to frame and unbanded bit maps, the displacement is added to the current destination address before the scanline is drawn. The scanline run is then completed in a left to right direction. When rendering to banded bit maps of 180° pages, the run length is first subtracted from the current destination address and that address is saved. Then the scanline run is carried out in a left to right direction and the displacement is subtracted from the saved destination address.

### 12.6.3 Executing During Banded Applications

When executing a scanline table for a banded bitmap, the MC68322 performs boundary checking to detect a band fault before executing each bit string specifier. A band fault occurs when the destination bitmap is a banded bitmap and the scanline frame extends below the end of the band. A band fault will cause the scanline transfer to terminate prematurely. The remainder of the display list is processed in the event of a band fault, but the execution of the scanline table resumes when the display list is rerun to render the next band of the page image.

The MC68322 does not, however, check for a band fault while processing the scanline run lengths. During banding applications, there are two ways to use bit string specifiers that can cause unwanted destruction of data. The destruction occurs when the bit string specifier causes processing to occur outside of the banded bitmap. One way is when a bit string specifier contains a signed offset that references a previous band (see Figure 12-5). The MC68322 checks only the lower boundary of the banded bitmap, so the violation goes undetected. Normally, the MC68322 would process the bit string specifier's run length at a memory location that is not contained within the target bitmap.

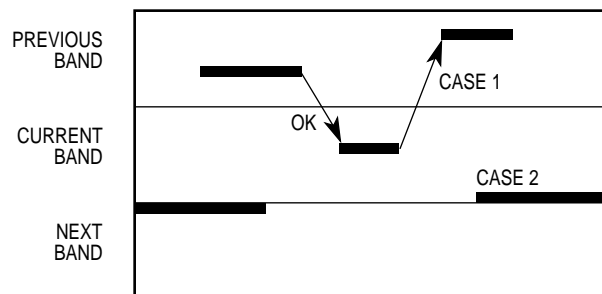


Figure 12-5. Illegal Bit String Specifier Use

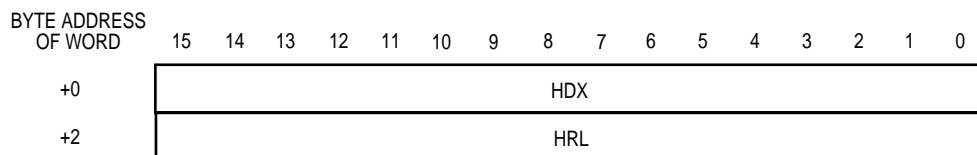
Another way is when a bit string specifier contains a run length that wraps from one scanline to another and the new scanline is beyond the end of the current band. The MC68322 does not check horizontal and vertical boundaries while processing the bit string specifier's run.

## 12.6.4 Halftone Companion Tables

A halftone bitmap is automatically replicated in both the X and Y dimensions. For scanline graphic orders, the task is complex since each scanline run describes an individual area on which to be operated. For example, on 0° pages after the displacement required by the bit string specifier has been applied, the proper location in the halftone bitmap must be calculated. For the 16-bit bit string specifier, this task is relatively straightforward and can be carried out with a minimum number of cycles. For the 32- and 48-bit specifiers, the task is considerably more complex because of the much larger displacement values associated with them. The task can be accomplished, but at a significant cost in overhead cycles. To reduce the penalty for the larger bit string specifier formats, a halftone companion table is employed. This eliminates virtually all overhead cycles in return for only slightly higher memory storage requirements within typical scanline tables.

The halftone companion table contains a list of corresponding halftone specifiers (similar in composition to bit string specifiers) for each 32- and 48-bit bit string specifier in the scanline table. There is no corresponding halftone specifier for 16-bit bit specifiers because they are halftoned without assistance. The halftone specifiers use displacements instead of absolute values. This minimizes the number of halftone tables to one per halftone pattern for each halftoned character. Thus, if a character is halftoned, only one halftone table must be constructed, regardless of the number of times the character is used. The same halftone table for a particular character can be used with different halftone patterns if the dimensions of each halftone bitmap are identical.

Each field of a halftone specifier corresponds to a similar field in the 32- or 48-bit bit string specifier. As illustrated in Figure 12-6, the 32-bit halftone specifier contains two fields—a halftone horizontal movement (HDX) and a halftone run length (HRL). HDX corresponds to the 32-bit bit string specifier's DX field and HRL corresponds to the RL field.



**NOTES:**

HRL = Halftone Run Length (unsigned)

HDX = X Dimension Displacement—Horizontal (unsigned)

**Figure 12-6. 32-Bit Halftone Specifier Format**

HDX and HRL are defined as follows where HW is equal to the warp of the target halftone bitmap:

$$\text{HDX} = \text{DX} \bmod \text{HW} \ (\text{DX} \geq 0) \vee [\text{HW} - (-\text{DX} \bmod \text{HW})] \bmod \text{HW} \ (\text{DX} < 0)$$

$$\text{HRL} = \text{RL} \bmod \text{HW}$$



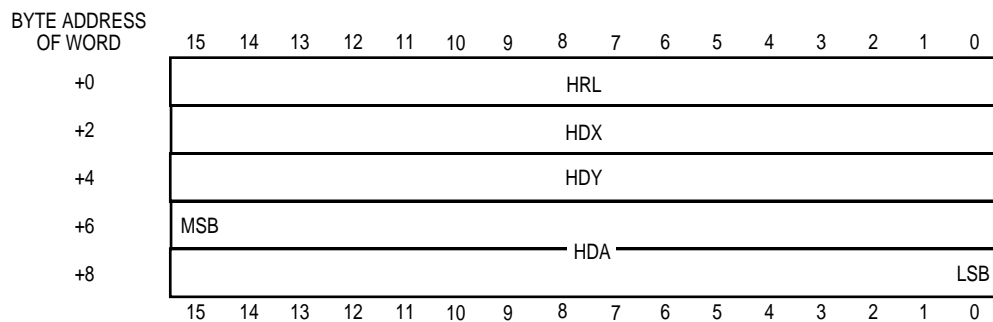
Both HDX and HRL are unsigned values, but their values should satisfy the following boundary conditions:

$$0 < \text{HRL} < \text{HW}$$

$$0 < \text{HDX} < \text{HW}$$

The Y displacement found in the 32-bit bit string specifier (DY field) is automatically handled by the MC68322 and has no corresponding field in the companion halftone specifier. HRL is required for 180° pages since the graphics unit uses this offset to internally render scanline runs from left to right.

As illustrated in Figure 12-7, the corresponding halftone specifier for the 48-bit bit string specifier is 80 bits long and contains four fields—halftone vertical movement (HDY), halftone horizontal movement (HDX), halftone run length (HRL), and halftone physical starting address (HDA).



NOTES:

- HRL = Halftone Run Length (unsigned)
- HDX = X Dimension Displacement—Horizontal (unsigned)
- HDY = Y Dimension Displacement—Vertical (unsigned)
- HDA = Halftone Starting Address Displacement (unsigned)

**Figure 12-7. 48-Bit Halftone Specifier Format**

The preferred horizontal and vertical displacements used in the original calculation of the bit string's DZ field are required to generate the parameters of the companion halftone specifier. The four parameters for the 48-bit companion halftone specifier are defined as follows:

$$\text{HRL} = \text{RL} \bmod \text{HW}$$

$$\text{HDX} = \text{DX} \bmod \text{HW} \ (\text{DX} \geq 0) \ \text{or} \ [\text{HW} - (-\text{DX} \bmod \text{HW})] \bmod \text{HW} \ (\text{DX} < 0)$$

$$\text{HDY} = \text{DY} \bmod \text{HH} \ (\text{DY} \geq 0) \ \text{or} \ [\text{HH} - (-\text{DY} \bmod \text{HH})] \bmod \text{HH} \ (\text{DY} < 0)$$

$$\text{HDA} = (\text{HDY} \times \text{HW}) + \text{HDX}$$

Where:

DX and DY are the signed X and Y dimension values used in the calculation of DZ

HH is the target halftone bitmap height

HW is the target halftone bitmap width

HRL, HDX, HDY, and HDA are all unsigned values and their values should satisfy the following boundary conditions:

$$0 < \text{HRL} < \text{HW}$$

$$0 < \text{HDX} < \text{HW}$$

$$0 < \text{HDY} < \text{HH}$$

$$0 < \text{HDA} < \text{HZ}$$

Where:

HZ is the total number of bits in the halftone bitmap

Any 32- or 48-bit bit string specifier, with a corresponding halftone specifier, must not have a horizontal displacement that extends beyond either side of the destination bitmap. The halftoning operation does not track the Y dimension displacement caused by wrapping around the sides of a bitmap. Of course, the same is true for the run lengths since no horizontal clipping is performed at the left or right edges of the destination bitmap for any graphic order.

Unlike the scanline table, the halftone table requires no  $0000_{16}$  terminators. The parsing of the halftone table tracks that of the scanline table, reading both tables in parallel. For 32- and 48-bit specifiers, the MC68322 reads a specifier from each table and for 16-bit specifiers, only the scanline table is read. The halftone specifiers, like their bit string counterparts, must be located on word-aligned boundaries and, consequently, the halftone table address pointer specified in the scanline graphic order must be word aligned (0 mod 2 byte address).

## 12.7 SCANLINE AND HALFTONE TABLE EXAMPLE

Figure 12-8 illustrates an example scanline table, its corresponding halftone table, and the resulting image. Notice that the halftone table contains only two specifiers, since the scanline table is mostly composed of 16-bit bit string specifiers, and that the halftone table is not terminated with  $0000_{16}$  null specifiers, but the scanline table is.

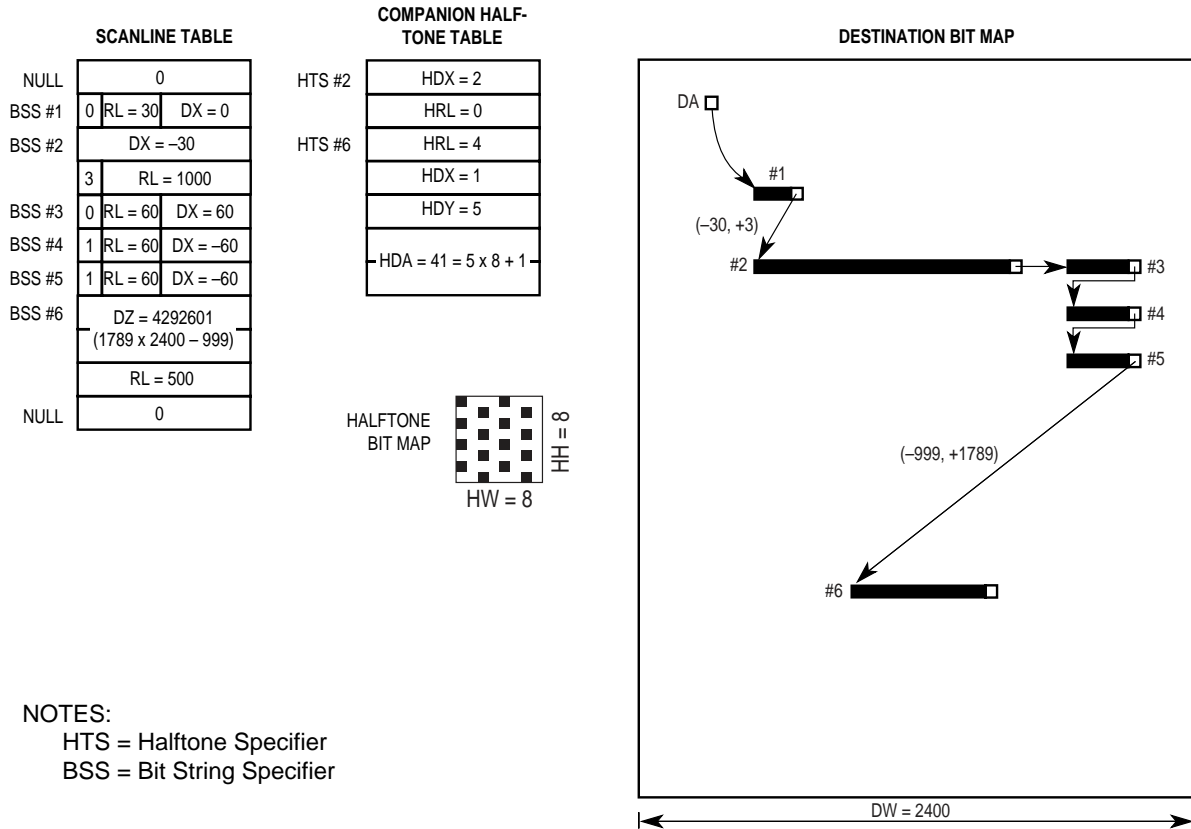


Figure 12-8. Scanline and Halftone Table Example

## 12.8 bitBLT AND SCANLINE ORDER EXECUTION

Table 12-3 lists the bitBLT and scanline transfer timings based on an MC68322 system running at 20 MHz with DRAM running at 3, 2, 1, 2 bus cycles.

**Table 12-3. bitBLT and Scanline Execution Times**

OPERANDS/ORDERS	TIME (MS)
<b>bitBLT EXECUTION TIME (32 × 32)</b>	
D = f(S)	30.75
D = f(S,D)	30.75
D = f(H,D)	31.05
D = f(S,H,D)	40.65
D = f() Solid fill	21.00
<b>bitBLT EXECUTION TIME (30 × 50)</b>	
D = f(S)	48.90
D = f(S,D)	48.90
D = f(H,D)	47.10
D = f(S,H,D)	63.30
D = f() Solid fill	31.65
<b>bitBLT EXECUTION TIME (100 × 100)</b>	
D = f(S)	139.05
D = f(S,D)	156.30
D = f(H,D)	148.05
D = f(S,H,D)	189.45
D = f() Solid fill	93.50
<b>SCANLINE EXECUTION TIME (30 × 50)</b>	
D = f(SL,D)	70.05
D = f(SL,H,D)	195.90

## 12.9 LOCATION AND ADDRESS CONSTRAINTS

Display lists, scanline tables, and halftone tables must reside in DRAM space because the graphics unit cannot access chip-select space. The graphics unit can, however, fetch instructions from any of the MC68322's six DRAM channels and from one channel to another if two channels define a continuous region of memory. Keep in mind though, that due to its internal prefetch queues, the graphics unit must be able to read eight words beyond every display list, scanline table, and halftone table. If this rule is broken, an RGP error interrupt event could occur and cause the graphics unit to shut down.

The addressing conventions discussed in previous sections also apply to graphic order address parameters. All address fields are 32 bits wide. Address parameters that reference bit maps specify bit addresses, while all others require conventional byte addresses. The RGP requires all display lists, scanline tables, and companion halftone tables to start on word-aligned addresses. Since all graphic orders, bit string specifiers, and halftone specifiers are an even number of bytes in length, this guarantees that all successive orders and specifiers begin at word-aligned addresses as well. This requirement also guarantees that all word and long-word operands and field values are aligned on word boundaries. This is important for the core, which can access word and long-word data only on word boundaries. These rules do not apply to the actual bitmap data, which has no alignment requirements.

The RGP internally stores all addresses in 32-bit registers and performs all address calculations using 32-bit arithmetic. When the RGP makes an access to DRAM for bitmap data, the bitmap address must first be converted to a byte address to accommodate the fetch. This conversion is done first by stripping off the low order four bits (these bits select a bit within the word) and then shifting the address to the right by three bits. This produces a word-aligned byte address that is used for a 16-bit DRAM fetch of the bitmap data.

# SECTION 13

## GRAPHIC ORDERS

This section describes the MC68322 graphic orders and lists them in alphabetical order. For each graphic order a functional description, opcode, parameter format, and definitions of its parameters are provided. Graphic orders specify one or more graphic operands, as indicated in the mnemonic by the last few characters. See **Section 12 Graphic Operations** for more information.

### 13.1 TYPES OF GRAPHIC ORDERS

The five types of graphic orders consist of the following:

- Initialization
- Program flow control
- Bit block transfer
- Expanded bit block transfer
- Scanline transfers

The initialization, bit block transfer, and expanded bit block transfer graphic orders are processed as a stream of instructions from the display list. The scanline graphic orders include a pointer to a scanline table that is a compressed run-length encoding of an image, such as a font character.

#### 13.1.1 Initialization

Initialization graphic orders (those whose mnemonics begin with SET) define bitmap parameters and Boolean codes to be used during transfers. The SET\_BMAP graphic orders are defined for four bitmap types—banded, unbanded, source, and halftone. They are used to load bitmap parameters into internal registers. The four SET\_BOOL graphic orders are defined for each type of Boolean operation—destination only (\_D), halftone and destination (\_HD), source and destination (\_SD), and source, halftone, and destination (\_SHD). They are all used to load the four Boolean code registers that will govern the bitBLT or scanline transfer graphic order to be executed.

The RGP has four internal registers that contain the Boolean code used during operand transfers—BOOL\_D, BOOL\_SD, BOOL\_HD, and BOOL\_SHD. The register BOOL\_D is loaded using the SET\_BOOL\_D graphic order and holds the Boolean code used during destination only bitBLTs and scanline transfers. BOOL\_SD and BOOL\_HD are loaded using the SET\_BOOL\_SD and SET\_BOOL\_HD graphic orders. BOOL\_SD holds the Boolean codes used during source, destination bitBLTs.

BOOL\_HD holds the Boolean code used during halftone, destination scanline transfers. BOOL\_SHD contains the Boolean code used during source, halftone, destination bitBLTs. bitBLTs can use BOOL\_D, BOOL\_SD or BOOL\_SHD Boolean codes and scanline transfers can use BOOL\_D or BOOL\_HD.

**Table 13-1. Graphic Order Organization**

TYPE	GRAPHIC ORDER	DESCRIPTION
Initialization	SET_BBMAP	Set Banded Destination Bitmap Parameters.
	SET_UBMAP	Set Unbanded Destination Bitmap Parameters.
	SET_SBMAP	Set Source Bitmap Parameters.
	SET_HTBMAP	Set Halftone Bitmap Parameters.
	SET_BOOL_D	Set Boolean Operator For Destination Only Transfers.
	SET_BOOL_HD	Set Boolean Operator For Halftone, Destination Transfers.
	SET_BOOL_SD	Set Boolean Operator For Source, Destination Transfers.
	SET_BOOL_SHD	Set Boolean Operator For Source, Halftone, Destination Transfers.
Program Flow Control	JUMP	Jump To The Specified Address In The Display List.
	STOP	Stop Execution Of The Graphic Orders.
Bit Block Transfers	BL2F_D	bitBLT To Frame, Destination Only.
	BL2F_SD	bitBLT To Frame Source, Destination.
	BL2F_SHD	bitBLT To Frame Source, Halftone, Destination.
	BL2UB_D	bitBLT To Unbanded bitmap, Destination Only.
	BL2UB_SD	bitBLT To Unbanded bitmap, Source, Destination.
	BL2UB_SHD	bitBLT To Unbanded bitmap, Source, Halftone, Destination.
	BL2BB_D	bitBLT To Banded bitmap, Destination Only.
	BL2BB_SD	bitBLT To Banded bitmap, Source, Destination.
BL2BB_SHD	bitBLT To Banded bitmap, Source, Halftone, Destination.	
Expanded Bit Block Transfers	BLT2F_XD	bitBLT To Frame expanded Source, Destination.
	BLT2F_XHD	bitBLT To Frame expanded Source, Halftone, Destination.
	BLT2UB_XD	bitBLT To Unbanded bitmap, expanded Source, Destination.
	BLT2UB_XHD	bitBLT To Unbanded bitmap, expanded Source, Halftone, Destination.
	BLT2BB_XD	bitBLT To Banded bitmap, expanded Source, Destination.
	BLT2BB_XHD	bitBLT To Banded bitmap, expanded Source, Halftone, Destination.
Scanline Transfers	SL2F_D	Scanline To Frame, Destination Only.
	SL2F_HD	Scanline To Frame, Halftone, Destination.
	SL2UB_D	Scanline To Unbanded bitmap, Destination Only.
	SL2UB_HD	Scanline To Unbanded bitmap, Halftone, Destination.
	SL2BB_D	Scanline To Banded bitmap, Destination Only.
	SL2BB_HD	Scanline To Banded bitmap, Halftone, Destination.

## 13.1.2 Program Flow Control

Program flow control graphic orders control the execution order for the display list. JUMP allows the execution to change to a different point in the display list. During banding, it is common to build a display list along with several headers. These headers contain a SET\_BBMAP graphic order to set the parameters for the current band and a JUMP graphic order to move execution to the main body of the display list. The STOP graphic order signals the end of the display list and normally results in an interrupt to the core.

## 13.1.3 Bit Block Transfer

Bit block transfer (bitBLT) graphic orders specify a rectangular bitmap transfer. There are nine different bitBLT graphic orders that provide the nine combinations of destination bitmap and operand type. Three of these control bitBLTs to frames, three control bitBLTs to unbanded bit maps, and three control bitBLTs to banded bit maps. These graphic orders rely on certain parameters being previously set by initialization graphic orders.

## 13.1.4 Expanded Bit Block Transfer

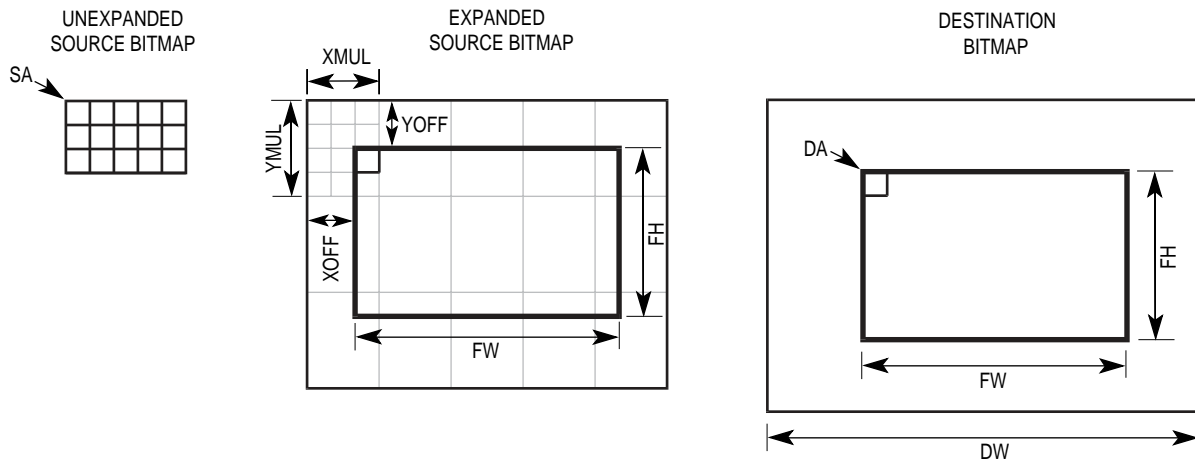
There are six expanded bitBLT graphic orders that are used to manipulate and transfer low resolution bitmap images. Two of these control expanded bitBLTs to frames, two control expanded bitBLTs to unbanded bit maps, and two control expanded bitBLTs to banded bit maps. These graphic orders rely on certain parameters being previously set by initialization graphic orders. Expanded bitBLT graphic orders are particularly useful in applications that regularly receive low resolution bitmap images. Note that the warp of the unexpanded bitmap is included as an operand of each expanded bitBLT graphic order.

Expanded bitBLT graphic orders can read unexpanded bit maps that are  $75 \times 75$ ,  $100 \times 100$ ,  $150 \times 150$ ,  $200 \times 200$ , and  $300 \times 300$  dpi, and expand the images in both the X and Y dimensions during the transfer to match any combination of 300, 600, or 1200 dpi. This is accomplished by specifying two expansion factors in the graphic order: one for the X dimension and one for the Y dimension. An expansion factor of 1, 2, 3, 4, 6, 8, 12, or 16 can be specified in the X dimension and any value from 1 to 16 in the Y dimension. This allows a single-step expansion to printer resolutions of 300, 600, and 1200 dpi.

The expanded bitBLT graphic orders support clipping of expanded bit maps, in both the X and Y dimensions. Clipping in the X dimension is controlled by two graphic order operand values, XOFF and FW, as illustrated in Figure 13-1. XOFF is an offset in bits from the left edge of the expanded bitmap and FW is the transfer frame width. Together, XOFF and FW provide bit-granular control of clipping at the left and right extremes of the expanded bitmap. When clipping, other graphic order operands may also require adjusting, including the destination address (DA), halftone address (HA), and halftone X remainder (HXR).

Clipping in the Y dimension is controlled in a similar way with the two graphic order operand values YOFF and FH. YOFF is an offset in scanlines from the top edge of the expanded bitmap (or bottom edge, when rendering in bottom to top order) and FH is the transfer frame height. Together, YOFF and FH provide scanline granular control of clipping at the top and bottom extents of the expanded bitmap. When clipping, other graphic order operands may also require adjusting, including the DA, HA, and the halftone Y remainder (HYR).





**Figure 13-1. Controlling Left and Right Clipping of Expanded Bit Maps**

### 13.1.5 Scanline Transfer

Scanline graphic orders differ from bitBLT graphic orders in that the individual scanline runs describe the transfer and can effectively generate a nonrectangular and unconnected destination area. There are six different scanline graphic orders that provide the six combinations of destination bit maps and operand types. Two of these control scanline transfers to frames, two control scanline transfers to unbanded bit maps, and two control scanline transfers to banded bit maps.

Each scanline graphic order specifies a pointer to a scanline table. The graphics unit executes the whole scanline table before continuing to execute the rest of the graphic orders in the display list. As with the bitBLT graphic orders, the scanline graphic orders rely on certain parameters being previously set by initialization graphic orders.

## 13.2 SEQUENCE OF THE DISPLAY LIST

The following is an example of a typical display list illustrating how the initialization graphic orders work with the bitBLT and scanline graphic orders.

SET_UBMAP	Set parameters for rendering a full page into an unbanded bitmap.
SET_BOOL_D and BLT2UB_D	Set the destination only Boolean code to draw in white, and clear the entire page.
SET_SBMAP	Set up for transfer from a font cartridge of a bit mapped font.
SET_BOOL_SD	Draw the following characters in black.
BLT2UB_SD, BLT2UB_SD, and BLT2UB_SD	Place three characters on the page.
SET_HTBMAP	Identify a light gray repeating halftone pattern.
SET_BOOL_HD	Identify a light gray repeating halftone pattern.
SL2UB_HD	Apply the halftone gray pattern during an upcoming transfer.
STOP	Render an entire scanline table image in light gray

## 13.3 GRAPHIC ORDER ADDRESSES

In the following graphic order descriptions, operand lengths for addresses are sometimes given as a pair of numbers. For example, 28 of 32. This indicates that the address (in this case a byte address) occupies the least-significant 28 bits of the long-word operand and that the MC68322 controls the four most-significant bits. For more information on address constraints for graphic orders, scanline tables, and bit maps see **Section 12 Graphic Operations**.

Some graphic orders specify address parameters that must be aligned to byte or word boundaries. When a byte address is required to be aligned on a word boundary, its least significant bit should be zero. The MC68322 forces the bit to be zero internally. There are two combinations of address granularity and alignment used by MC68322 graphic orders. In the descriptions of graphic orders and definitions of their parameters, the following two phrases are used:

- Bit address—A 32-bit bitmap address that points to any bit location in memory and specifies bitmap locations and pixels.
- Byte address, word aligned—A 28-bit word-aligned address that points to any word location in DRAM space. It is used for referencing graphic order instructions, scanline tables, and halftone companion tables.

### 13.3.1 Physical vs Logical Address

For graphic orders that render images to frames or unbanded bit maps, the address parameters are interpreted as physical addresses and the MC68322 uses them to access memory directly. In the case of banded bit maps, where only a portion of the physical page's image may be present in memory, address parameters are interpreted as logical addresses.

The MC68322 must translate logical addresses to physical space before graphic order execution begins. Translation information is provided to the MC68322 at the same time the banded bitmap dimensions are defined (via the SET\_BBMAP order). Physical addresses are translated back to logical addresses when a band fault occurs, so an updated logical address parameter can be written back to the graphic order.

### 13.3.2 Duplex Addresses

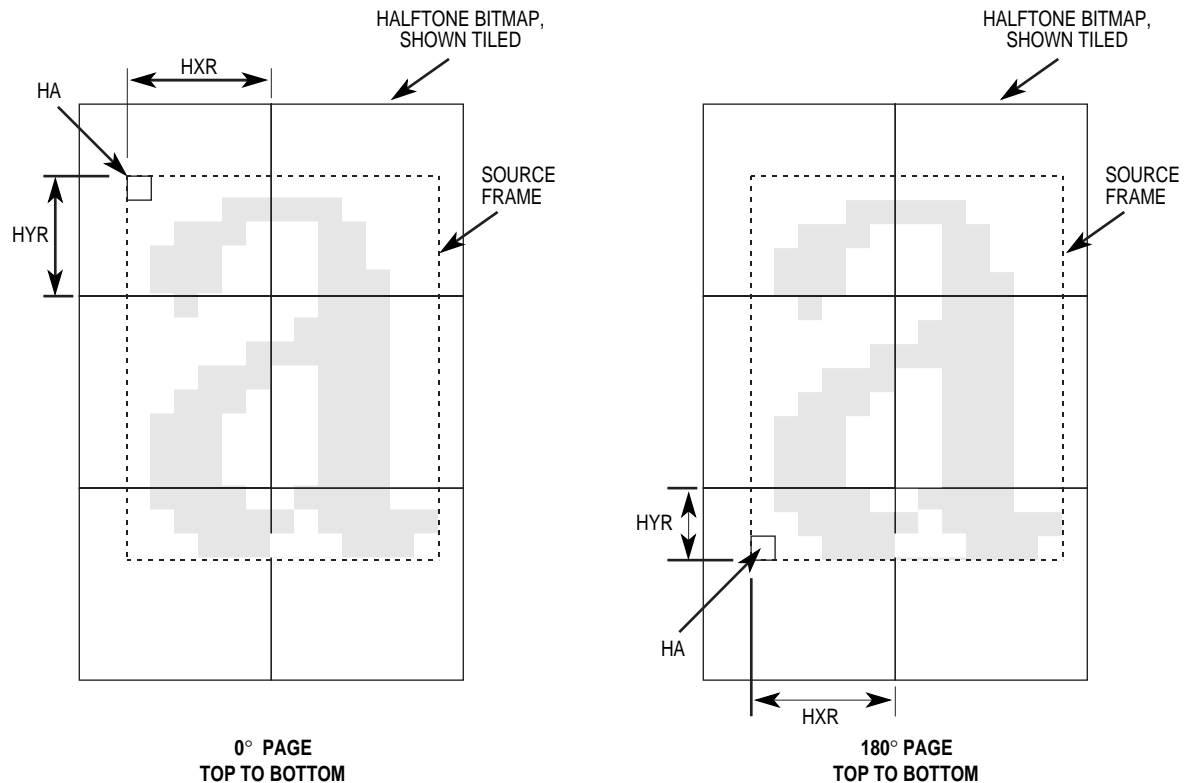
Based on the value of the bottom-to-top (B2T) parameter in the SET\_BBMAP graphic order, bitBLTs and scanline transfers can be rendered in a top to bottom direction for a 0° page or in a bottom to top direction for a 180° page. When bit block transferring to a banded bitmap with the B2T flag set, the definition of certain graphic order parameters change. Namely, the frame bitmap address parameters DA, SA, and HA must be provided so they are pointing to the bottom left corners of their respective frames (instead of the upper left) and the HYR halftone parameter must provide the number of scanlines remaining to reach the top of the bitmap (instead of the bottom). Figure 13-2 uses the halftone bitmap parameters to illustrate this requirement.

The parameters of scanline operations are also affected when rendering a page from bottom-to-top, since scanline and halftone tables must be executed in reverse order when the B2T flag is set. DA, which normally gives the starting position for the scanline table, must instead give the address of the pixel just beyond the last run in the scanline table. The scanline table address, which normally points to the most-significant byte of the first bit string specifier in the table, must instead point to the most-significant byte of the last word. In addition, if a halftone is involved, HA, HXR, and HYR must be provided with respect to the same pixel at the end of the image and halftone table address must point to the most-significant byte of the last word in the halftone companion table.



**Note:** in contrast to the order in which the print engine video controller reads an image from memory. When the B2T bit is set in the PCB control register, the PVC reads memory in a bottom to top, right to left order, and in doing so, produces an image that appears rotated 180°.

Unbanded and frame bitmap operations are always rendered in top to bottom order by the graphics unit, independent of the value of the B2T flag. Notice that for full-page duplex applications, page images can always be rendered top-to-bottom and then printed 0° or 180° when rotated by the PVC. The B2T flag is important for banded duplex applications because it allows the bands to be rendered in the reverse order as needed for delivery to the print engine.



**Figure 13-2. Halftone Specification for bitBLT Operations**

## 13.4 BAND NUMBER AND BAND FAULTS

Each of the MC68322 graphic orders that operate on a banded bitmap contains a band number parameter. The band number is used by the MC68322 to determine when to execute the graphic order. Graphic orders whose band numbers match the current band being rendered are executed. Otherwise, the graphic order is skipped. Band numbers always increase in value from the top band to the bottom band of the image, regardless of the type of page ( $0^\circ$  or  $180^\circ$ ) being rendered. The band number for a graphic order must be determined by the software. It must be the number of the band where the first scanline of a graphic order resides when rendering a  $0^\circ$  page or the number of the band where the last scanline of a graphics order resides when rendering a  $180^\circ$  page.

In banding applications, bitBLT and scanline operations can extend beyond the bottom (or top, for  $180^\circ$  pages) boundary of a band. The MC68322 detects such band crossings (or band faults) during the course of the bitBLT or scanline operation. When a band fault occurs, the operation is prematurely terminated and the MC68322 accesses the display list to update the graphic order's parameters, including its band number, bitmap and scanline table addresses, and frame bitmap heights. The graphic order parameters are written back into the display list to begin the operation following the point of the band fault at the top (or bottom for  $180^\circ$  pages) of the next band. The MC68322 automatically adjusts the band number when the graphic order spans multiple bands. When a band fault occurs, the band number is incremented or decremented depending on the setting of the B2T flag. If the B2T flag is clear, the band number is incremented when an operation crosses a band boundary. If the B2T flag is set, the band number is decremented.

Band faults only occur during graphic orders that operate on banded bit maps. In the descriptions that follow, parameters updated by the MC68322 as the result of a band fault are marked with asterisks. In some cases, however, certain parameters will not be updated. Specifically, this affects source and halftone parameters. If the Boolean code for a two or three operand graphic order describes a function that does not require a source and/or halftone operand, then no bitmap is accessed and no parameters will be updated if a band fault occurs.

## 13.5 GRAPHIC ORDER DESCRIPTIONS

Table 13-2 contains the graphic orders sorted by opcode value with the page number they appear on.

**Table 13-2. Graphic Orders Sorted by Opcode**

OPCODE	GRAPHIC ORDER	PAGE	OPCODE	GRAPHIC ORDER	PAGE
0x00	STOP	13-60	0x17	BLT2F_XHD	13-29
0x01	JUMP	13-41	0x20	BLT2UB_D	13-32
0x08	SET_BBMAP	13-42	0x22	BLT2UB_SD	13-33
0x09	SET_UBMAP	13-48	0x23	BLT2UB_SHD	13-34
0x0A	SET_SBMAP	13-47	0x24	SL2UB_D	13-57
0x0B	SET_HTBMAP	13-46	0x25	SL2UB_HD	13-58
0x0C	SET_BOOL_D	13-45	0x26	BLT2UB_XD	13-36
0x0D	SET_BOOL_HD	13-45	0x27	BLT2UB_XHD	13-38
0x0E	SET_BOOL_SD	13-45	0x30	BLT2BB_D	13-9
0x0F	SET_BOOL_SHD	13-45	0x32	BLT2BB_SD	13-11
0x10	BLT2F_D	13-23	0x33	BLT2BB_SHD	13-13
0x12	BLT2F_SD	13-24	0x34	SL2BB_D	13-49
0x13	BLT2F_SHD	13-25	0x35	SL2BB_HD	13-51
0x14	SL2F_D	13-54	0x36	BLT2BB_XD	13-16
0x15	SL2F_HD	13-55	0x37	BLT2BB_XHD	13-19
0x16	BLT2F_XD	13-27			

## BLT2BB\_D

### Destination Only bitBLT to Banded Bitmap

PARAMETERS	SIZE	DESCRIPTION
0x30	Byte	BLT2BB_D Opcode.
BAND*	Byte	Band number when graphic order is executed.
DA*	Long Word	Destination logical bit address.
FW	Word	Frame width in bits.
FH*	Word	Frame height in scanlines.

Note: \* Denotes A Parameter That The MC68322 Updates When The Frame Crosses A Band Boundary.

The BLT2BB\_D graphic order causes the MC68322 to modify a frame of a destination banded bitmap. The destination pixels are manipulated as specified by the current BOOL\_D Boolean code. The destination bitmap parameters must have been previously defined by the SET\_BBMAP graphic order.

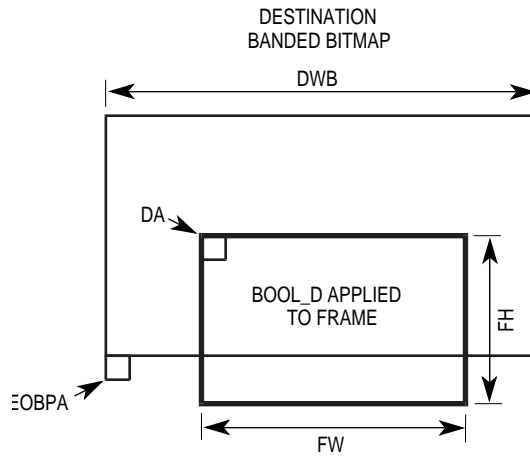
DA specifies the logical bit address of the area, or transfer frame, that starts in the destination banded bitmap. The logical bit address is converted to a physical bit address, by adding the PSUBL value set by the SET\_BBMAP graphic order. The start of the transfer frame must be within the bounds of the banded bitmap, but the end of the transfer frame may extend past the end of the bitmap. DA must point to the upper left corner of the transfer frame when the B2T flag is clear and to the lower left corner of the transfer frame when B2T is set. The warp of the destination banded bitmap is set by the SET\_BBMAP graphic order, which allows the bitmap to be packed or unpacked.

When a band fault is detected, the MC68322 rewrites the graphic order to update some of its parameters. The BAND number is incremented (or decremented, when the B2T flag is set), DA is repositioned to the starting pixel of the respective frame to be processed in the next band and FH is written back with the number of remaining scanlines in the destination frame.

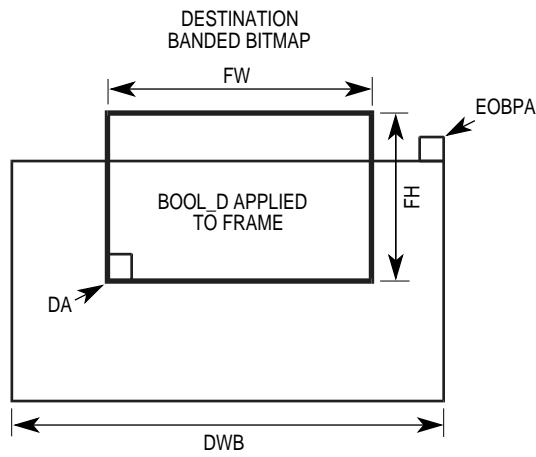
### Related Graphic Orders

SET\_BOOL\_D

SET\_BBMAP



**Figure 13-3. Destination bitBLT to Banded Bitmap—0° Page**



**Figure 13-4. Destination bitBLT to Banded Bitmap—180° Page**

## BLT2BB\_SD

### Source/Destination bitBLT to Banded Bitmap

PARAMETERS	SIZE	DESCRIPTION
0x32	Byte	BLT2BB_SD Opcode.
BAND*	Byte	Band number when graphic order is executed.
DA*	Long Word	Destination logical bit address.
FW	Word	Frame width in bits.
FH*	Word	Frame height in scanlines.
SA*	Long Word	Source physical bit address.

Note: \* Denotes A Parameter That The MC68322 Updates When The Frame Crosses A Band Boundary.

The BLT2BB\_SD graphic order causes the MC68322 to bitBLT a source frame to a destination banded bitmap. The source and destination pixels are combined as specified by the current BOOL\_SD Boolean code. The destination bitmap parameters must have been previously defined by the SET\_BBMAP graphic order. The source frame warp is assumed to be the FW specified in the BLT2BB\_SD graphic order unless a non-zero source bitmap warp was previously defined by the SET\_SBMAP graphic order, in which case the latter is used.

DA specifies the logical bit address of the area or transfer frame that starts in the destination banded bitmap. The logical bit address is converted to a physical bit address by adding the PSUBL value set by the SET\_BBMAP graphic order. The start of the transfer frame must be within the bounds of the banded bitmap, but the end of the transfer frame may extend past the end of the bitmap. DA must point to the upper left corner of the transfer frame when the B2T flag is clear and to the lower left corner of the transfer frame when B2T is set. The warp of the destination banded bitmap is set by the SET\_BBMAP graphic order, which allows the bitmap to be packed or unpacked. The source physical bit address (SA) must point to the upper left corner of the source frame when the B2T flag is clear and to the lower left corner of the source frame when the B2T flag is set.

When a band fault is detected, the MC68322 rewrites the graphic order to update some of the parameters. The BAND number is incremented (or decremented when the B2T flag is set), DA and SA are repositioned to the starting pixel of the respective frame to be processed in the next band, and FH is written back with the number of remaining scanlines in the frame to be transferred.

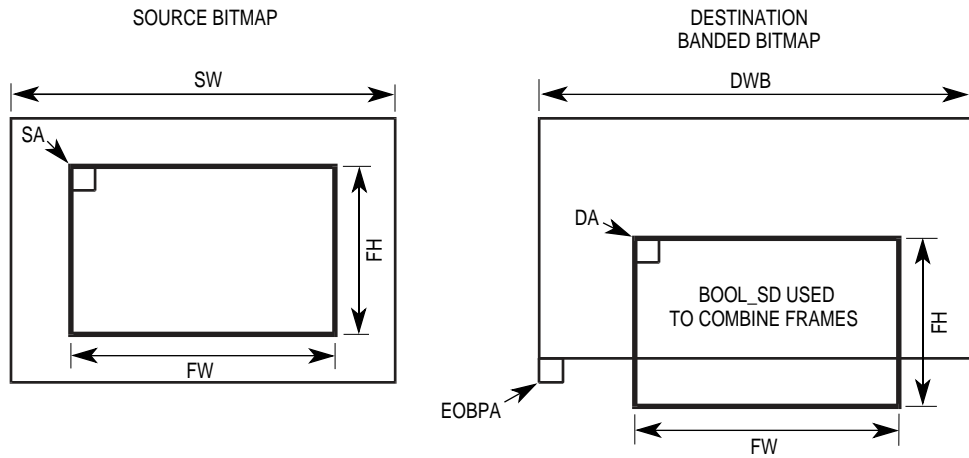
### Related Graphic Orders

SET\_BOOL\_SD

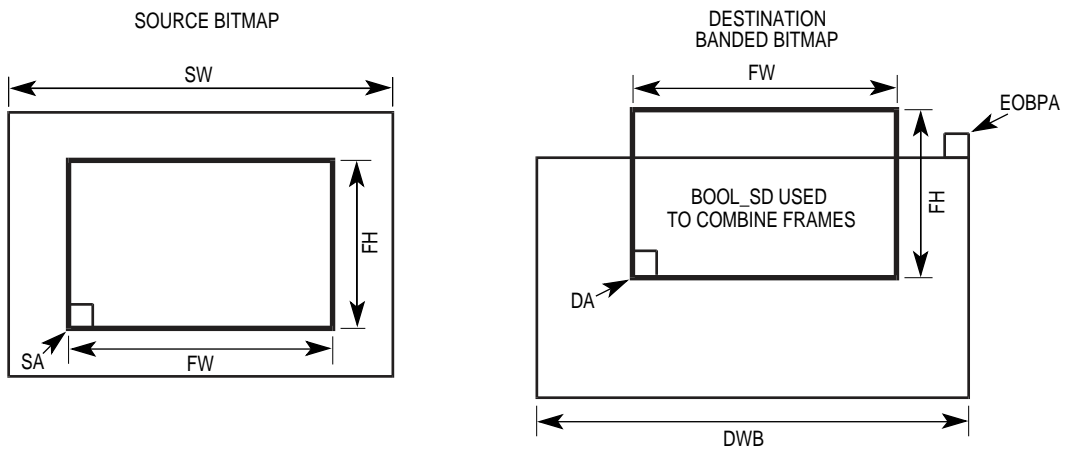
SET\_BBMAP

SET\_SBMAP





**Figure 13-5. Source/Destination bitBLT to Banded Bitmap—0° Page**



**Figure 13-6. Source/Destination bitBLT to Banded Bitmap—180° Page**

## BLT2BB\_SHD

### Source/Halftone/Destination bitBLT to Banded Bitmap

PARAMETERS	SIZE	DESCRIPTION
0x33	Byte	BLT2BB_SHD Opcode.
BAND*	Byte	Band number when graphic order is executed.
DA*	Long Word	Destination logical bit address.
FW	Word	Frame width in bits.
FH*	Word	Frame height in scanlines.
SA*	Long Word	Source physical bit address.
HXR	Word	Half-tone X remainder.
HYR*	Word	Half-tone Y remainder.
HA*	Long Word	Half-tone physical bit address of the starting pixel.

Note: \* Denotes A Parameter That The MC68322 Updates When The Frame Crosses A Band Boundary.

The BLT2BB\_SHD graphic order causes the MC68322 to bitBLT a source frame to a destination banded bitmap and apply a half-tone bitmap in the process. The source, half-tone, and destination pixels are combined, as specified by the current BOOL\_SHD Boolean code. The destination bitmap parameters must have been previously defined by the SET\_BBMAP graphic order.

The source frame warp is assumed to be the FW specified in the BLT2BB\_SHD graphic order, unless a non-zero source bitmap warp was previously defined by the SET\_SBMAP graphic order, in which case the latter is used. The half-tone bitmap dimensions must have been previously defined by the SET\_HTBMAP graphic order. During the processing of half-tones, wrapping occurs at the edges of the bitmap and this results in horizontal and vertical replication (tiling) of the bitmap to cover the entire bitBLT frame area.

Half-tone tiled patterns are typically anchored to the page. Thus, a bitBLT may need to take on the half-tone pattern starting at various points in the half-tone bitmap depending on where it is being positioned on the page. The half-tone parameters HXR, HYR, and HA define the precise half-tone pixel that corresponds to the upper left (or lower left, when the B2T flag is set) corners of the source and destination frames. HXR specifies the number of pixels remaining to the right edge of the bitmap and HYR defines the number of scanlines remaining to the bottom edge (or top edge, when the B2T flag is set). HXR and HYR must be in the following ranges— $1 \leq HXR \leq HW$  and  $1 \leq HYR \leq HH$ , where HW and HH are the width and height of the half-tone bitmap, respectively.

For example, when the B2T flag is clear, if the starting pixel in the half-tone bitmap is determined to be at the upper left, HXR must be set to HW, and HYR to HH. If, instead, a 180° page is being rendered and the B2T flag is set, HXR must still be set to HW, but HYR must be set to one.

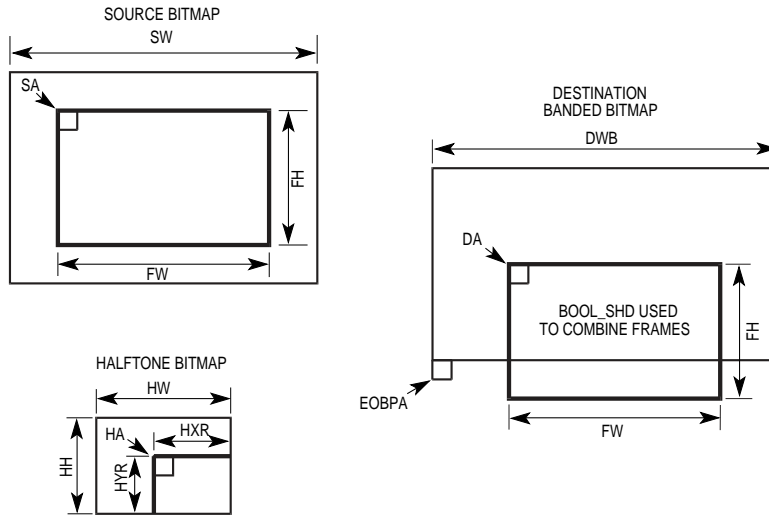
DA specifies the logical bit address of the area or transfer frame that begins in the destination banded bitmap. The logical bit address is converted to a physical bit address, by adding the PSUBL value set by the SET\_BBMAP graphic order. The start of the transfer frame must be within the bounds of the banded bitmap, but the end of the transfer frame may extend past the end of the bitmap.

DA must point to the upper left corner of the transfer frame when the B2T flag is clear and to the lower left corner of the transfer frame when B2T is set. The warp of the destination banded bitmap is set by the SET\_BBMAP graphic order, which allows the bitmap to be packed or unpacked. The SA must point to the upper left corner of the source frame when the B2T flag is clear and to the lower left corner of the source frame when the B2T flag is set.

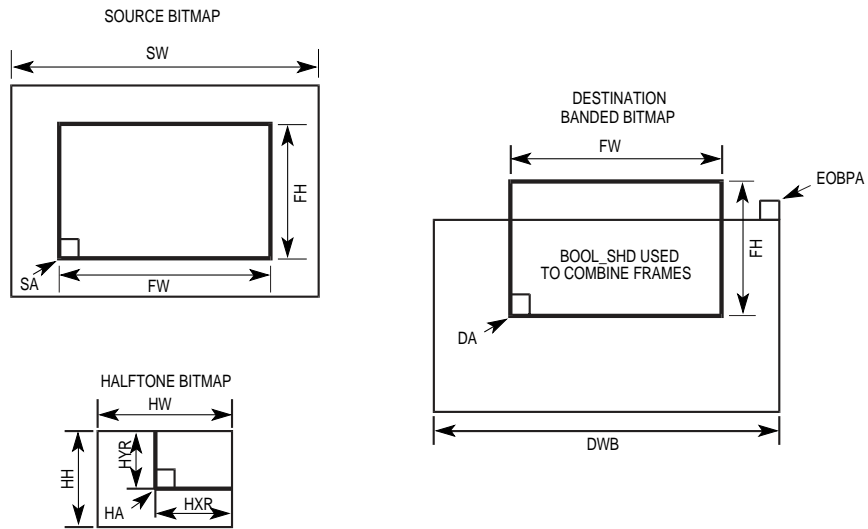
When a band fault is detected, the MC68322 rewrites the graphic order to update some of its parameters. The BAND number is incremented (or decremented when the B2T flag is set). DA, SA, and HA are repositioned to the starting pixel of the respective frame to be processed in the next band. Lastly, FH and HYR are written back with the number of remaining scanlines in their respective frames to be transferred.

### **Related Graphic Orders**

- SET\_BOOL\_SHD
- SET\_BBMAP
- SET\_HTBMAP
- SET\_SBMAP



**Figure 13-7. Source/Halftone/Destination bitBLT to Banded Bitmap—0° Page**



**Figure 13-8. Source/Halftone/Destination bitBLT to Banded Bitmap—180° Page**

## BLT2BB\_XD

### Expanded Source, Destination bitBLT to Banded Bitmap

PARAMETERS	SIZE	DESCRIPTION
0x36	Byte	BLT2BB_XD Opcode.
BAND*	Byte	Band number when graphic order is executed.
DA*	Long Word	Destination logical bit address.
FW	Word	Destination frame width in bits.
FH*	Word	Destination frame height in scanlines.
SA*	Long Word	Unexpanded source physical bit address.
SW	Word	Unexpanded source frame warp in bits.
XOFF:XMUL	Two 4-bit Fields	X offset and X multiplier in bits.
YOFF:YMUL	Two 4-bit Fields	Y offset and Y multiplier in scanlines.

Note: \* Denotes A Parameter That The MC68322 Updates When The Frame Crosses A Band Boundary.

The BLT2BB\_XD graphic order transfers a low resolution source frame to a destination banded bitmap. Before being combined with the destination, the source frame is scaled to match the resolution of the destination bitmap, which results in an intermediate expanded source bitmap. The pixels of the expanded source and destination bit maps are combined, as specified by the Boolean code set in the last SET\_BOOL\_SD order. The destination bitmap parameters must have been previously defined by the SET\_BBMAP graphic order. DA specifies the logical bit address of the area or transfer frame that begins in the destination banded bitmap. The logical bit address is converted to a physical bit address by adding the PSUBL value set by the SET\_BBMAP graphic order. The start of the transfer frame must be within the bounds of the banded bitmap, but the end of the transfer frame may extend past the end of the bitmap. DA must point to the upper left corner of the transfer frame when the B2T flag is clear and to the lower left corner of the transfer frame when B2T is set. The warp of the destination banded bitmap is set by the SET\_BBMAP graphic order, which allows the bitmap to be packed or unpacked.

The FW and FH parameters define the area of the destination bitmap on which the operation is performed. FW is the frame width in bits and at a maximum equals the quantity  $W \times (XMUL + 1)$  where W is the width of the unexpanded source bitmap. FH is the frame height in scanlines and at a maximum equals the quantity  $H \times (YMUL + 1)$  where H is the height of the unexpanded source bitmap. Specifying FW and FH as defined above causes the entire expanded source image to be combined with the destination bitmap. Specifying an FW and/or FH value less than the values defined by the above equations causes only a portion of the expanded source frame to be applied to the destination. When used in combination with XOFF and YOFF, clipping can be affected at any or all extents of the expanded source bitmap.

The SA parameter defines the unexpanded source bitmap bit address. It must point to the upper left corner of the bitmap when the B2T flag is clear and to the lower left corner of the bitmap when the B2T flag is set. The warp of the unexpanded bitmap is set by the source width operand. This value is added to SA to locate the beginning of each successive scanline. Note that the source warp set by the SET\_SBMAP graphic order has no effect on this graphic order.

The XOFF:XMUL parameter is divided into two fields. The least-significant four bits contain the XMUL field, which specifies the factor used to scale the unexpanded source bitmap in the X dimension. XMUL must equal a specific value from 0–15, which represents a scaling factor of 1–16. Only certain scaling factors are supported, as defined in Table 13-3. Values other than those listed are ignored and no X scaling is performed.

**Table 13-3. Supported Scaling Factors**

EXPANSION FACTOR	XMUL VALUE
1	0
2	1
3	2
4	3
6	5
8	7
12	11
16	15

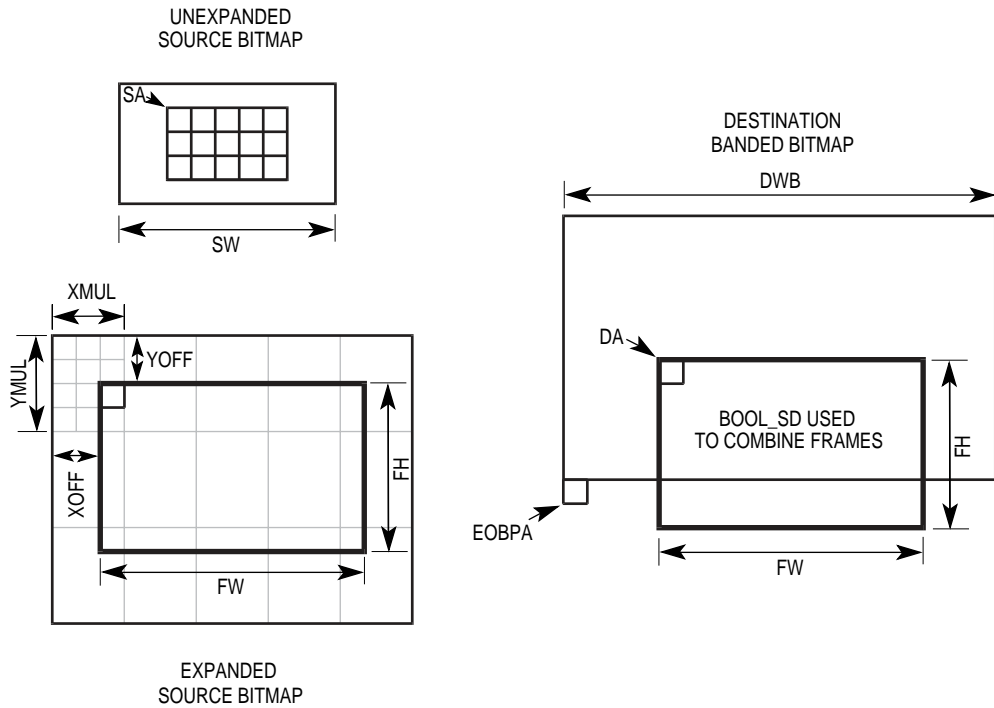
The XOFF field occupies the four most-significant bits and indicates the number of bits to be clipped at the left edge of the expanded source bitmap. XOFF ranges from 0–XMUL. If XOFF is zero, no clipping occurs at the left extent, but if it is non-zero, XOFF number of bits in the left edge of the expanded bitmap are skipped and the next bit is the first one transferred to the destination bitmap. The YOFF:YMUL parameter is also divided into two fields. The least-significant four bits contain the YMUL field, which specifies the factor used to scale the unexpanded source bitmap in the Y dimension. YMUL can equal any value from 0–15, which represents a scaling factor of 1–16. The YOFF field occupies the four most-significant bits and indicates the number of scanlines to be clipped at the top or bottom edge of the expanded source bitmap, depending on the value of the B2T flag. When B2T is clear, YOFF is defined from the top edge of the source bitmap. When B2T is set, YOFF is defined from the bottom edge. YOFF ranges from 0–YMUL. If YOFF is zero, no clipping occurs at the top or bottom extent, but if it is non-zero, YOFF number of scanlines at the top or bottom edge of the expanded bitmap are skipped and the next scanline is the first line transferred to the destination bitmap.

A band fault is detected when the transfer frame extends past the end of the destination bitmap, which is defined by the EOBPA operand in the SET\_BBMAP graphic order. When a band fault is detected, the MC68322 rewrites the graphic order to update its operands. The BAND number is incremented (or decremented, when the B2T flag is set), DA and SA are repositioned to the starting pixel of the respective frame to be processed in the next band, and YOFF is updated according to the current position in the expanded source bitmap. Lastly, FH is written back with the number of scanlines in the frame to be transferred.

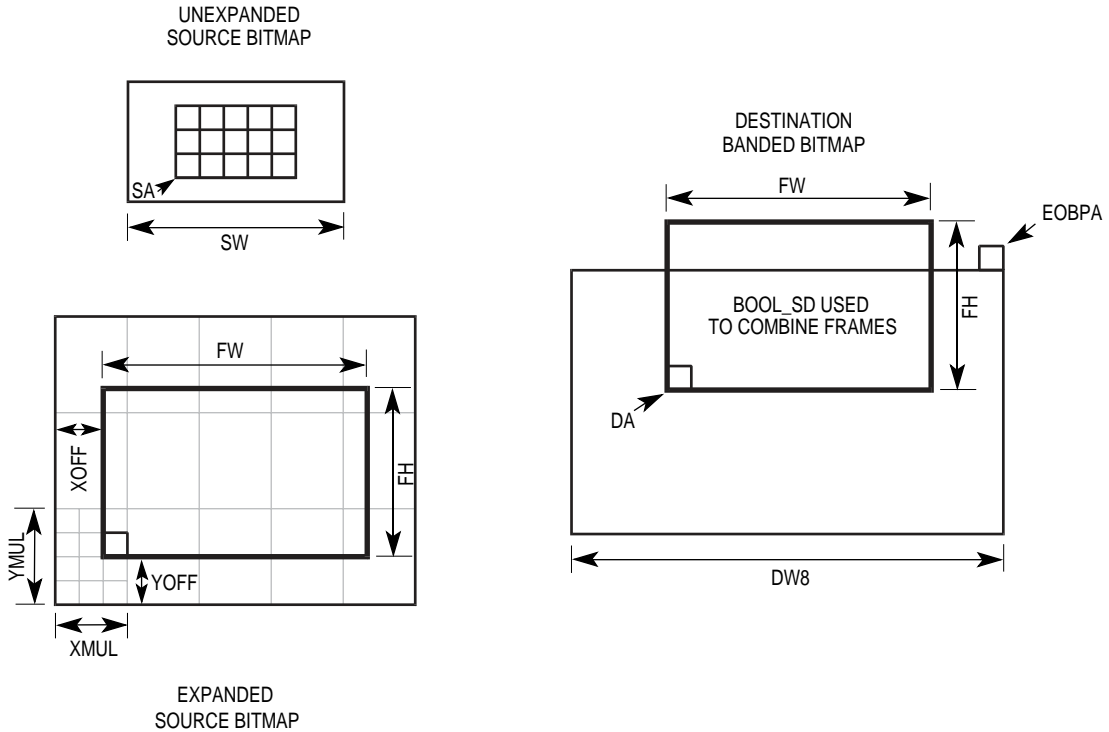
### Related Graphic Orders

SET\_BOOL\_SD

SET\_BBMAP



**Figure 13-9. Expanded Source, Destination bitBLT To Banded Bitmap, 0° Page**



**Figure 13-10. Expanded Source, Destination bitBLT To Banded Bitmap, 180° Page**

## BLT2BB\_XHD

### Expanded Source, Halftone, Destination bitBLT to Banded Bitmap

PARAMETERS	SIZE	DESCRIPTION
0x37	Byte	BLT2BB_XHD Opcode.
BAND*	Byte	Band number when graphic order is executed.
DA*	Long Word	Destination logical bit address.
FW	Word	Destination frame width in bits.
FH*	Word	Destination frame height in scanlines.
SA*	Long Word	Unexpanded source physical bit address.
SW	Word	Unexpanded source frame warp in bits.
XOFF:XMUL	Two 4-bit Fields	X offset and X multiplier in bits.
YOFF:YMUL	Two 4-bit Fields	Y offset and Y multiplier in scanlines.
HXR	Word	Halftone X remainder.
HYR*	Word	Halftone Y remainder.
HA*	Long Word	Halftone physical bit address of the starting pixel.

Note: \* Denotes A Parameter That The MC68322 Updates When The Frame Crosses A Band Boundary.

The BLT2BB\_XHD graphic order transfers a low resolution source frame to a destination banded bitmap and applies a halftone bitmap in the process. Before being combined with the destination, the source frame is scaled to match the resolution of the destination bitmap. This results in an intermediate expanded source bitmap. The pixels of the expanded source, halftone, and destination bit maps are combined as specified by the Boolean code set in the last SET\_BOOL\_SHD order. The destination bitmap parameters must have been previously defined by the SET\_BBMAP graphic order and the halftone bitmap dimensions by the SET\_HTBMAP graphic order. During the processing of halftones, wrapping occurs at the edges of the bitmap. This results in horizontal and vertical replication (tiling) of the bitmap to cover the entire destination frame area.

The DA parameter specifies the logical bit address of the area or transfer frame that starts in the destination banded bitmap. The logical bit address is converted to a physical bit address by adding the PSUBL value in the SET\_BBMAP graphic order. The start of the transfer frame must be within the bounds of the banded bitmap, but the end of the transfer frame may extend past the end of the bitmap. DA must point to the upper left corner of the transfer frame when the B2T flag is clear and to the lower left corner of the transfer frame when B2T is set. The warp of the destination banded bitmap is set by the SET\_BBMAP graphic order, which allows the bitmap to be packed or unpacked.

The FW and FH parameters define the area of the destination bitmap on which the operation is performed. FW is the frame width in bits and at a maximum equals the quantity  $W \times (XMUL + 1)$  where W is the width of the unexpanded source bitmap. FH is the frame height in scanlines and at a maximum equals the quantity  $H \times (YMUL + 1)$  where H is the height of the unexpanded source bitmap. Specifying FW and FH as defined above causes the entire expanded source image to be combined with the destination bitmap. Specifying an FW and/or FH value less than the values defined by the above equations causes only a portion of the expanded source frame to be applied to the destination. When used in combination with XOFF and YOFF, clipping can be affected at any or all extents of the expanded source bitmap.



The SA parameter defines the unexpanded source bitmap bit address. It must point to the upper left corner of the bitmap when the B2T flag is clear and to the lower left corner of the bitmap when the B2T flag is set. The warp of the unexpanded bitmap is set by the SW parameter. This value is added to SA to locate the beginning of each successive scanline. Notice that the source warp set by the SET\_SBMAP graphic order has no effect on this graphic order.

The XOFF:XMUL operand is divided into two fields. The least-significant four bits contain the XMUL field, which specifies the factor used to scale the unexpanded source bitmap in the X dimension. XMUL must equal a specific value from 0–15, which represents a scaling factor of 1–16. Only certain scaling factors are supported, as defined in Table 13-3. Values other than those listed are ignored and no X scaling is performed. The XOFF field occupies the four most-significant bits and indicates the number of bits to be clipped at the left edge of the expanded source bitmap. XOFF ranges from 0–XMUL. If XOFF is zero, no clipping occurs at the left extent, but if it is non-zero, XOFF number of bits in the left edge of the expanded bitmap are skipped and the next bit is the first bit transferred to the destination bitmap. The YOFF:YMUL operand is also divided into two fields. The least-significant four bits contain the YMUL field, which specifies the factor used to scale the unexpanded source bitmap in the Y dimension. YMUL can equal any value from 0–15, which represents a scaling factor of 1–16.

The YOFF field occupies the four most-significant bits in the YOFF:YMUL operand. YOFF indicates the number of scanlines to be clipped at the top or bottom edge of the expanded source bitmap, depending on the value of the B2T flag. When B2T is clear, YOFF is defined from the top edge of the source bitmap. When B2T is set, YOFF is defined from the bottom edge. YOFF ranges from 0–YMUL. If YOFF is zero, no clipping occurs at the top or bottom extent, but if it is non-zero, YOFF number of scanlines at the top or bottom edge of the expanded bitmap are skipped and the next scanline is the first one transferred to the destination bitmap.

Halftone tiled patterns are typically anchored to the page. Thus, a bitBLT may need to take on the halftone pattern starting at various points in the halftone bitmap depending on where it is positioned on the page. The halftone parameters HXR, HYR, and HA define the precise halftone pixel that corresponds to the upper left (or lower left, when the B2T flag is set) corners of the source and destination frames. HXR specifies the number of pixels remaining to the right edge of the bitmap and HYR defines the number of scanlines remaining to the bottom edge (or top edge, when the B2T flag is set).

HXR and HYR must be in the following ranges— $1 \leq \text{HXR} \leq \text{HW}$  and  $1 \leq \text{HYR} \leq \text{HH}$ , where HW and HH are the width and height of the halftone bitmap, respectively. For example, when the B2T flag is clear, if the starting pixel in the halftone bitmap is determined to be at the upper left, HXR must be set to HW and HYR to HH. If, instead, a 180° page is rendered and the B2T flag is set, HXR must still be set to HW, but HYR must be set to one. The HA parameter defines a physical bit address within the halftone bitmap. HA must point to the upper left corner of the transfer frame when the B2T flag is clear and to the lower left corner of the frame when the B2T flag is set. Also, HA must be consistent with HXR and HYR.

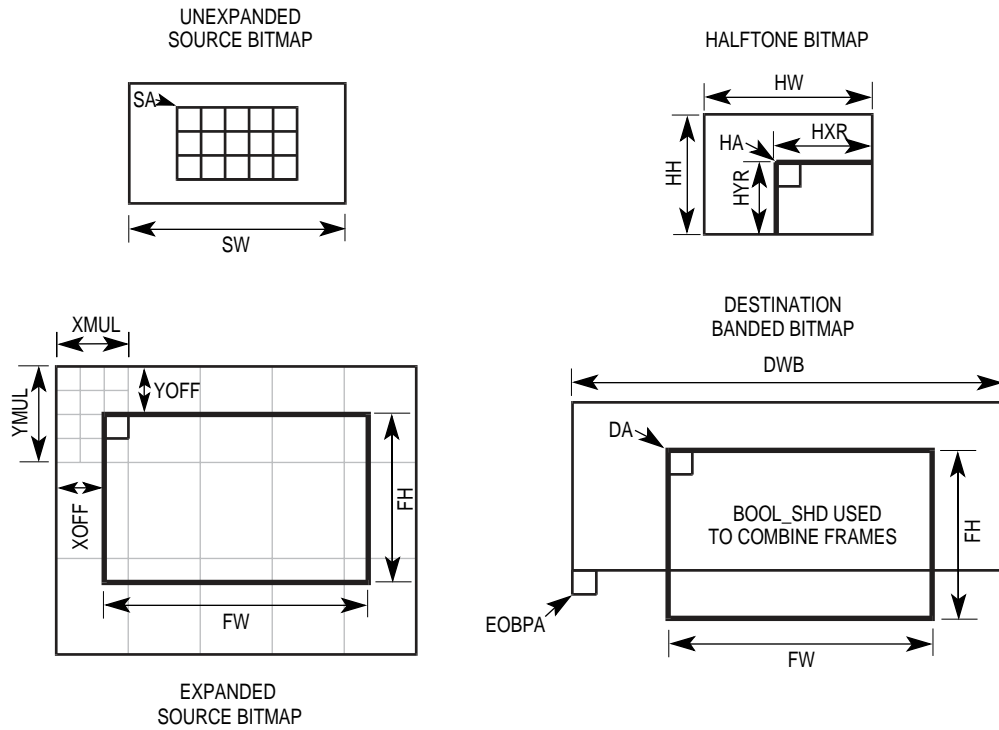
A band fault is detected when the transfer frame extends past the end of the destination bitmap which is defined by the EOBPA operand in the SET\_BBMAP graphic order. When a band fault is detected, the MC68322 rewrites the graphic order to update its operands. The BAND number is incremented (or decremented, when the B2T flag is set), DA, SA, and HA are repositioned to the starting pixel of the respective frame to be processed in the next band and YOFF is updated according to the current position in the expanded source bitmap. Lastly, FH and HYR are written back with the number of remaining scanlines in their respective frames to be transferred.

### **Related Graphic Orders**

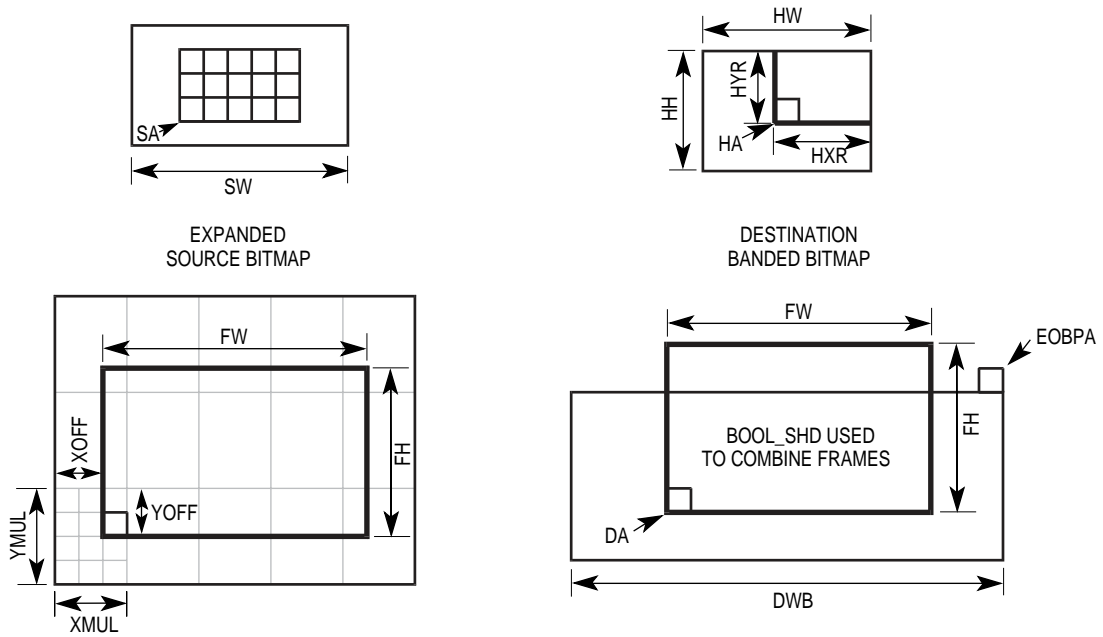
SET\_BOOL\_SHD

SET\_BBMAP

SET\_HTBMAP



**Figure 13-11. Expanded Source, Halftone, Destination bitBLT To Banded Bitmap, 0° Page**



**Figure 13-12. Expanded Source, Halftone, Destination bitBLT To Banded Bitmap, 180° Page**

## BLT2F\_D

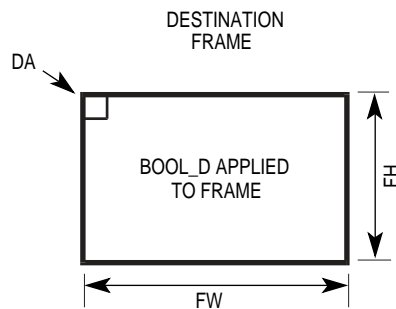
### Destination Only bitBLT to Frame

PARAMETERS	SIZE	DESCRIPTION
0x10	Byte	BLT2F_D Opcode.
0x00	Byte	Reserved.
DA	Long Word	Destination physical bit address.
FW	Word	Frame width in bits.
FH	Word	Frame height in scanlines.

The BLT2F\_D graphic order causes the MC68322 to modify a destination frame bitmap. The destination pixels are manipulated as specified by the current BOOL\_D Boolean code. The destination bitmap warp is taken to be the FW specified in the BLT2F\_D graphic order. The destination physical bit address (DA) must point to the upper left corner of the frame.

### Related Graphic Orders

#### SET\_BOOL\_D



**Figure 13-13. Destination bitBLT to Frame**

## BLT2F\_SD

### Source/Destination bitBLT to Frame

PARAMETERS	SIZE	DESCRIPTION
0x12	Byte	BLT2F_SD Opcode.
0x00	Byte	Reserved.
DA	Long Word	Destination physical bit address.
FW	Word	Frame width in bits.
FH	Word	Frame height in scanlines.
SA	Long Word	Source physical bit address.

The BLT2F\_SD graphic order causes the MC68322 to bitBLT a source frame to a destination frame. The source and destination pixels are combined as specified by the current BOOL\_SD Boolean code.

The destination bitmap warp is taken to be the FW specified in the BLT2F\_SD graphic order. FW is also assumed to be the source frame warp, unless a non-zero source bitmap warp was previously defined by the SET\_SBMAP graphic order, in which case the latter is used. The DA and SA parameters must point to the upper left corners of their respective frames.

### Related Graphic Orders

SET\_BOOL\_SD

SET\_SBMAP

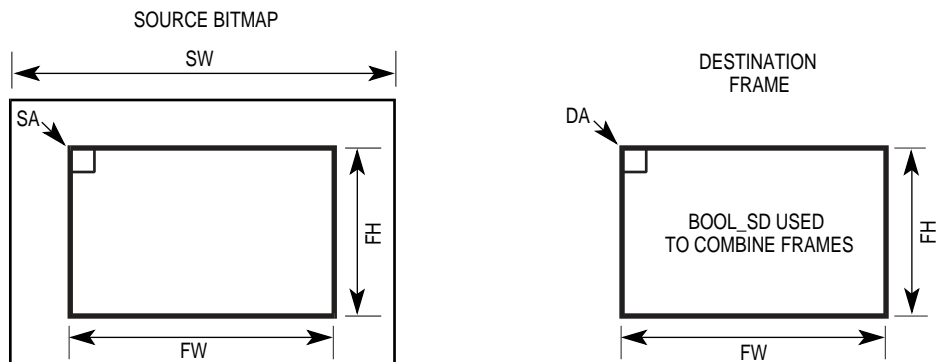


Figure 13-14. Source/Destination bitBLT to Frame

## BLT2F\_SHD

### Source/Halftone/Destination bitBLT to Frame

PARAMETERS	SIZE	DESCRIPTION
0x13	Byte	BLT2F_SHD Opcode.
0x00	Byte	Reserved.
DA	Long Word	Destination physical bit address.
FW	Word	Frame width in bits.
FH	Word	Frame height in scanlines.
SA	Long Word	Source physical bit address.
HXR	Word	Halftone X remainder.
HYR	Word	Halftone Y remainder.
HA	Long Word	Halftone physical bit address of the starting pixel.

The BLT2F\_SHD graphic order causes the MC68322 to bitBLT a source frame to a destination frame and apply a halftone bitmap in the process. The source, halftone, and destination pixels are combined as specified by the current BOOL\_SHD Boolean code. The destination bitmap warp is taken to be the FW specified in the BLT2F\_SHD graphic order. FW is also assumed to be the source frame warp, unless a non-zero source bitmap warp was previously defined by the SET\_SBMAP graphic order, in which case the latter is used. The SET\_HTBMAP graphic order must previously define the halftone bitmap dimensions. During the processing of halftones, wrapping occurs at the edges of the bitmap and results in horizontal and vertical replication (tiling) of the bitmap to cover the entire bitBLT frame area.

Halftone tiled patterns are typically anchored to the page. A bitBLT may need to take on the halftone pattern starting at various points in the halftone bitmap, depending on where it is positioned on the page. The halftone parameters HXR, HYR, and HA define the precise halftone pixel that corresponds to the upper left corners of the source and destination frames. HXR specifies the number of pixels remaining to the right edge of the bitmap and HYR defines the number of scanlines remaining to the bottom edge. HXR and HYR must be in the following ranges:  $1 \leq HXR \leq HW$  and  $1 \leq HYR \leq HH$ , where HW and HH are the width and height of the halftone bitmap, respectively. For example, if the starting pixel in the halftone bitmap is determined to be at the upper left, HXR must be set to HW, and HYR to HH; to start at the lower right, HXR and HYR must both be set to one. The DA, SA, and HA parameters must point to the upper left corners of their respective frames.

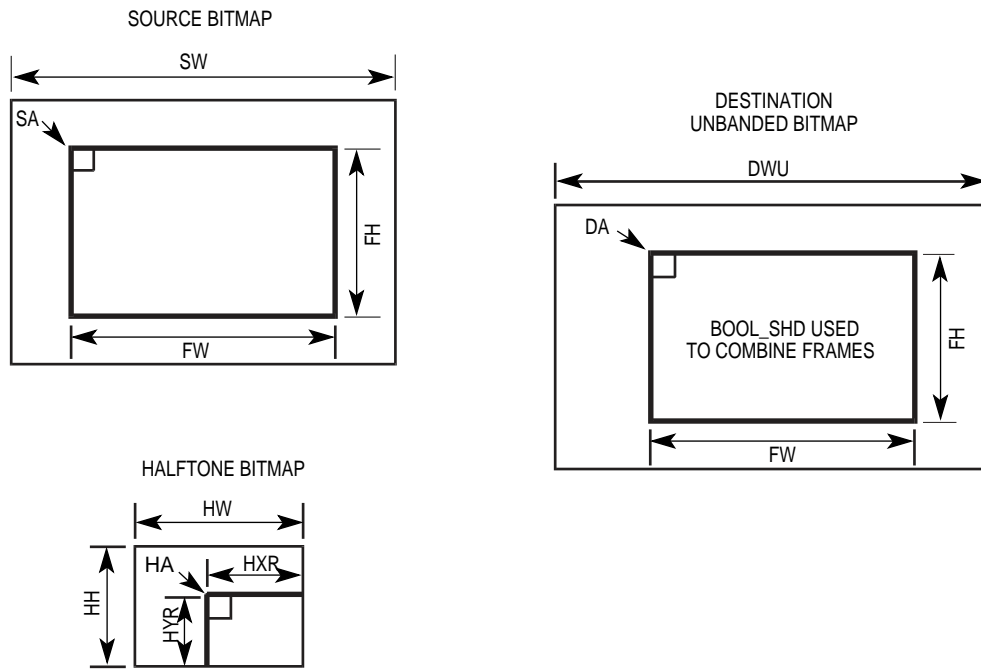
### Related Graphic Orders

SET\_BOOL\_SHD

SET\_HTBMAP

SET\_SBMAP

B



**Figure 13-15. Source/Halftone/  
Destination bitBLT to Frame**

## BLT2F\_XD

### Expanded Source, Destination bitBLT to Frame

PARAMETERS	SIZE	DESCRIPTION
0x16	Byte	BLT2F_XD Opcode.
0x00	Byte	Reserved.
DA	Long Word	Destination physical bit address.
FW	Word	Destination frame width in bits.
FH	Word	Destination frame height in scanlines.
SA	Long Word	Unexpanded source physical bit address.
SW	Word	Unexpanded source frame warp in bits.
XOFF:XMUL	Two 4-bit Fields	X offset and X multiplier in bits.
YOFF:YMUL	Two 4-bit Fields	Y offset and Y multiplier in scanlines.

The BLT2F\_XD graphic order transfers a low resolution source frame to a destination frame. Before being combined with the destination, the source frame is scaled to match the resolution of the destination bitmap. This results in an intermediate expanded source bitmap. The pixels of the expanded source and destination bit maps are combined as specified by the Boolean code set in the last SET\_BOOL\_SD order.

The DA parameter specifies the physical bit address of the destination frame. It must point to the upper left corner of the destination frame. The destination frame will be packed since its warp is assumed to equal FW. The FW and FH parameters define the area of the destination bitmap on which the operation is performed. FW is the frame width in bits and at a maximum equals the quantity  $W \times (XMUL + 1)$  where W is the width of the unexpanded source bitmap. FH is the frame height in scanlines and at a maximum equals the quantity  $H \times (YMUL + 1)$  where H is the height of the unexpanded source bitmap. Specifying FW and FH as defined above causes the entire expanded source image to be combined with the destination bitmap. Specifying an FW and/or FH value less than the values defined by the above equations causes only a portion of the expanded source frame to be applied to the destination. When used in combination with XOFF and YOFF, clipping can be affected at any or all extents of the expanded source bitmap.

The SA parameter defines the unexpanded source bitmap bit address. It must point to the upper left corner of the bitmap. The warp of the bitmap is set by the SW operand. This value is added to SA to locate the beginning of each successive scanline. Note that the source warp set by the SET\_SBMAP graphic order has no effect on this graphic order.

The XOFF:XMUL operand is divided into two fields. The least-significant four bits contain the XMUL field, which specifies the factor used to scale the unexpanded source bitmap in the X dimension. XMUL must equal a specific value from 0–15, which represents a scaling factor of 1–16. Only certain scaling factors are supported, as defined in Table 13-3. Values other than those listed are ignored and no X scaling is performed.



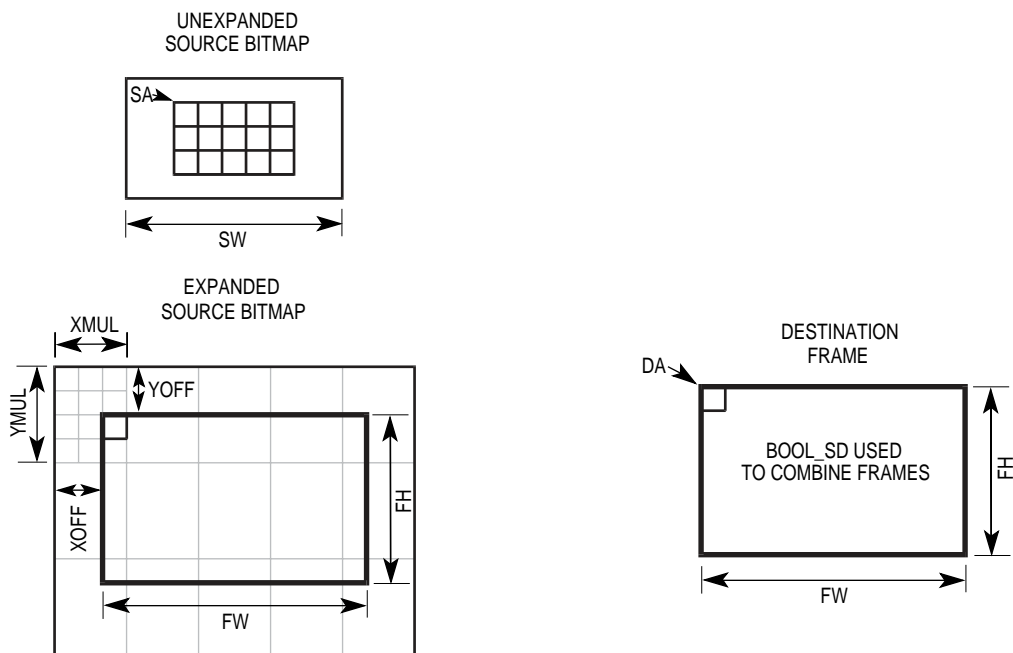
The XOFF field occupies the four most-significant bits and indicates the number of bits to be clipped at the left edge of the expanded source bitmap. XOFF ranges from 0–XMUL. If XOFF is zero, no clipping occurs at the left extent, but if it is non-zero, XOFF number of bits in the left edge of the expanded bitmap are skipped and the next bit is the first one transferred to the destination bitmap.

The YOFF:YMUL parameter is also divided into two fields. The least-significant four bits contain the YMUL field, which specifies the factor used to scale the unexpanded source bitmap in the Y dimension. YMUL can equal any value from 0–15, which represents a scaling factor of 1–16.

The YOFF field occupies the four most-significant bits and indicates the number of scanlines to be clipped at the top edge of the expanded source bitmap. YOFF ranges from 0–YMUL. If YOFF is zero, no clipping occurs at the top or bottom extent, but if it is non-zero, YOFF number of scanlines at the top edge of the expanded bitmap are skipped and the next scanline is the first one transferred to the destination bitmap.

**Related Graphic Orders**

`SET_BOOL_SD`



**Figure 13-16. Expanded Source, Destination bitBLT To Frame**

## BLT2F\_XHD

### Expanded Source, Halftone, Destination bitBLT to Banded Bitmap

PARAMETERS	SIZE	DESCRIPTION
0x17	Byte	BLT2F_XHD Opcode.
0x00	Byte	Reserved.
DA	Long Word	Destination physical bit address.
FW	Word	Destination frame width in bits.
FH	Word	Destination frame height in scanlines.
SA	Long Word	Unexpanded source physical bit address.
SW	Word	Unexpanded source frame warp in bits.
XOFF:XMUL	Two 4-bit Fields	X offset and X multiplier in bits.
YOFF:YMUL	Two 4-bit Fields	Y offset and Y multiplier in scanlines.
HXR	Word	Halftone X remainder.
HYR	Word	Halftone Y remainder.
HA	Long Word	Halftone physical bit address of the starting pixel.

The BLT2F\_XHD graphic order transfers a low resolution source frame to a destination frame and applies a halftone bitmap in the process. Before being combined with the destination, the source frame is scaled to match the resolution of the destination bitmap, which results in an intermediate expanded source bitmap. The pixels of the expanded source, halftone, and destination bit maps are combined as specified by the Boolean code set in the last SET\_BOOL\_SHD order. The SET\_HTBMAP graphic order must previously define the halftone bitmap dimensions. During the processing of halftones, wrapping occurs at the edges of the bitmap and this results in horizontal and vertical replication (tiling) of the bitmap to cover the entire destination frame area. The DA parameter specifies the physical bit address of the destination frame. It must point to the upper left corner of the destination frame. The destination frame will be packed since its warp is assumed to equal FW.

The FW and FH parameters define the area of the destination bitmap on which the operation is performed. FW is the frame width in bits and at a maximum equals the quantity  $W \times (XMUL + 1)$  where W is the width of the unexpanded source bitmap. FH is the frame height in scanlines and at a maximum equals the quantity  $H \times (YMUL + 1)$  where H is the height of the unexpanded source bitmap. Specifying FW and FH as defined above causes the entire expanded source image to be combined with the destination bitmap. Specifying an FW and/or FH value less than the values defined by the above equations causes only a portion of the expanded source frame to be applied to the destination. When used in combination with XOFF and YOFF, clipping can be affected at any or all extents of the expanded source bitmap.

The SA parameter defines the unexpanded source bitmap bit address. It must point to the upper left corner of the bitmap. The warp of the bitmap is set by the SW parameter. This value is added to SA to locate the beginning of each successive scanline. Note that the source warp set by the SET\_SBMAP graphic order has no effect on this graphic order.

The XOFF:XMUL parameter is divided into two fields. The least-significant four bits contain the XMUL field, which specifies the factor used to scale the unexpanded source bitmap in the X dimension. XMUL must equal a specific value from 0–15, which represents a scaling factor of 1–16. Only certain scaling factors are supported, as defined in Table 13-3. Values other than those listed are ignored and no X scaling is performed.

The XOFF field occupies the four most-significant bits and indicates the number of bits to be clipped at the left edge of the expanded source bitmap. XOFF ranges from 0–XMUL. If XOFF is zero, no clipping occurs at the left extent, but if it is non-zero, XOFF number of bits in the left edge of the expanded bitmap are skipped and the next bit is the first one transferred to the destination bitmap. The YOFF:YMUL parameter is also divided into two fields. The least-significant four bits contain the YMUL field, which specifies the factor used to scale the unexpanded source bitmap in the Y dimension. YMUL can equal any value from 0–15, which represents a scaling factor of 1–16.

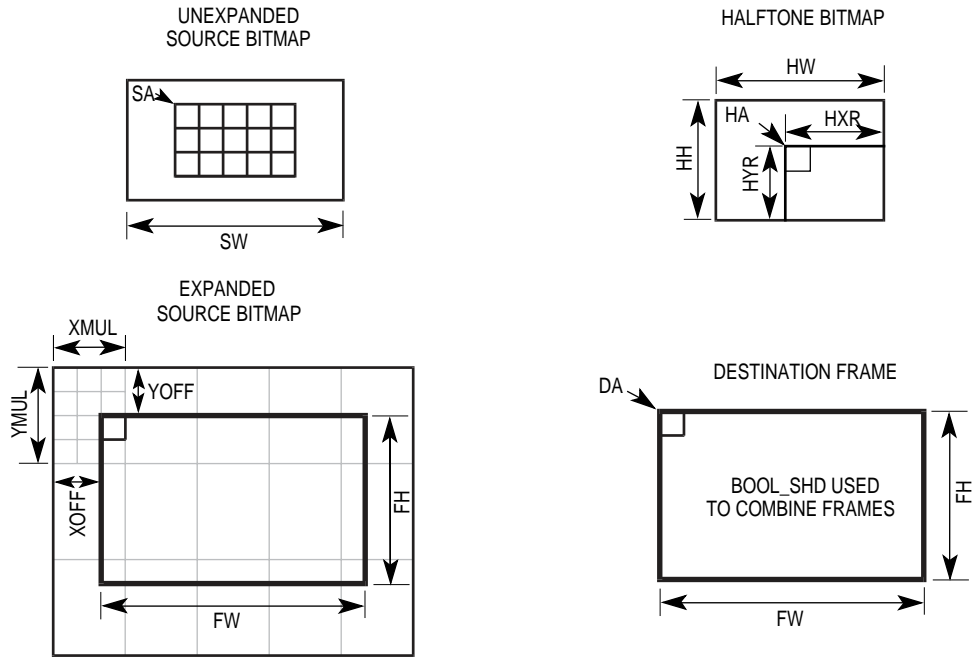
The YOFF field occupies the four most-significant bits and indicates the number of scanlines to be clipped at the top edge of the expanded source bitmap. YOFF ranges from 0–YMUL. If YOFF is zero, no clipping occurs at the top or bottom extent, but if it is non-zero, YOFF number of scanlines at the top edge of the expanded bitmap are skipped and the next scanline is the first one transferred to the destination bitmap.

Halftone tiled patterns are typically anchored to the page. A bitBLT may need to take on the halftone pattern starting at various points in the halftone bitmap, depending on where it is positioned on the page. The halftone parameters HXR, HYR, and HA define the precise halftone pixel that corresponds to the upper left corners of the source and destination frames. HXR specifies the number of pixels remaining to the right edge of the bitmap and HYR defines the number of scanlines remaining to the bottom edge. HXR and HYR must be in the following ranges:  $1 \leq HXR \leq HW$  and  $1 \leq HYR \leq HH$ , where HW and HH are the width and height of the halftone bitmap, respectively. For example, if the starting pixel in the halftone bitmap is determined to be at the upper left, HXR must be set to HW, and HYR to HH; to start at the lower right, HXR and HYR must be set to one. The HA parameter defines a physical bit address within the halftone bitmap and must point to the upper left corner of the transfer frame. HA must be consistent with HXR and HYR.

### Related Graphic Orders

SET\_BOOL\_SHD

SET\_HTBMAP



**Figure 13-17. Expanded Source, Halftone, Destination bitBLT To Frame**

# BLT2UB\_D

## Destination Only bitBLT to Unbanded Bitmap

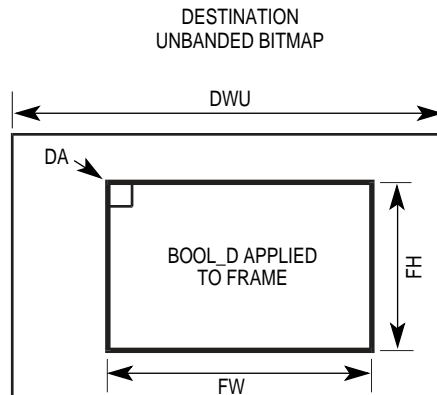
PARAMETERS	SIZE	DESCRIPTION
0x20	Byte	BLT2UB_D Opcode.
0x00	Byte	Reserved.
DA	Long Word	Destination physical bit address.
FW	Word	Frame width in bits.
FH	Word	Frame height in scanlines.

The BLT2UB\_D graphic order causes the MC68322 to modify the frame of a destination unbanded bitmap. The destination pixels are manipulated as specified by the current BOOL\_D Boolean code. The SET\_UBMAP graphic order must previously define the destination bitmap warp. The DA parameter must always point to the upper left corner of the frame.

### Related Graphic Orders

SET\_BOOL\_D

SET\_UBMAP



**Figure 13-18. Destination bitBLT to Unbanded Bitmap**

# BLT2UB\_SD

## Source/Destination bitBLT to Unbanded Bitmap

PARAMETERS	SIZE	DESCRIPTION
0x22	Byte	BLT2UB_SD Opcode.
0x00	Byte	Reserved.
DA	Long Word	Destination physical bit address.
FW	Word	Frame width in bits.
FH	Word	Frame height in scanlines.
SA	Long Word	Source physical bit address.

The BLT2UB\_SD graphic order causes the MC68322 to bitBLT a source frame to a destination unbanded bitmap. The source and destination pixels are combined as specified by the current value in the BOOL\_SD Boolean code register.

The SET\_UBMAP graphic order must previously define the destination bitmap warp. The source frame warp is assumed to be the FW specified in the BLT2UB\_SD graphic order unless a non-zero source bitmap warp was previously defined by the SET\_SBMAP graphic order, in which case the latter is used. The DA and SA parameters must point to the upper left corners of their respective frames.

### Related Graphic Orders

SET\_BOOL\_SD

SET\_SBMAP

SET\_UBMAP

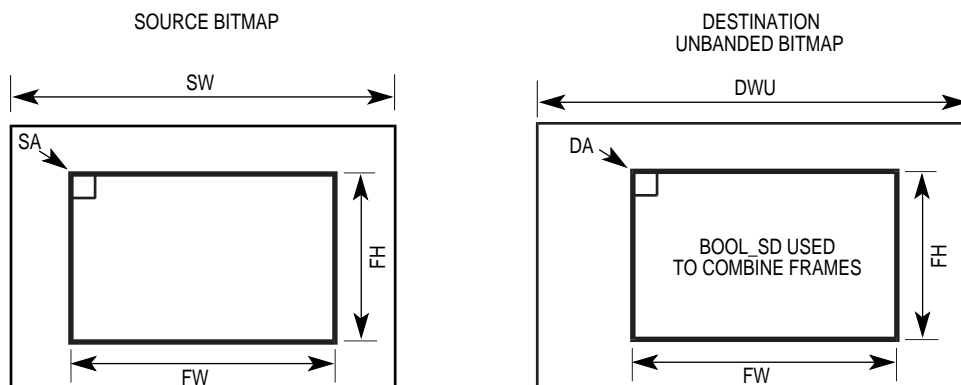


Figure 13-19. Source/Destination bitBLT to Unbanded Bitmap

## BLT2UB\_SHD

### Source/Halftone/Destination bitBLT to Unbanded Bitmap

PARAMETERS	SIZE	DESCRIPTION
0x23	Byte	BLT2UB_SHD Opcode.
0x00	Byte	Reserved.
DA	Long Word	Destination physical bit address.
FW	Word	Frame width in bits.
FH	Word	Frame height in scanlines.
SA	Long Word	Source physical bit address.
HXR	Word	Half-tone X remainder.
HYR	Word	Half-tone Y remainder.
HA	Long Word	Half-tone physical bit address of the starting pixel.

The BLT2UB\_SHD graphic order causes the MC68322 to bitBLT a source frame to a destination unbanded bitmap and apply a half-tone bitmap in the process. The source, half-tone, and destination pixels are combined as specified by the current value in the BOOL\_SHD Boolean code register.

The SET\_UBMAP graphic order must be previously define the destination bitmap warp. The source frame warp is assumed to be the FW specified in the BLT2UB\_SHD graphic order unless a non-zero source bitmap warp was previously defined by the SET\_SBMAP graphic order, in which case the latter is used. The SET\_HTBMAP graphic order must previously define the half-tone bitmap dimensions. During the processing of half-tones, wrapping occurs at the edges of the bitmap and results in horizontal and vertical replication (tiling) of the bitmap to cover the entire bitBLT frame area.

Half-tone tiled patterns are typically anchored to the page. Thus, a bitBLT may need to take on the half-tone pattern starting at various points in the half-tone bitmap, depending on where it is positioned on the page. The half-tone parameters HXR, HYR, and HA define the precise half-tone pixel that corresponds to the upper left corners of the source and destination frames. HXR specifies the number of pixels remaining to the right edge of the bitmap and HYR defines the number of scanlines remaining to the bottom edge. HXR and HYR must be in the following ranges:  $1 \leq HXR \leq HW$  and  $1 \leq HYR \leq HH$ , where HW and HH are the width and height of the half-tone bitmap, respectively. For example, if the starting pixel in the half-tone bitmap is determined to be at the upper left, HXR must be set to HW, and HYR to HH; to start at the lower right, HXR and HYR must both be set to one. The DA, SA, and HA parameters must point to the upper left corners of their respective frames.

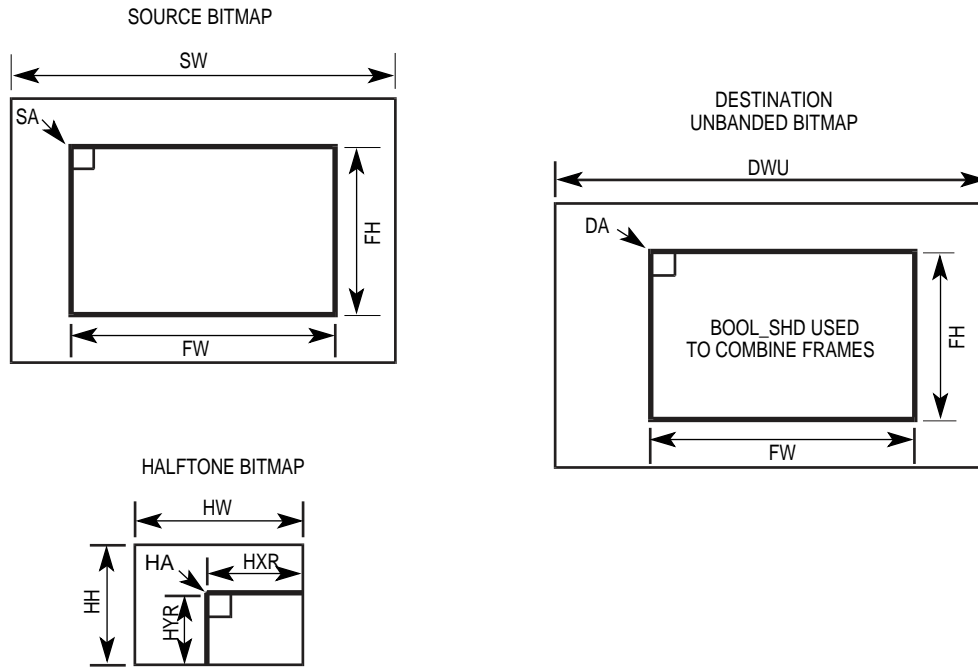
### Related Graphic Orders

SET\_BOOL\_SHD

SET\_HTBMAP

SET\_SBMAP

SET\_UBMAP



**Figure 13-20. Source/Halftone/Destination bitBLT to Unbanded Bitmap**



## BLT2UB\_XD

### Expanded Source, Destination bitBLT to Unbanded Bitmap

PARAMETERS	SIZE	DESCRIPTION
0x26	Byte	BLT2UB_XD Opcode.
0x00	Byte	Reserved.
DA	Long Word	Destination physical bit address.
FW	Word	Destination frame width in bits.
FH	Word	Destination frame height in scanlines.
SA	Long Word	Unexpanded source physical bit address.
SW	Word	Unexpanded source frame warp in bits.
XOFF:XMUL	Two 4-bit Fields	X offset and X multiplier in bits.
YOFF:YMUL	Two 4-bit Fields	Y offset and Y multiplier in scanlines.

The BLT2UB\_XD graphic order transfers a low resolution source frame to a destination unbanded bitmap. Before being combined with the destination, the source frame is scaled to match the resolution of the destination bitmap. This results in an intermediate expanded source bitmap. The pixels of the expanded source and destination bit maps are combined as specified by the Boolean code set in the last SET\_BOOL\_SD order. The SET\_UBMAP graphic order must previously define the destination bitmap warp.

The DA parameter specifies the physical bit address of the area or transfer frame within the destination unbanded bitmap. The entire transfer frame must be within the bounds of the banded bitmap. DA must point to the upper left corner of the transfer frame. The warp of the destination unbanded bitmap is set by the SET\_UBMAP graphic order, which allows the bitmap to be packed or unpacked.

The FW and FH parameters define the area of the destination bitmap on which the operation is performed. FW is the frame width in bits and at a maximum equals the quantity  $W \times (XMUL + 1)$  where W is the width of the unexpanded source bitmap. FH is the frame height in scanlines and at a maximum equals the quantity  $H \times (YMUL + 1)$  where H is the height of the unexpanded source bitmap. Specifying FW and FH as defined above causes the entire expanded source image to be combined with the destination bitmap. Specifying an FW and/or FH value less than the values defined by the above equations causes only a portion of the expanded source frame to be applied to the destination. When used in combination with XOFF and YOFF, clipping can be affected at any or all extents of the expanded source bitmap.

The SA parameter defines the unexpanded source bitmap bit address. It must point to the upper left corner of the bitmap. The warp of the bitmap is set by the SW operand. This value is added to SA to locate the beginning of each successive scanline. Note that the source warp set by the SET\_SBMAP graphic order has no effect on this graphic order.

The XOFF:XMUL parameter is divided into two fields. The least-significant four bits contain the XMUL field, which specifies the factor used to scale the unexpanded source bitmap in the X dimension. XMUL must equal a specific value from 0–15, which represents a scaling factor of 1–16. Only certain scaling factors are supported, as defined in Table 13-3. Values other than those listed are ignored and no X scaling is performed.

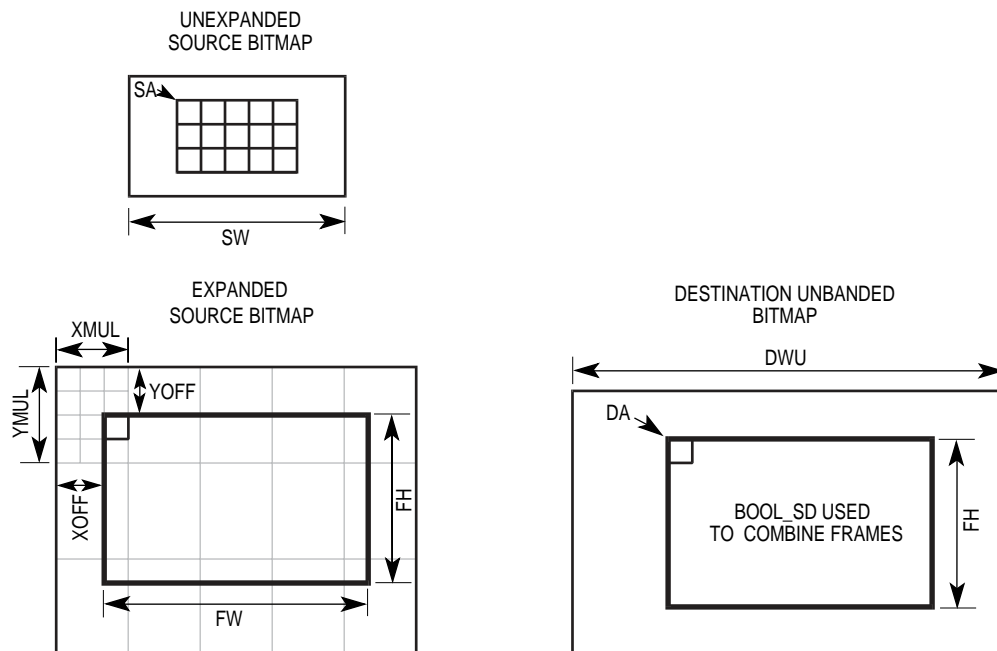
The XOFF field occupies the four most-significant bits and indicates the number of bits to be clipped at the left edge of the expanded source bitmap. XOFF ranges from 0–XMUL. If XOFF is zero, no clipping occurs at the left extent, but if it is non-zero, XOFF number of bits in the left edge of the expanded bitmap are skipped and the next bit is the first one transferred to the destination bitmap. The YOFF:YMUL parameter is also divided into two fields. The least-significant four bits contain the YMUL field, which specifies the factor used to scale the unexpanded source bitmap in the Y dimension. YMUL can equal any value from 0–15, which represents a scaling factor of 1–16.

The YOFF field occupies the four most-significant bits and indicates the number of scanlines to be clipped at the top edge of the expanded source bitmap. YOFF ranges from 0–YMUL. If YOFF is zero, no clipping occurs at the top or bottom extent, but if it is non-zero, YOFF number of scanlines at the top edge of the expanded bitmap are skipped and the next scanline is the first one transferred to the destination bitmap.

### Related Graphic Orders

SET\_BOOL\_SD

SET\_UBMAP



**Figure 13-21. Expanded Source, Destination bitBLT To Unbanded Bitmap**

## BLT2UB\_XHD

### Expanded Source, Halftone, Destination bitBLT to Unbanded Bitmap

PARAMETERS	SIZE	DESCRIPTION
0x27	Byte	BLT2UB_XHD Opcode.
0x00	Byte	Reserved.
DA	Long Word	Destination physical bit address.
FW	Word	Destination frame width in bits.
FH	Word	Destination frame height in scanlines.
SA	Long Word	Unexpanded source physical bit address.
SW	Word	Unexpanded source frame warp in bits.
XOFF:XMUL	Two 4-bit Fields	X offset and X multiplier in bits.
YOFF:YMUL	Two 4-bit Fields	Y offset and Y multiplier in scanlines.
HXR	Word	Halftone X remainder.
HYR	Word	Halftone Y remainder.
HA	Long Word	Halftone physical bit address of the starting pixel.

The BLT2UB\_XHD graphic order transfers a low resolution source frame to a destination unbanded bitmap and applies a halftone bitmap in the process. Before being combined with the destination, the source frame is scaled to match the resolution of the destination bitmap. This results in an intermediate expanded source bitmap. The pixels of the expanded source, halftone, and destination bit maps are combined as specified by the Boolean value set in the last SET\_BOOL\_SHD order.

The SET\_UBMAP graphic order must previously define the destination bitmap warp and the SET\_HTBMAP graphic order must do the same for the halftone bitmap dimensions. During the processing of halftones, wrapping occurs at the edges of the bitmap and this results in horizontal and vertical replication (tiling) of the bitmap to cover the entire destination frame area. The DA parameter specifies the physical bit address of the area or transfer frame within the destination unbanded bitmap. The entire transfer frame must be within the bounds of the banded bitmap. DA must point to the upper left corner of the transfer frame. The warp of the destination unbanded bitmap is set by the SET\_UBMAP graphic order, which allows the bitmap to be packed or unpacked.

The FW and FH parameters define the area of the destination bitmap on which the operation is performed. FW is the frame width in bits and at a maximum equals the quantity  $W \times (XMUL + 1)$  where W is the width of the unexpanded source bitmap. FH is the frame height in scanlines and at a maximum equals the quantity  $H \times (YMUL + 1)$  where H is the height of the unexpanded source bitmap. Specifying FW and FH as defined above causes the entire expanded source image to be combined with the destination bitmap. Specifying an FW and/or FH value less than the values defined by the above equations causes only a portion of the expanded source frame to be applied to the destination. When used in combination with XOFF and YOFF, clipping can be affected at any or all extents of the expanded source bitmap.

The SA parameter defines the unexpanded source bitmap bit address. It must point to the upper left corner of the bitmap. The warp of the bitmap is set by the SW operand. This value is added to SA to locate the beginning of each successive scanline. Note that the source warp set by the SET\_SBMAP graphic order has no effect on this graphic order.

The XOFF:XMUL parameter is divided into two fields. The least-significant four bits contain the XMUL field, which specifies the factor used to scale the unexpanded source bitmap in the X dimension. XMUL must equal a specific value from 0–15, which represents a scaling factor of 1–16. Only certain scaling factors are supported, as defined in Table 13-3. Values other than those listed are ignored and no X scaling is performed.

The XOFF field occupies the four most-significant bits and indicates the number of bits to be clipped at the left edge of the expanded source bitmap. XOFF ranges from 0–XMUL. If XOFF is zero, no clipping occurs at the left extent, but if it is non-zero, XOFF number of bits in the left edge of the expanded bitmap are skipped and the next bit is the first one transferred to the destination bitmap. The YOFF:YMUL parameter is also divided into two fields. The least-significant four bits contain the YMUL field, which specifies the factor used to scale the unexpanded source bitmap in the Y dimension. YMUL can equal any value from 0–15, which represents a scaling factor of 1–16.

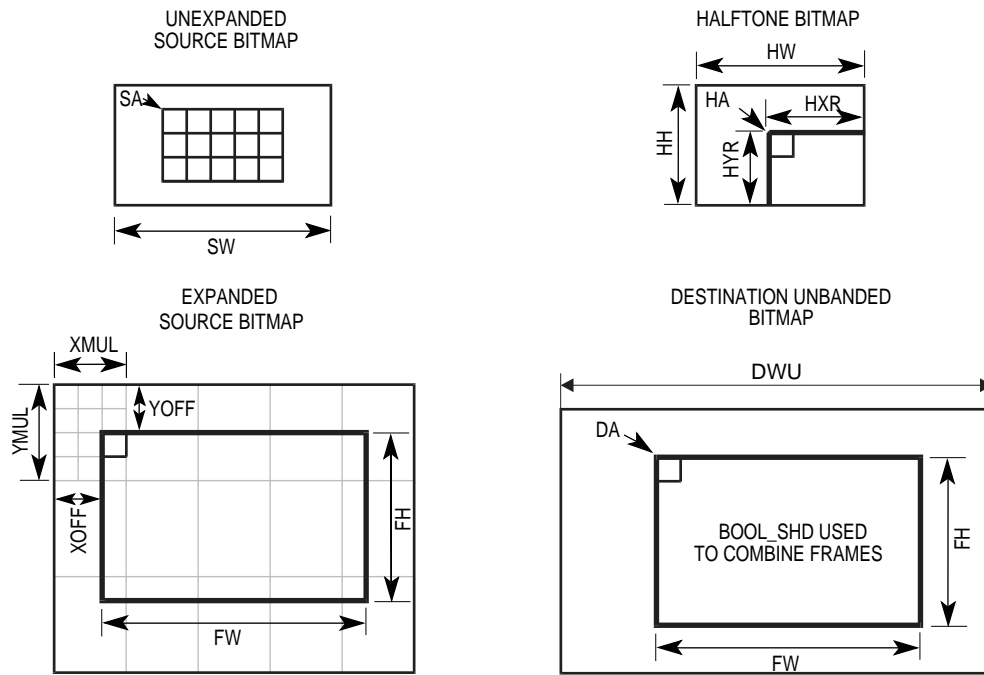
The YOFF field occupies the four most-significant bits and indicates the number of scanlines to be clipped at the top edge of the expanded source bitmap. YOFF ranges from 0–YMUL. If YOFF is zero, no clipping occurs at the top or bottom extent, but if it is non-zero, YOFF number of scanlines at the top edge of the expanded bitmap are skipped and the next scanline is the first one transferred to the destination bitmap.

Halftone tiled patterns are typically anchored to the page. Thus, a bitBLT may need to take on the halftone pattern starting at various points in the halftone bitmap, depending on where it is positioned on the page. The halftone parameters HXR, HYR, and HA define the precise halftone pixel that corresponds to the upper left corners of the source and destination frames. HXR specifies the number of pixels remaining to the right edge of the bitmap, and HYR defines the number of scanlines remaining to the bottom edge. HXR and HYR must be in the following ranges:  $1 \leq HXR \leq HW$  and  $1 \leq HYR \leq HH$ , where HW and HH are the width and height of the halftone bitmap, respectively. For example, if the starting pixel in the halftone bitmap is determined to be at the upper left, HXR must be set to HW, and HYR to HH; to start at the lower right, HXR and HYR must both be set to one.

The HA parameter defines a physical bit address within the halftone bitmap and must point to the upper left corner of the transfer frame. HA must be consistent with HXR and HYR.

### Related Graphic Orders

SET\_BOOL\_SHD  
SET\_UBMAP  
SET\_HTBMAP



**Figure 13-22. Expanded Source, Halftone, Destination bitBLT to Unbanded Bitmap**

# JUMP

## Jump to Graphic Order

PARAMETERS	SIZE	DESCRIPTION
0x01	Byte	JUMP Opcode.
0x00	Byte	Reserved.
GOA	28 of 32 bits	Graphic order physical byte address.

The JUMP graphic order indicates a new byte address where the display list continues. The MC68322 updates its internal display list pointer with the address following the graphic order and resumes execution at this address when it encounters the JUMP graphic order. GOA must be a word-aligned address, with the least-significant bit equal to zero. If the first bit is equal to one, the RGP forces it to zero.

## SET\_BBMAP

### Set Banded Bitmap Parameters

PARAMETERS	SIZE	DESCRIPTION
0x08	Byte	SET_BBMAP Opcode.
0x00	Byte	Reserved.
BAND	Byte	Current band number.
B2T	1 of 8 bits	Bottom to top duplex direction.
DWB	Word	Destination banded bitmap warp in bits.
PSUBL	Long Word	Signed difference between the physical and logical address spaces in bits.
EOBPA	Long Word	End of band physical bit address.

The SET\_BBMAP graphic order specifies the structure of a banded bitmap. The current band number, render direction, and warp are provided in addition to a signed value for mapping logical addresses to physical addresses and a physical address indicating the end of the band buffer. These parameters are used in all subsequent graphic orders that only operate on a banded bitmap. They do not, however, affect transfers to frames or unbanded bit maps.

The SET\_BBMAP graphic order specifies a current band number to be rendered from the remainder of the display list. The current band number is used in comparison against the band numbers found in subsequent bitBLT and scanline graphic orders that operate on the banded bitmap. The result of each band number comparison determines whether a graphic order is executed during the current pass of the display list. Note that a banded display list is executed several times. One pass for each band of the page.

The B2T parameter contains a 1-bit flag to indicate render direction. When the least-significant bit of the B2T byte is set, all subsequent bitBLT and scanline transfers to banded bit maps are rendered in bottom-to-top order. In other words, bitBLTs begin at the bottom-most scanline of each frame and proceed towards the top of the frame. Scanline line tables are read starting with the last bit string specifier and executed in reverse order. When the B2T flag is set, the MC68322 expects all starting pixel addresses and table addresses to point to the bottom or end of their respective operands instead of the top. Graphic orders that operate on frames or unbanded bit maps are unaffected by the B2T flag and are always rendered in a forward direction. By rendering in a reverse direction, the bands of a 180° page can be created in opposite order and transmitted to the print engine to print the back side of duplex pages. The DWB parameter specifies the destination warp of the banded bitmap. Note that if the banded bitmap is destined to be printed by the print engine video controller, the bitmap must adhere to page image requirements. In part, the warp of the bitmap must equal to 0 mod 16 and the bitmap must not contain any pad words.

The value of PSUBL is used to translate the logical destination addresses found in subsequent banded graphic orders into physical memory addresses. Once mapped to physical memory space, the graphic transfer can begin. The value for PSUBL can be computed from the physical address of the current band buffer and the logical address of the top-most pixel in the band:

$$\text{PSUBL} = \text{buffer address}_{\text{PHYSICAL}} - \text{top-most pixel address}_{\text{LOGICAL}}$$

Given PSUBL, the graphics unit can translate logical destination addresses to and from physical address space using the following calculations:

$$\text{Map logical into physical: } DA_{\text{PHYSICAL}} = DA_{\text{LOGICAL}} + \text{PSUBL}$$

$$\text{Map physical into logical: } DA_{\text{LOGICAL}} = DA_{\text{PHYSICAL}} - \text{PSUBL}$$

The EOBPA parameter is used to detect band faults. The EOBPA address must specify the first bit outside the banded bitmap buffer. As a graphic transfer proceeds, the current destination address—a physical address—is compared against EOBPA. If it is found to be greater than or equal to EOBPA (or, when the B2T flag is set, less than or equal to EOBPA), a band fault occurs. The transfer is prematurely terminated, the current destination address is translated back into logical space, and the logical address is written (along with accompanying parameters) back into the display list.

### Related Graphic Orders

BLT2BB\_D

BLT2BB\_SD

BLT2BB\_SHD

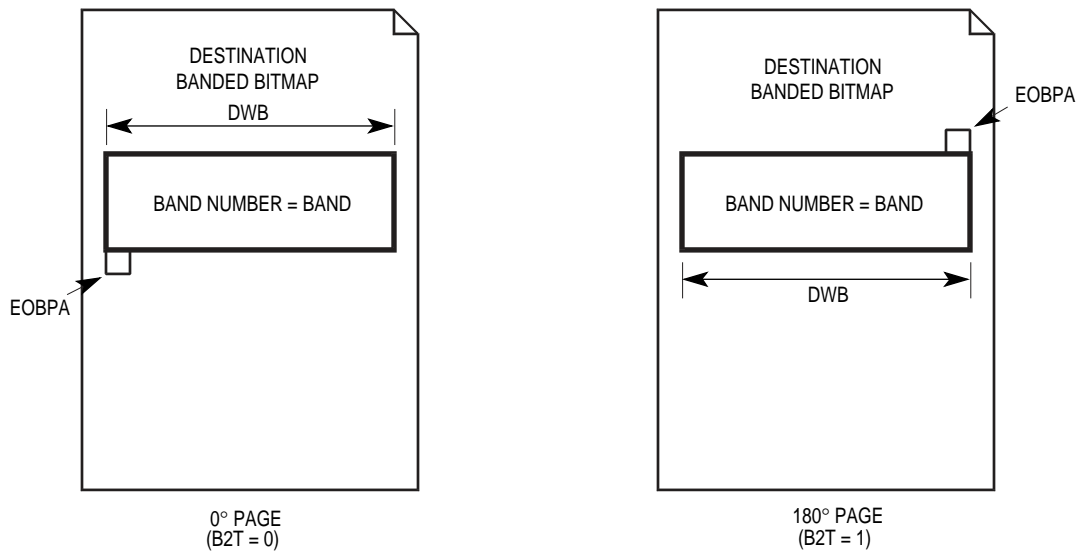
BLT2BB\_XD

BLT2BB\_XHD

SL2BB\_D

SL2BB\_HD





**Figure 13-23. Banded Bitmap Parameters**

## SET\_BOOL\_XXX

### Set Boolean Code

PARAMETERS	SIZE	DESCRIPTION
0x0C BOOL_D	Byte Byte	SET_BOOL_D Opcode. Destination-only Boolean code.
0x0D BOOL_HD	Byte Byte	SET_BOOL_HD Opcode. Halftone/Destination-only Boolean code.
0x0E BOOL_SD	Byte Byte	SET_BOOL_SD Opcode. Source/Destination-only Boolean code.
0x0F BOOL_SHD	Byte Byte	SET_BOOL_SHD Opcode. Source/Halftone/Destination Boolean code.

Note: A generalized algorithm for generating the Boolean code values is given in **Section 12 Graphic Operations**.

The SET\_BOOL\_D graphic order specifies the Boolean code to be used by all 1-operand transfer graphic orders that specify only the destination bitmap. The SET\_BOOL\_HD graphic order specifies the Boolean code to be used by all 2-operand transfer graphic orders that specify a halftone bitmap as one of their operands.

The SET\_BOOL\_SD graphic order specifies the Boolean code to be used by all 2-operand transfer graphic orders that specify a source bitmap as one of their operands. The SET\_BOOL\_SHD graphic order specifies the Boolean code to be used by all 3-operand transfer graphic orders that specify both a source and halftone bitmap as well as a destination bitmap as operands.

### Related Graphic Orders

FOR SET_BOOL_D	FOR SET_BOOL_HD	FOR SET_BOOL_SD	FOR SET_BOOL_SHD
BLT2BB_D	SL2BB_HD	BLT2BB_SD	BLT2BB_SHD
BLT2F_D	SL2F_HD	BLT2F_SD	BLT2F_SHD
BLT2UB_D	SL2UB_HD	BLT2UB_SD	BLT2UB_SHD
SL2BB_D		BLT2BB_XD	BLT2BB_XHD
SL2F_D		BLT2F_XD	BLT2F_XHD
SL2UB_D		BLT2UB_XD	BLT2UB_XHD

## SET\_HTBMAP

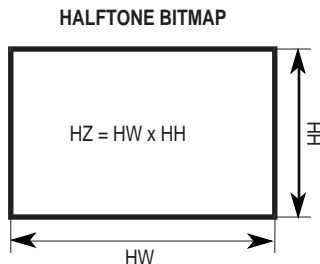
### Set Halftone Bitmap Parameters

PARAMETERS	SIZE	DESCRIPTION
0x0B	Byte	SET_HTBMAP Opcode.
0x00	Byte	Reserved.
HZ	Long Word	Halftone bitmap total size in bits.
HW	Word	Halftone bitmap width in bits.
HH	Word	Halftone bitmap height in scanlines.

The SET\_HTBMAP graphic order specifies the structure of a halftone bitmap. This graphic order specifies the width (HW), height (HH), and total size in bits (HZ) of the halftone bitmap. These parameters are used in all subsequent graphic orders that operate with halftones and they are dimensional only, meaning that no physical base address is given in the parameters. Instead, each subsequent graphic order operating with the halftone defines its own physical starting address in the bitmap, in addition to positional parameters. Both HW and HH must be in the range 1–65,535 inclusive (zero values are illegal). HZ is always equal to the product of HW and HH. The halftone bitmap must be packed. The width of the bitmap can be any value from 1–65,535 bits. A halftone bitmap narrower than 32 bits should be replicated to at least 32 bits to minimize the cost of replicating the halftone pattern across wide areas. This type of replication is not necessary in the Y dimension. The MC68322 contains specialized caching hardware to achieve the greatest performance with 32- and 64-bit wide word-aligned halftones.

### Related Graphic Orders

BLT2BB\_SHD  
 BLT2F\_SHD  
 BLT2UB\_SHD  
 BLT2BB\_XHD  
 BLT2F\_XHD  
 BLT2UB\_XHD  
 SL2BB\_HD  
 SL2F\_HD  
 SL2UB\_HD



**Figure 13-24. Halftone Bitmap Parameters**

# SET\_SBMAP

## Set Source Bitmap Parameters

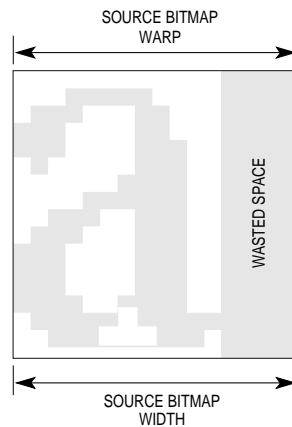
PARAMETERS	SIZE	DESCRIPTION
0x0A	Byte	SET_SBMAP Opcode.
0x00	Byte	Reserved.
SW	Word	Source bitmap warp in bits.

The SET\_SBMAP graphic order specifies the structure of a source bitmap. The warp of the bitmap is provided and, if non-zero, is used in all subsequent bitBLT graphic orders that specify a source bitmap. If the source bitmap warp is set to zero, the width of the destination frame specified by a subsequent bitBLT graphic order is assumed to be the warp of the source bitmap. This feature is particularly useful when a series of bitBLT operations are performed from a font or collection of source frames where each frame has a different width. By setting the source warp to zero, the SET\_SBMAP graphic order does not need to be included prior to each transfer.

The SW parameter is typically set to a non-zero value when it references a source which is not packed in memory. This is a common occurrence in bitmapped fonts stored in font cartridges. Each scanline of a character in a font begins on a word boundary (0 mod 2 byte address).

### Related Graphic Orders

- BLT2BB\_SD
- BLT2F\_SD
- BLT2UB\_SD
- BLT2BB\_SHD
- BLT2F\_SHD
- BLT2UB\_SHD



**Figure 13-25. Unpacked Source Bitmap**

## SET\_UBMAP

### Set Unbanded Bitmap Parameters

PARAMETERS	SIZE	DESCRIPTION
0x09	Byte	SET_UBMAP Opcode.
0x00	Byte	Reserved.
DWU	Word	Destination unbanded bitmap warp in bits.

The SET\_UBMAP graphic order specifies the structure of an unbanded bitmap. The warp of the bitmap is provided and is used in all subsequent graphic orders that operate on an unbanded bitmap. Transfers to frames and banded bit maps are not affected by the DWU value.

### Related Graphic Orders

- BLT2UB\_D
- BLT2UB\_SD
- BLT2UB\_SHD
- SL2UB\_D
- SL2UB\_HD

## SL2BB\_D

### Destination Only Scanline Transfer to Banded Bitmap

PARAMETERS	SIZE	DESCRIPTION
0x34	Byte	SL2BB_D Opcode.
BAND*	Byte	Band number when graphic order is executed.
DA*	Long Word	Destination logical bit address.
SLTA*	28 of 32 bits	Scanline table physical byte address, word aligned.

Note: \* Denotes A Parameter That The MC68322 Updates When The Frame Crosses A Band Boundary.

The SL2BB\_D graphic order causes the MC68322 to render a scanline table image to a banded bitmap. The destination is manipulated as specified by the Boolean code last set by the SET\_BOOL\_D graphic order. The SET\_BBMAP graphic order must previously define the destination banded bitmap parameters.

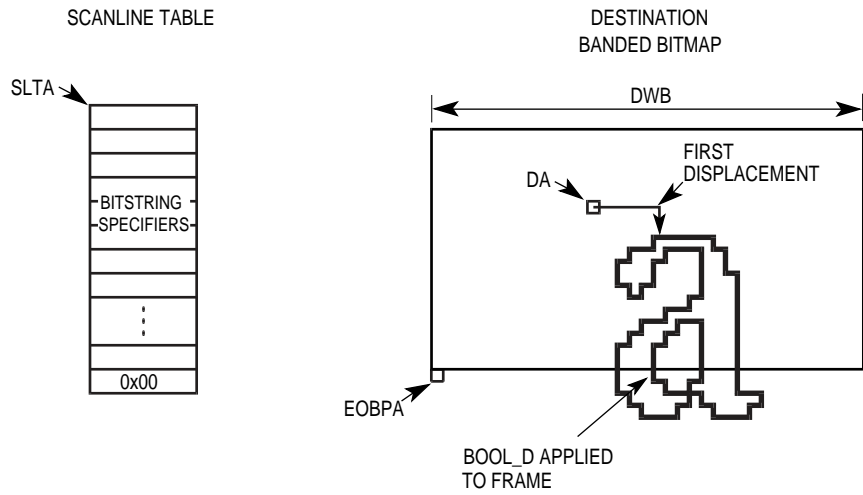
When the B2T flag is clear, SLTA points to the most-significant byte of the first bit string specifier in the table and DA refers to the pixel to which the displacement of the first bit string specifier is added (not necessarily the first bit of the first run). When the B2T flag is set, SLTA points to the most-significant byte of the last word of the final bit string specifier in the table, and DA refers to the pixel that lies just past the end of the final bit string specifier run. In neither case does SLTA point to the  $0000_{16}$  scanline table terminators. Since the scanline table's bit string specifiers must be placed at word boundaries, SLTA must be word-aligned.

When a band fault is detected, the MC68322 rewrites the scanline graphic order to update its parameters. The BAND number is incremented (or decremented when the B2T flag is set). DA is written back corresponding to the pixel following the last run rendered (or when the B2T flag is set, the pixel preceding the next bit string specifier's run) and SLTA points to the next specifier to be executed when the rest of the scanline table is rendered to the next band. The contents of the scanline table is left unchanged after a band fault.

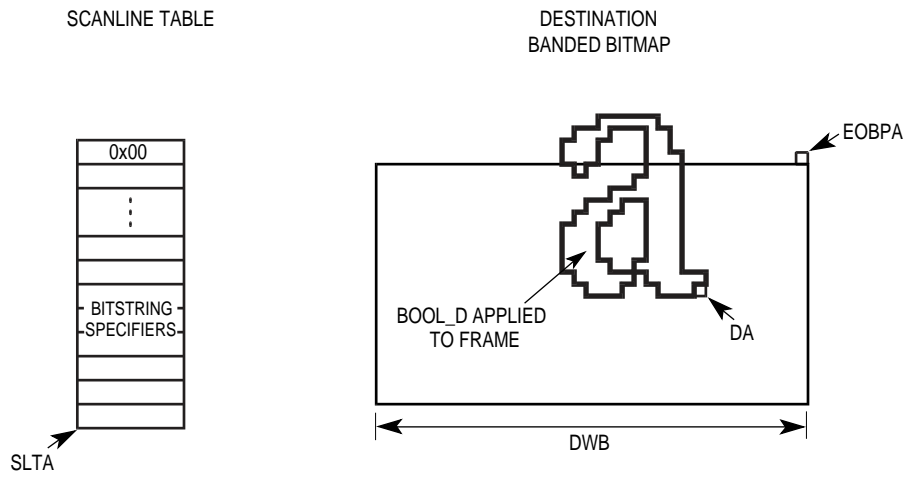
### Related Graphic Orders

SET\_BOOL\_D

SET\_BBMAP



**Figure 13-26. Destination Scanline Transfer to Banded Bitmap, 0° Page**



**Figure 13-27. Destination Scanline Transfer to Banded Bitmap, 180° Page**

## SL2BB\_HD

### Halftone/Destination Scanline Transfer to Banded Bitmap

PARAMETERS	SIZE	DESCRIPTION
0x35	Byte	SL2BB_HD Opcode.
BAND*	Byte	Band number when graphic order is executed.
DA*	Long Word	Destination logical bit address.
HXR*	Word	Halftone X remainder.
HYR*	Word	Halftone Y remainder.
HA*	Long Word	Halftone physical address of the starting pixel.
HTTA*	28 of 32 bits	Companion halftone table physical byte address, word aligned.
SLTA*	28 of 32 bits	Scanline table physical byte address, word aligned.

Note: \* Denotes A Parameter That The MC68322 Updates When The Frame Crosses A Band Boundary.

The SL2BB\_HD graphic order causes the MC68322 to render a scanline table image to a banded bitmap and apply a halftone bitmap in the process. The destination and halftone pixels are combined as specified by the Boolean code last set by the SET\_BOOL\_HD graphic order. The SET\_BBMAP graphic order must previously define the destination banded bitmap parameters and the SET\_HTBMAP graphic order must do the same for the halftone bitmap dimensions.

Halftone tiled patterns are typically anchored to the page. The rendering of a scanline table may need to take on the halftone pattern starting at various points in the halftone bitmap, depending on where it is positioned on the page. The halftone parameters HXR, HYR, and HA define the precise halftone pixel that corresponds to the initial destination address given in the graphic order. Remember that the initial destination address is not where the first pixel is drawn—when the B2T flag is clear, it is the point to which the first bit string specifier's displacement is added and when the B2T flag is set, the point immediately to the right of the last bit string specifier's run.

HXR specifies the number of pixels remaining to the right edge of the bitmap and HYR defines the number of scanlines remaining to the bottom edge (or top edge, when the B2T flag is set). HXR and HYR must be in the following ranges:  $1 \leq HXR \leq HW$  and  $1 \leq HYR \leq HH$ , where HW and HH are the width and height of the halftone bitmap, respectively. For example, when the B2T flag is clear, if the starting pixel in the halftone bitmap is determined to be at the upper left, HXR must be set to HW, and HYR to HH; if instead a 180° page is being rendered and the B2T flag is set, HXR must still be set to HW, but HYR must be set to one.



When the B2T flag is clear, SLTA and HTTA point to the most-significant byte of the first specifier in their respective tables, and DA, HXR, HYR, and HA refer to the pixel to which the displacement of the first bit string specifier is added (not necessarily the first bit of the first run). When the B2T flag is set, SLTA and HTTA point to the most-significant byte of the last word of the final specifier in their respective tables, and DA, HXR, HYR, and HA refer to the pixel just past the end of the final bit string specifier run. In neither case does SLTA point to the 0000<sub>16</sub> scanline table terminators. Since both the scanline table and companion halftone table's bit string specifiers must be placed at word boundaries, HTTA and SLTA must be word aligned.

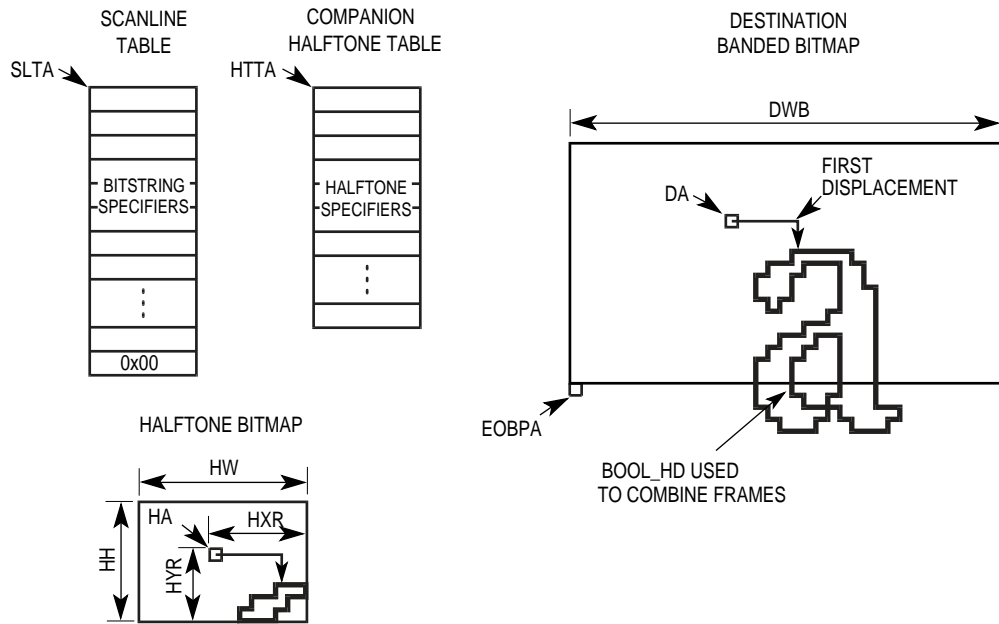
When a band fault is detected, the MC68322 rewrites the scanline graphic order to update most of its parameters. The BAND number is incremented (or decremented when the B2T flag is set). DA and HA are written back corresponding to the pixel following the last run rendered (or when the B2T flag is set, the pixel preceding the next bit string specifier's run) and HYR is written back with the number of remaining scanlines in the halftone frame to be transferred. Last, HTTA and SLTA point to the next specifier to be executed when the rest of the scanline table is rendered to the next band. The contents of the scanline and companion halftone tables are left unchanged after a band fault.

### Related Graphic Orders

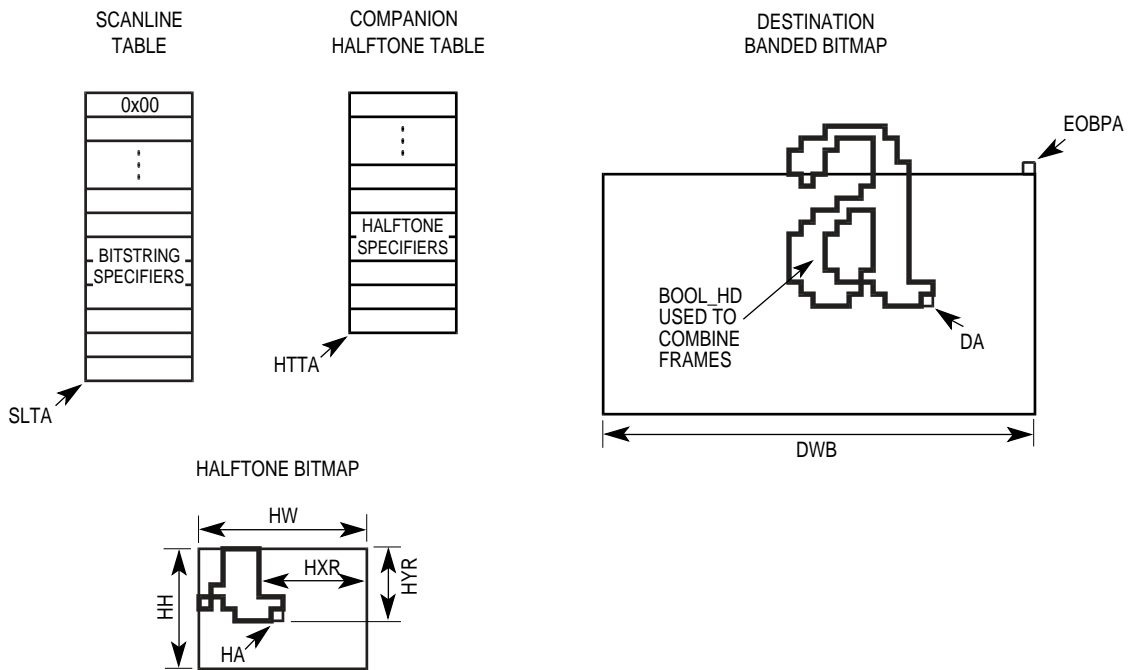
SET\_BOOL\_HD

SET\_HTBMAP

SET\_BBMAP



**Figure 13-28. Halftone, Destination Scanline Transfer to Banded Bitmap, 0° Page**



**Figure 13-29. Halftone, Destination Scanline Transfer to Banded Bitmap, 180° Page**

# SL2F\_D

## Destination Only Scanline Transfer to Frame

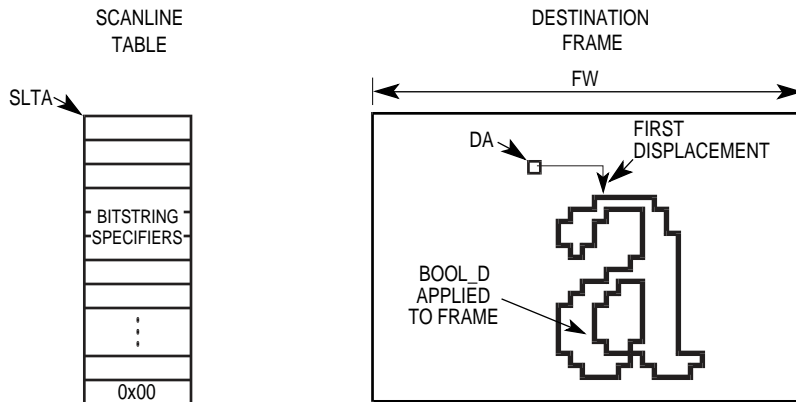
PARAMETERS	SIZE	DESCRIPTION
0x14	Byte	SL2F_D Opcode.
0x00	Byte	Reserved.
DA	Long Word	Destination physical bit address.
FW	Word	Frame width in bits.
SLTA	28 of 32 bits	Scanline table physical byte address, word aligned.

The SL2F\_D graphic order causes the MC68322 to render a scanline table image to a frame. The destination is manipulated as specified by the Boolean code last set by the SET\_BOOL\_D graphic order. The destination frame warp is taken from the graphic order's FW parameter.

SLTA points to the most-significant byte of the first bit string specifier in the table (not to the 0000<sub>16</sub> header) and DA refers to the pixel to which the displacement of the first bit string specifier is added (not necessarily the first bit of the run). Since the scanline table's bit string specifiers must be placed at word boundaries, SLTA must be word-aligned.

### Related Graphic Orders

#### SET\_BOOL\_D



**Figure 13-30. Destination Scanline Transfer to Frame**

## SL2F\_HD

### Halftone/Destination Scanline Transfer to Frame

PARAMETERS	SIZE	DESCRIPTION
0x15	Byte	SL2F_HD Opcode.
0x00	Byte	Reserved.
DA	Long Word	Destination physical bit address.
FW	Word	Frame width in bits.
HXR	Word	Halftone X remainder.
HYR	Word	Halftone Y remainder.
HA	Long Word	Halftone physical address of the starting pixel.
HTTA	28 of 32 bits	Companion halftone table physical byte address, word aligned.
SLTA	28 of 32 bits	Scanline table physical byte address, word aligned.

The SL2F\_HD graphic order causes the MC68322 to render a scanline table image to a frame and apply a halftone bitmap in the process. The destination and halftone pixels are combined as specified by the Boolean code last set by the SET\_BOOL\_HD graphic order. The destination frame warp is taken from the graphic order's frame width (FW) parameter. The SET\_HTBMAP graphic order must previously define the halftone bitmap dimensions.

Halftone tiled patterns are typically anchored to the page. The rendering of a scanline table may need to take on the halftone pattern starting at various points in the halftone bitmap, depending on where it is positioned on the page. The halftone parameters HXR, HYR, and HA define the precise halftone pixel that corresponds to the initial destination address given in the graphic order. Remember that the initial destination address is not where the first pixel is drawn—it is the point to which the first bit string specifier's displacement is added.

HXR specifies the number of pixels remaining to the right edge of the bitmap and HYR defines the number of scanlines remaining to the bottom edge. HXR and HYR must be in the following ranges:  $1 \leq HXR \leq HW$  and  $1 \leq HYR \leq HH$ , where HW and HH are the width and height of the halftone bitmap, respectively. For example, if the starting pixel in the halftone bitmap is determined to be at the upper left, HXR must be set to HW, and HYR to HH.

HTTA and SLTA point to the most-significant byte of the first specifier in their respective tables, and DA, HXR, HYR, and HA refer to the pixel to which the displacement of the first bit string specifier is added (not necessarily the first bit of the first run). Since both the scanline table and companion halftone table's specifiers must be placed at word boundaries, HTTA and SLTA must be word-aligned.

### Related Graphic Orders

SET\_BOOL\_HD

SET\_HTBMAP

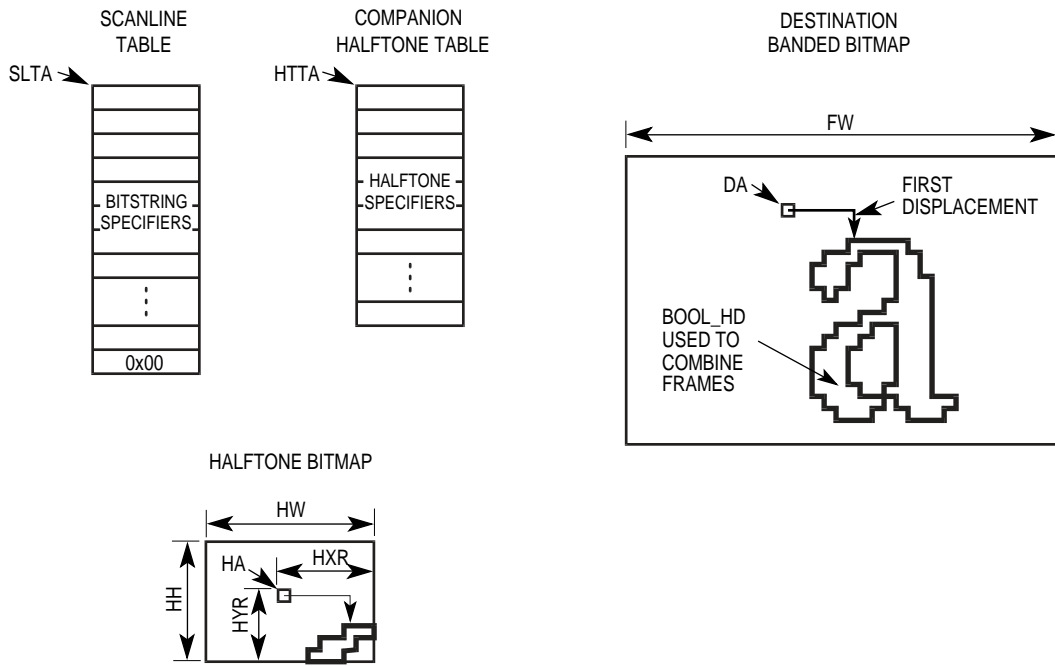


Figure 13-31. Halftone, Destination Scanline Transfer to Frame

# SL2UB\_D

## Destination Only Scanline Transfer to Unbanded Bitmap

PARAMETERS	SIZE	DESCRIPTION
0x24	Byte	SL2F_D opcode.
0x00	Byte	Reserved.
DA	Long Word	Destination physical bit address.
SLTA	28 of 32 bits	Scanline table physical byte address, word aligned.

The SL2UB\_D graphic order causes the MC68322 to render a scanline table image to an unbanded bitmap. The destination is manipulated as specified by the SET\_BOOL\_D graphic order. The SET\_UBMAP graphic order must previously define the destination unbanded bitmap warp.

SLTA points to the most-significant byte of the first bit string specifier in the table (not to the 0000<sub>16</sub> header) and DA refers to the pixel to which the displacement of the first bit string specifier is added (not necessarily the first bit of the run). Since the scanline table's bit string specifiers must be placed at word boundaries, SLTA must be word aligned.

### Related Graphic Orders

SET\_BOOL\_D

SET\_UBMAP

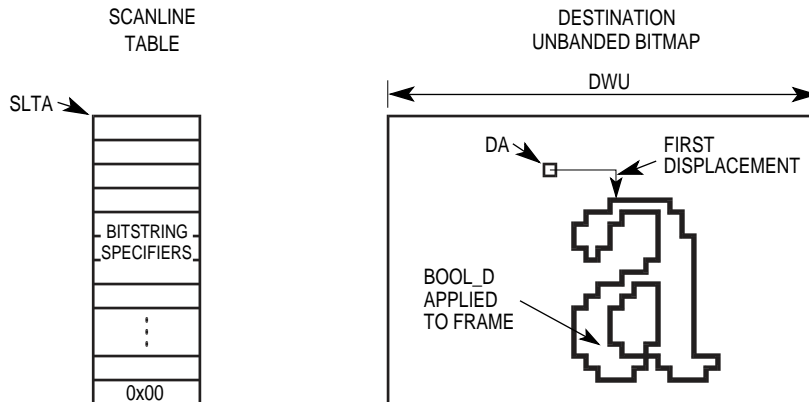


Figure 13-32. Destination Scanline Transfer to Unbanded Bitmap

## SL2UB\_HD

### Halftone/Destination Scanline Transfer to Unbanded Bitmap

PARAMETERS	SIZE	DESCRIPTION
0x25	Byte	SL2F_HD Opcode.
0x00	Byte	Reserved.
DA	Long Word	Destination physical bit address.
HXR	Word	Halftone X remainder.
HYR	Word	Halftone Y remainder.
HA	Long Word	Halftone physical address of the starting pixel.
HTTA	28 of 32 bits	Companion halftone table physical byte address, word aligned.
SLTA	28 of 32 bits	Scanline table physical byte address, word aligned.

The SL2UB\_HD graphic order causes the MC68322 to render a scanline table image to an unbanded bitmap and apply a halftone bitmap in the process. The destination and halftone pixels are combined as specified by the Boolean code last set by the SET\_BOOL\_HD graphic order. The SET\_UBMAP graphic order must previously define the destination unbanded bitmap warp and the SET\_HTBMAP graphic order must do the same for the halftone bitmap dimensions.

Halftone tiled patterns are typically anchored to the page. Thus, the rendering of a scanline table may need to take on the halftone pattern starting at various points in the halftone bitmap, depending on where it is positioned on the page. The halftone parameters HXR, HYR, and HA define the precise halftone pixel that corresponds to the initial destination address given in the graphic order. Remember that the initial destination address is not where the first pixel is drawn—it is the point to which the first bit string specifier's displacement is added.

HXR specifies the number of pixels remaining to the right edge of the bitmap and HYR defines the number of scanlines remaining to the bottom edge. HXR and HYR must be in the following ranges:  $1 \leq HXR \leq HW$  and  $1 \leq HYR \leq HH$ , where HW and HH are the width and height of the halftone bitmap, respectively. For example, if the starting pixel in the halftone bitmap is determined to be at the upper left, HXR must be set to HW, and HYR to HH.

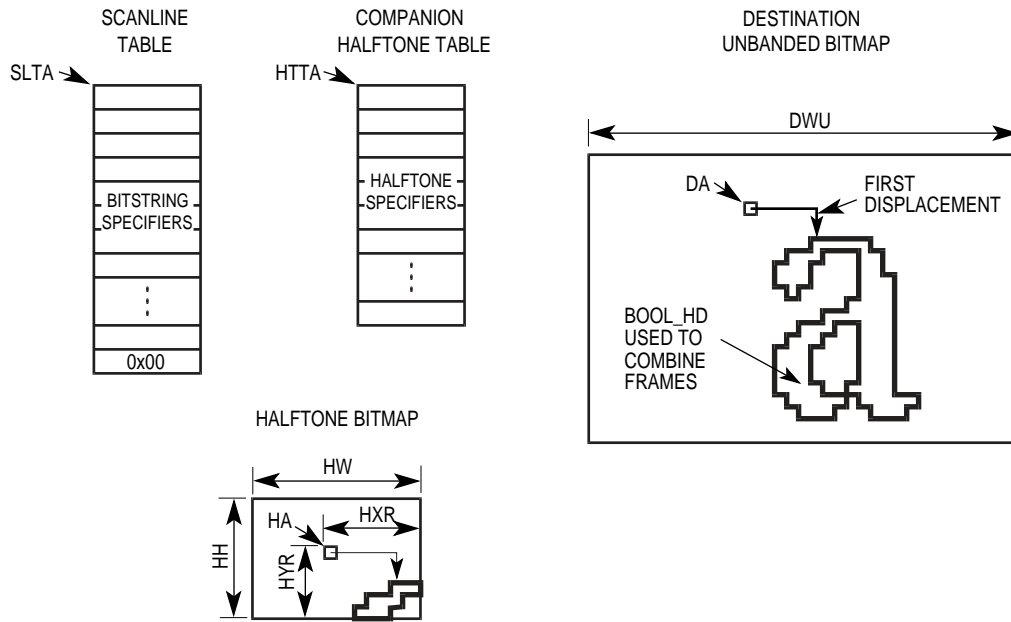
HTTA and SLTA point to the most-significant byte of the first specifier in their respective tables and DA, HXR, HYR, and HA refer to the pixel to which the displacement of the first bit string specifier is added (not necessarily the first bit of the first run). Since both the scanline table and companion halftone table's specifiers must be placed at word boundaries, HTTA and SLTA must be word-aligned.

### Related Graphic Orders

SET\_BOOL\_HD

SET\_HTBMAP

SET\_UBMAP



**Figure 13-33. Halftone, Destination Scanline Transfer to Unbande Bitmap**



# STOP

## Stop Display List Execution

PARAMETERS	SIZE	DESCRIPTION
0x00	Byte	STOP Opcode.
0x00	Byte	Reserved.

The STOP graphic order indicates the end of a display list. The MC68322 RGP halts execution and generates an RGP done interrupt event when the STOP graphic order is encountered.

## SECTION 14

# ELECTRICAL AND THERMAL CHARACTERISTICS

This section contains information on the maximum rating and thermal characteristics for the MC68322, which is subject to change. For the most recent specifications, contact a Motorola sales office.

### 14.1 MAXIMUM RATINGS

CHARACTERISTIC	SYMBOL	VALUE	UNIT
Supply Voltage <sup>1 2</sup>	$V_{CC}$	-0.3 to +7.0	V
Input Voltage <sub>1 2</sub>	$V_{in}$	-0.3 to +7.0	V
Maximum Operating Junction Temperature	$T_J$	TBD	°C
Minimum Operating Ambient Temperature	$T_A$	0 to 70 °C	°C
Storage Temperature Range	$T_{stg}$	-55 to +150	°C

This device contains protective circuitry against damage due to high static voltages or electrical fields. However, it is advised that normal precautions be taken to avoid application of any voltages higher than maximum-rated voltages to this high-impedance circuit. Reliability of operation is enhanced if unused inputs are tied to an appropriate logic voltage level (e.g., either GND or  $V_{CC}$ ).

#### NOTES:

1. Permanent damage can occur if maximum ratings are exceeded. Exposure to voltages or currents in excess of recommended values affects device reliability. Device modules may not operate normally while being exposed to electrical extremes.
2. Although sections of the device contain circuitry to protect against damage from high static voltages or electrical fields, take normal precautions to Thermal Characteristics

### 14.2 THERMAL CHARACTERISTICS

CHARACTERISTIC	SYMBOL	VALUE	RATING
Thermal Resistance, Junction to Case—PGA Package	$\theta_{JC}$	TBD	°C/W

## 14.3 DC ELECTRICAL SPECIFICATIONS

CHARACTERISTICS	SYMBOL	MIN.	MAX.	UNIT
Input High Voltage	$V_{IH}$	2.0	$V_{CC}$	V
Input Low Voltage	$V_{IL}$	GND	0.8	V
Input Leakage Current <sup>1</sup>	$I_{IN}$	—	2.5	$\mu$ A
Three-State and Open Drain Leakage Current	$I_{TSL}$	—	20	$\mu$ A
Output High Voltage ( $I_{OH}$ = Rated Maximum) $V_{CC} = 4.75$ V	$V_{OH}$	2.4	—	V
Output Low Voltage ( $I_{OL}$ = Rated Maximum)	$V_{OL}$	—	0.5	V
Current Drain, $T_A = 70^\circ\text{C}$ , $V_{CC} = 5.25\text{V}$ , $f = 16.667$ MHz <sup>2</sup>	$I_D$	—	TBD	mA
Power Dissipation 16.67 MHz = f	$P_D$	—	TBD	W
Input Capacitance All Input-Only Pins All I/O Pins	$C_{IN}$	— —	10 20	pF
Load Capacitance	$C_L$	—	80	pF
Output Drive Derating Factor <sup>3</sup> 8 mA Output Pins: ( $\overline{WE}$ , $\overline{WRU}$ , $\overline{WRL}$ , $\overline{RD}$ , D15–D0, $\overline{CS7}$ – $\overline{CS0}$ , A25–A1, $\overline{PRINT}$ , MD15–MD0, MA10–MA0, $\overline{I\_RESET}$ , $\overline{I\_DTACK}$ , $\overline{I\_CLK1}$ , R/W, DTACK, BG, AS, DACK, CBSY, CCLK, and CMD/STS) 16 mA Output Pins: ( $\overline{SELECT}$ , $\overline{RAS5}$ – $\overline{RAS0}$ , PERROR, PD7–PD0, FAULT, CAS1, CAS0, BUSY, and $\overline{ACK}$ ) 24 mA Output Pin: ( $\overline{VIDEO}$ )	$K_L$	— — —	0.09 0.05 0.04	ns/pF
ICE Bond-Out Option 4 mA Output Pins: ( $\overline{I\_DLEN}$ , $\overline{I\_IPL2}$ – $\overline{I\_IPL0}$ , $\overline{I\_HALT}$ , $\overline{I\_BR}$ , and $\overline{I\_AVEC}$ )		—	0.13	

- NOTES:
- Not including internal pull-up.
  - Currents listed are with no loading.
  - The output drive derating factor applies only to load capacitance values greater than  $C_L$ . Output drive derating factors are not accurate for load capacitance values less than  $C_L$  or for capacitance greater than 250 pF.
  - Capacitance is periodically sampled rather than always tested.

## 14.4 AC ELECTRICAL SPECIFICATIONS

### 14.4.1 Clock and Reset Timing

NUM	CHARACTERISTIC	MIN	MAX	UNIT
	Frequency of Operation <sup>1 2</sup>	4	20	MHz
1	CLK2 Period	25	—	ns
2, 3	CLK2 Pulse Width	8	—	ns
4, 5	CLK2 Rise and Fall Times	—	4	ns
6	$\overline{\text{RESET}}$ Pulse Width <sup>3</sup>	20	—	CLK2s
7	$\overline{\text{RESET}}$ Asynchronous Input Setup before CLK2 <sup>4</sup>	5	—	ns

NOTES:

1. The frequency of operation is equal to one-half the CLK2 frequency.
2. The MC68322 is a 100% static cell design, and has no absolute lower limit on operating frequency. However, system requirements, including the minimum DRAM refresh period, require special attention below 4 MHz.
3. For power-up, the MC68322 must be held in the reset state for 100 ms to allow stabilization of on-chip circuitry. After the system is powered up, this requirement refers to the minimum pulse width required to reset the processor.
4.  $\overline{\text{RESET}}$  is an asynchronous input and is synchronized internally by the MC68322. It requires no setup or hold time in order to be recognized for proper operation. However, to guarantee recognition of the input at a certain edge of CLK2,  $\overline{\text{RESET}}$  must satisfy the setup requirement.

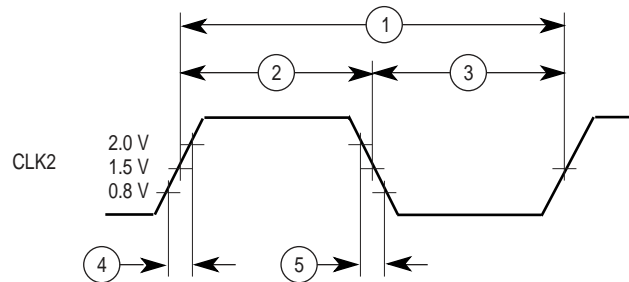


Figure 14-1. Clock AC Timing

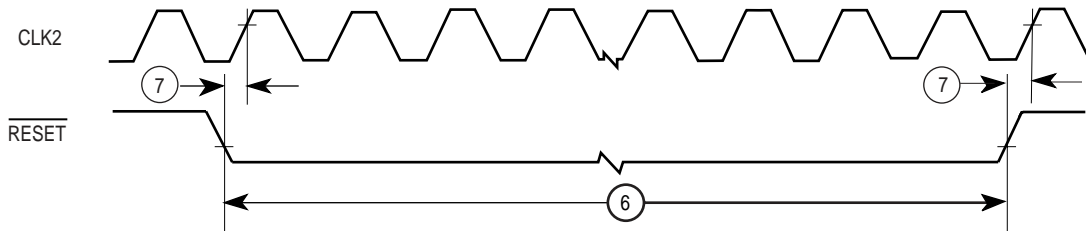


Figure 14-2.  $\overline{\text{RESET}}$  AC Timing

## 14.4.2 MC68322 Bus Timing

NUM	CHARACTERISTIC	MIN	MAX	UNIT
8	Address Bus Valid from CLK2, EC000 Cycle	3	30	ns
9	Address Bus Valid from CLK2, DMA Cycle	2	20	ns
10	Data Bus Driven and Valid from CLK2	2	20	ns
11	Data Bus High Impedance from CLK2	2	20	ns
12	Data Bus Setup before CLK2	2	—	ns
13	Data Bus Hold after CLK2	5	—	ns
14	$\overline{CS7}$ – $\overline{CS0}$ Valid from CLK2, EC000 Cycle	3	30	ns
15	$\overline{CS7}$ – $\overline{CS0}$ Valid from CLK2, DMA Cycle	2	20	ns
16	$\overline{RD}$ , $\overline{WRL}$ , $\overline{WRU}$ Valid from CLK2	2	20	ns
17	$\overline{WAIT}$ Asynchronous Input Hold after CLK2 *	5	—	ns
18	$\overline{AS}$ , $R/\overline{W}$ Valid from CLK2	3	30	ns

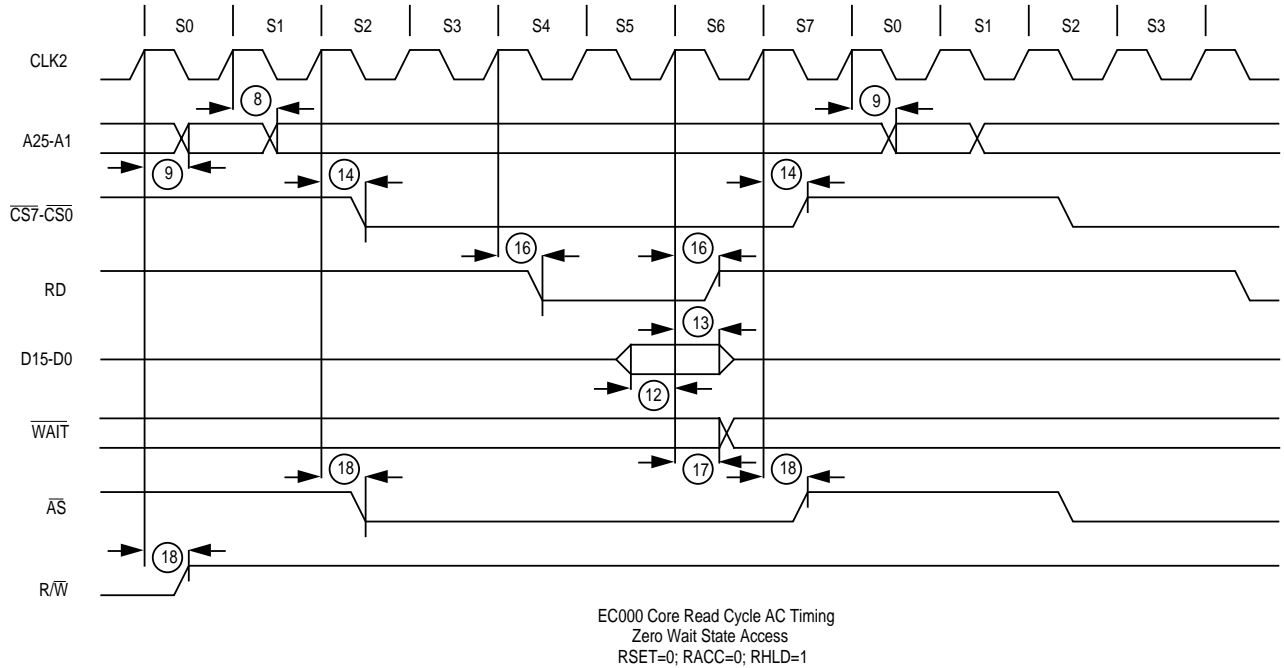
NOTE:  $\overline{WAIT}$  is an asynchronous input and is synchronized internally by the MC68322. It requires no setup or hold time in order to be recognized for proper operation. However, to guarantee recognition of the input at a certain edge of CLK2,  $\overline{WAIT}$  must satisfy the hold requirement.

The timing diagrams that follow illustrate core reads and writes. Figures 14-3 through 14-5 illustrate combinations of chip-select parameters that produce zero wait-state reads on the EC000 bus. Figures 14-6 through 14-9 illustrate combinations of chip-select parameters that produce one wait-state reads on the EC000 bus. Figure 14-11 illustrates the only combination of chip-select parameters that produce zero wait-state writes on the EC000 bus. Figures 14-12 through 14-14 illustrate combinations of chip-select parameters that produce one wait-state writes on the EC000 bus. The access times for each timing diagram are shown in parentheses in CLK2s. The numbers within the parentheses are defined as follows:

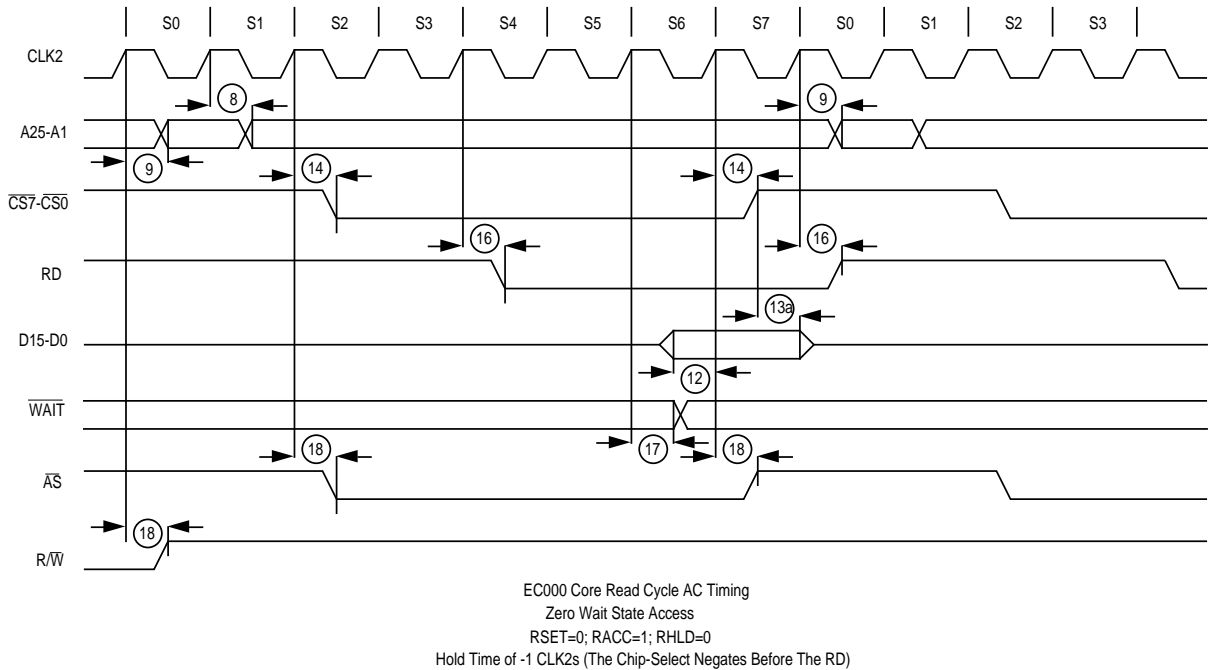
(Setup:Access:Hold:Recover)

The Setup value indicates the number of CLK2s between the assertion of the chip-select and  $\overline{RD}$ ,  $\overline{WRU}$ , or  $\overline{WRL}$ . The Access value indicates the number of CLK2s that the  $\overline{RD}$ ,  $\overline{WRU}$ , or  $\overline{WRL}$  signal will remain asserted. The Hold value indicates the number of CLK2s between the negation of  $\overline{RD}$ ,  $\overline{WRU}$ , or  $\overline{WRL}$  and chip-select. Note that some of the access times are flagged with an asterisk (\*) because the access has a hold time of -1 CLK2s. This situation occurs whenever the “hold” value in one of the chip-select registers is set to zero. In this case, the chip-select actually negates one CLK2 before the  $\overline{RD}$ ,  $\overline{WRU}$ , or  $\overline{WRL}$ . This is important because for reads in such cases, the data must be set up to the negation of the chip-select rather than the negation of the RD signal. The Recover value indicates the number of CLK2s between the negation and reassertion of the chip-select (chip-select high time.) These timing diagrams are all shown without extra recovery clocks, so the recovery time for each of the cycles will be 3 CLK2s.

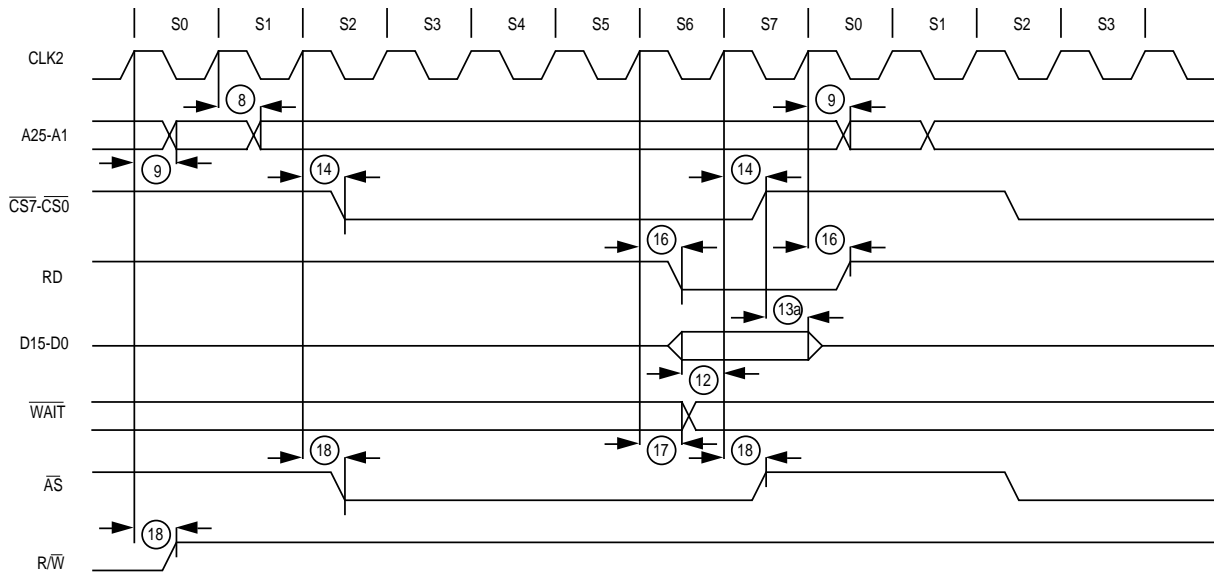
To calculate the total cycle time, add the values in the parentheses. For example, Figure 1-3 shows an access time of (2:2:1:3). The total cycle time for this setup will be 8 CLK2s or 4 CLK1s, which is a zero wait-state access for the core.



**Figure 14-3. Read Access (2:2:1:3)**

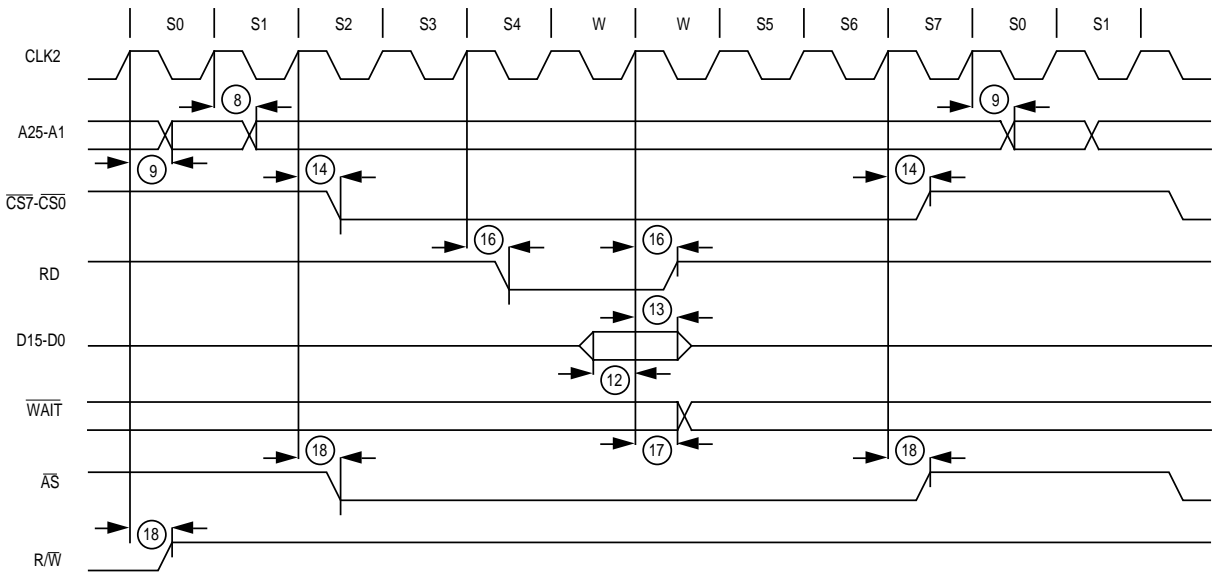


**Figure 14-4. Read Access (2:4:-1:3)\***



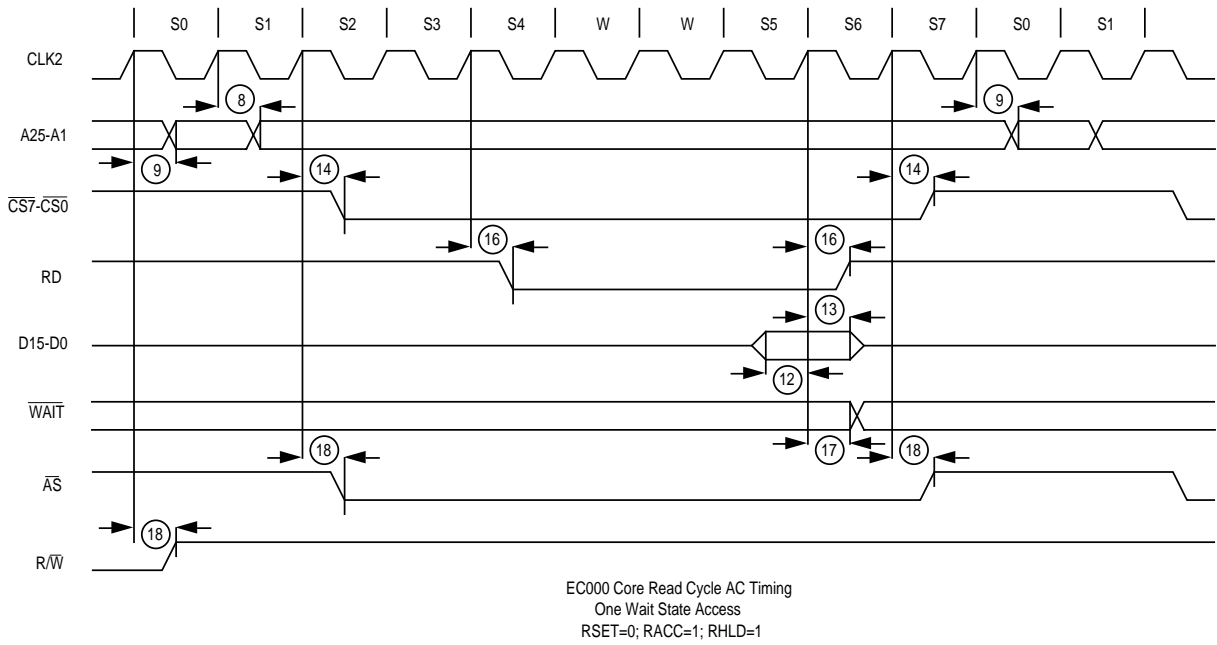
EC000 Core Read Cycle AC Timing  
 Zero Wait State Access  
 RSET=1; RACC=0; RHLD=0  
 Hold Time of -1 CLK2s (The Chip-Select Negates Before The RD)

**Figure 14-5. Read Access (4:2:-1:3)**

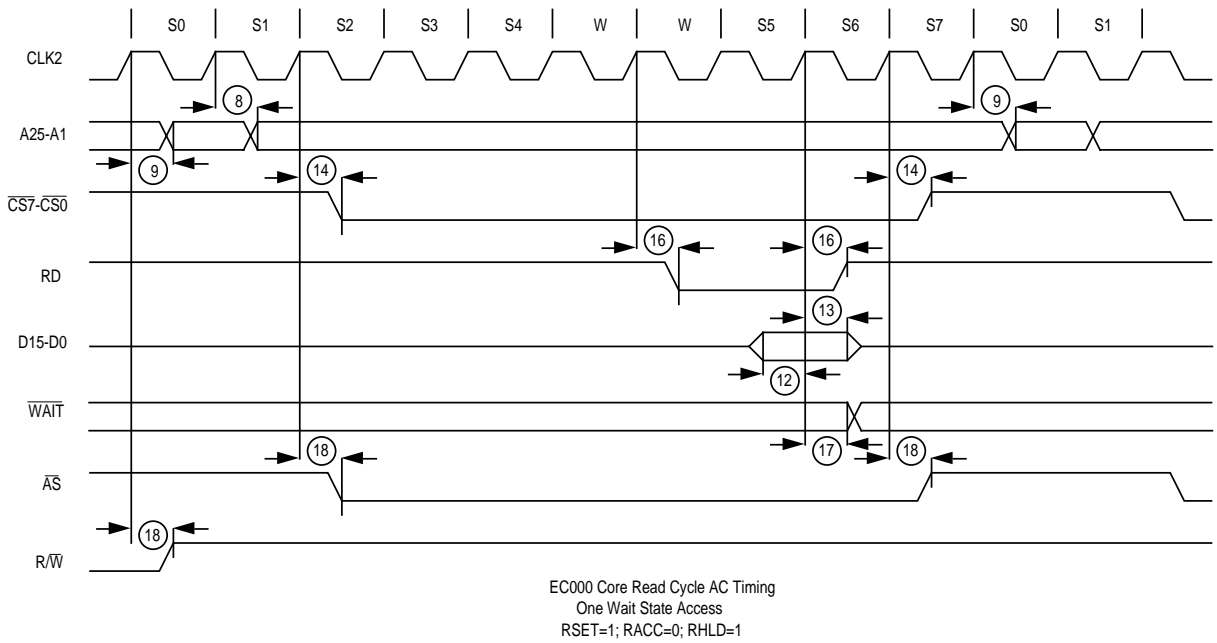


EC000 Core Read Cycle AC Timing  
 One Wait State Access  
 RSET=0; RACC=0; RHLD=2

**Figure 14-6. Read Access (2:2:3:3)**

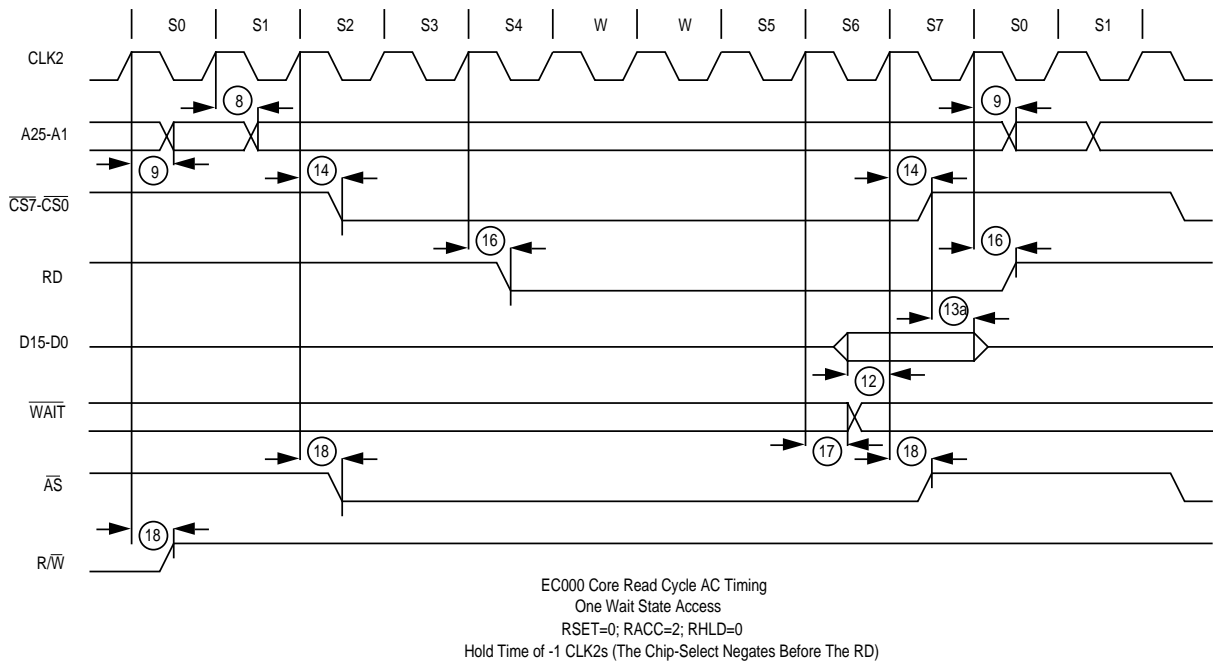


**Figure 14-7. Read Access (2:4:1:3)**

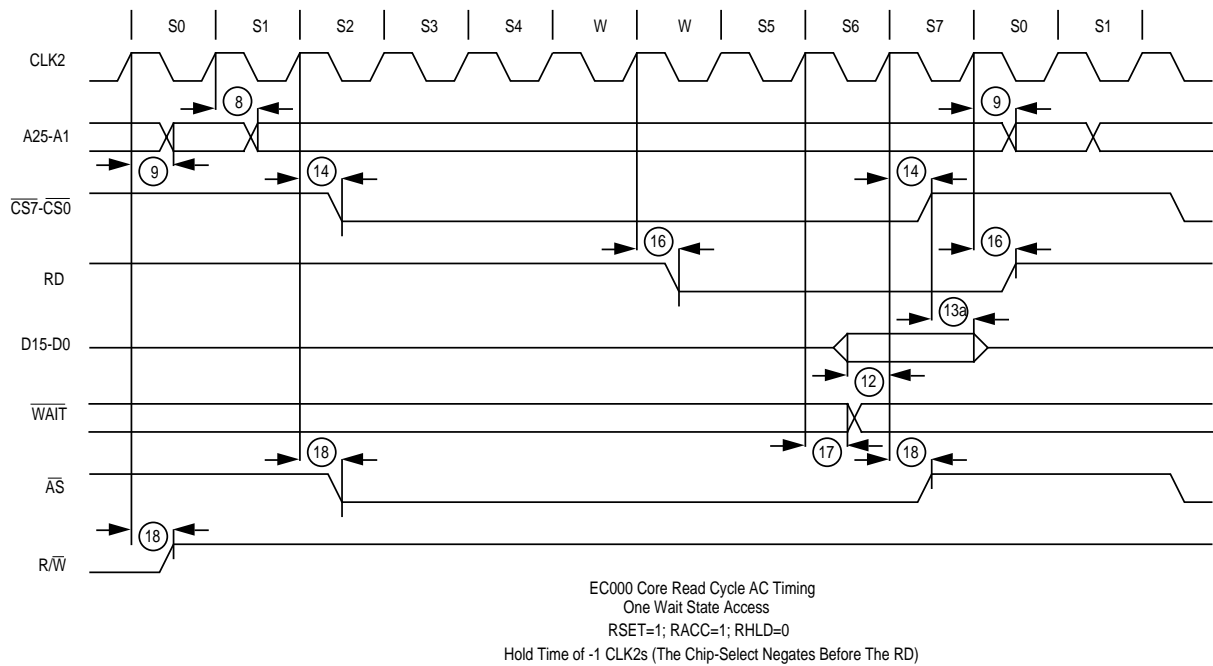


**Figure 14-8. Read Access (4:2:1:3)**

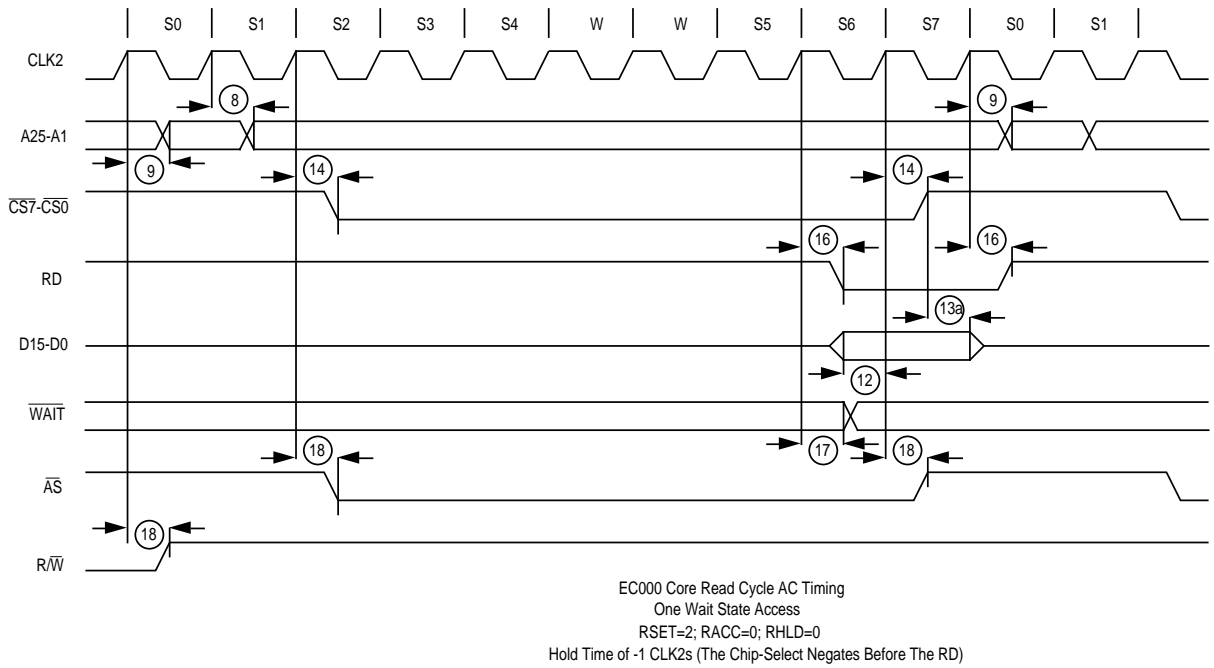




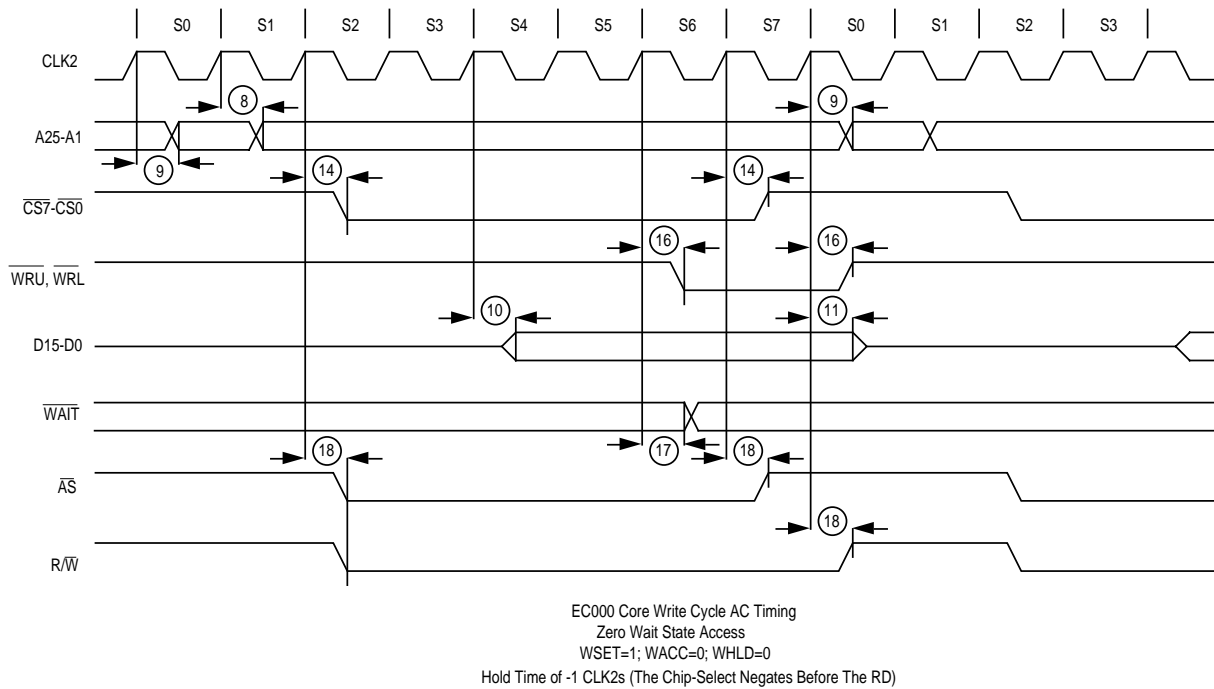
**Figure 14-9. Read Access (2:6:-1:3)**



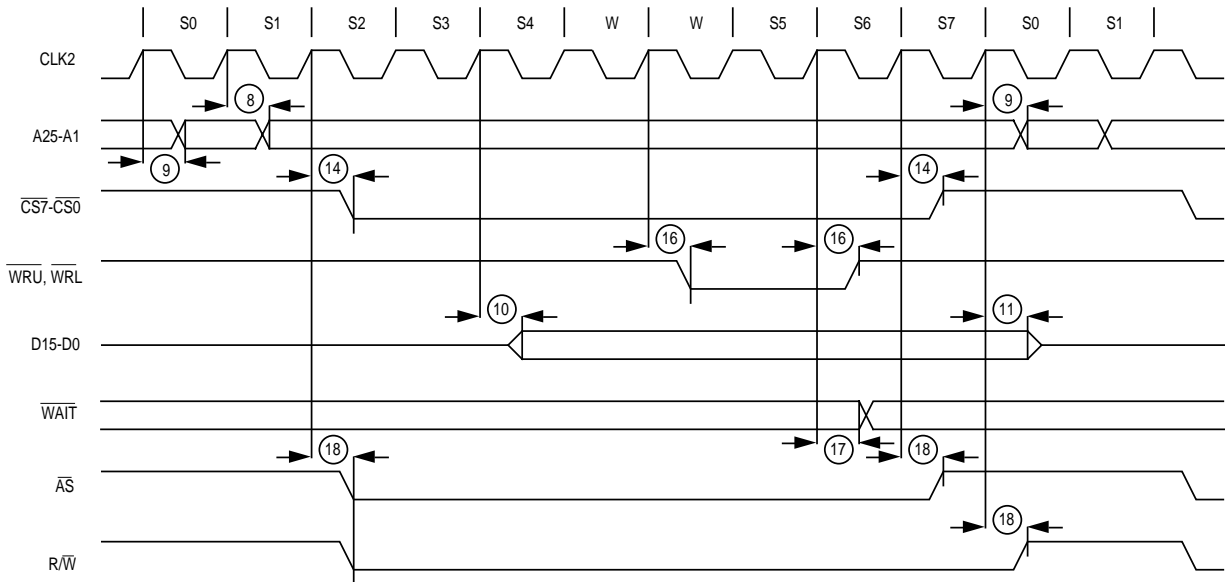
**Figure 14-10. Read Access (4:4:-1:3)**



**Figure 14-11. Read Access (6:2:-1:3)**

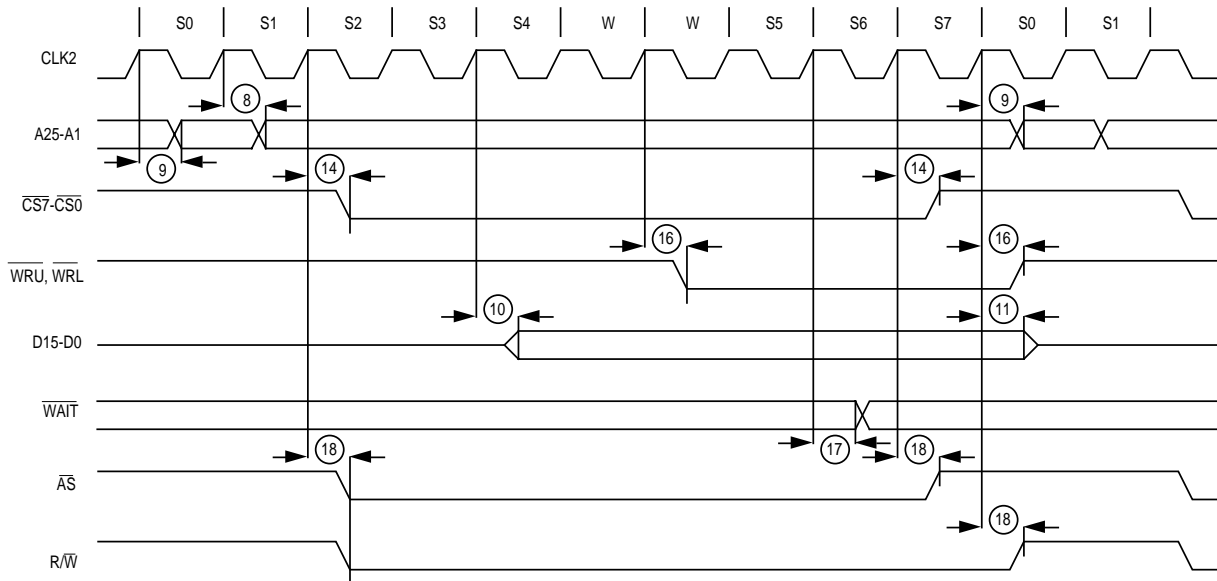


**Figure 14-12. Write Access (4:2:-1:3)**



EC000 Core Write Cycle AC Timing  
 One Wait State Access  
 WSET=1; WACC=0; WHLD=1  
 Hold Time of -1 CLK2s (The Chip-Select Negates Before The wru/wrl)

**Figure 14-13. Write Access (4:2:1:3)**



EC000 Core Write Cycle AC Timing  
 One Wait State Access  
 WSET=1; WACC=1; WHLD=0  
 Hold Time of -1 CLK2s (The Chip-Select Negates Before The WRU/WRL)

**Figure 14-14. Write Access (4:4:-1:3)**

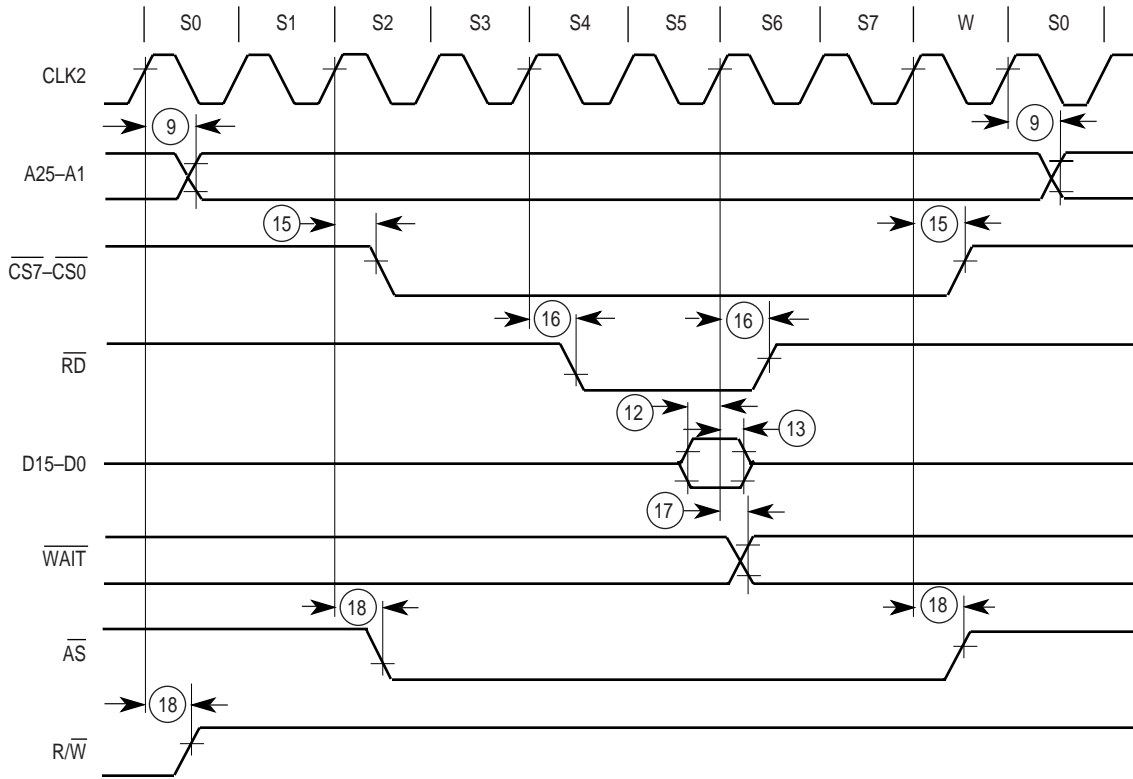


Figure 14-15. DMA Read Cycle AC Timing

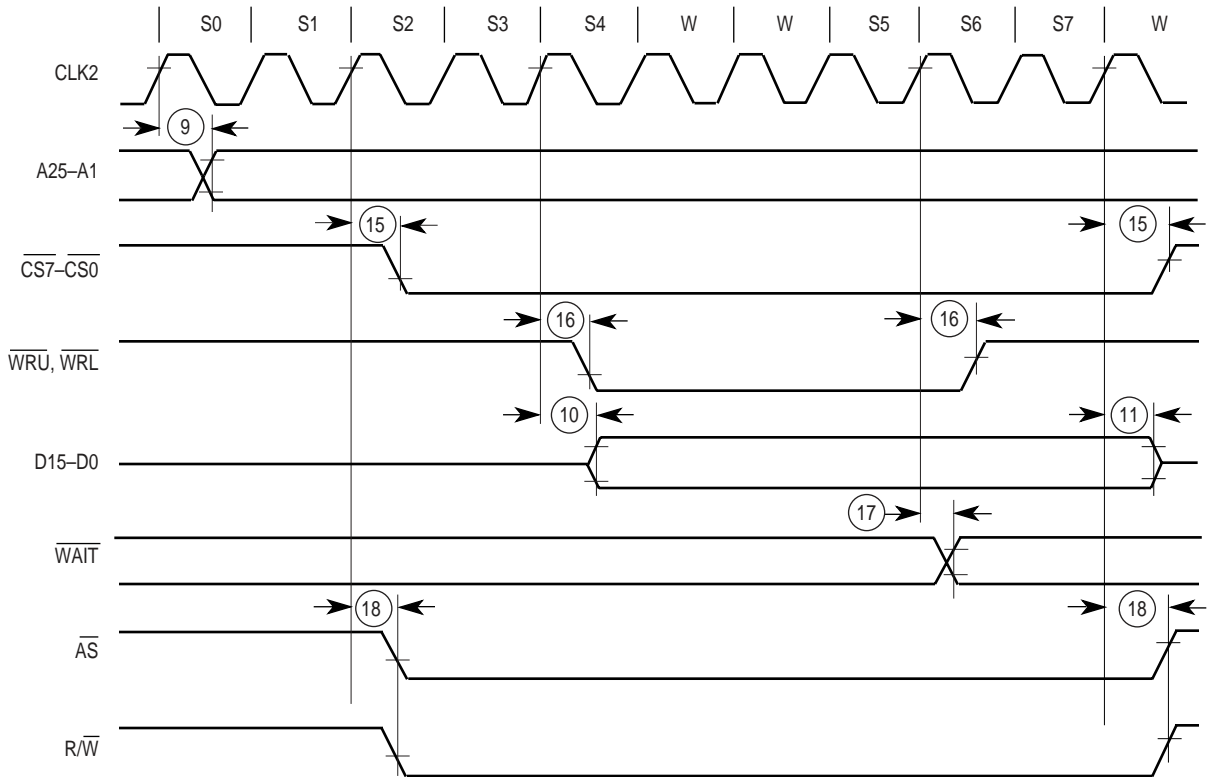


Figure 14-16. DMA Write Cycle AC Timing

### 14.4.3 DRAM Timing

NUM	CHARACTERISTIC	MIN	MAX	UNIT
19	MA10–MA0 Row Address Valid from CLK2	2	25	ns
20	MA10–MA0 Column Address Valid from CLK2	2	20	ns
21	MD15–MD0 Driven and Valid from CLK2	2	20	ns
22	MD15–MD0 High Impedance from CLK2	2	20	ns
23	MD15–MD0 Setup before CLK2	5	—	ns
23A	MD15–MD0 Hold after CLK2	5	—	ns
24	$\overline{\text{RAS}}5\text{--}\overline{\text{RAS}}0, \overline{\text{CAS}}1\text{--}\overline{\text{CAS}}0$ Valid from CLK2	2	14	ns
25	$\overline{\text{WE}}$ Valid from CLK2	2	20	ns

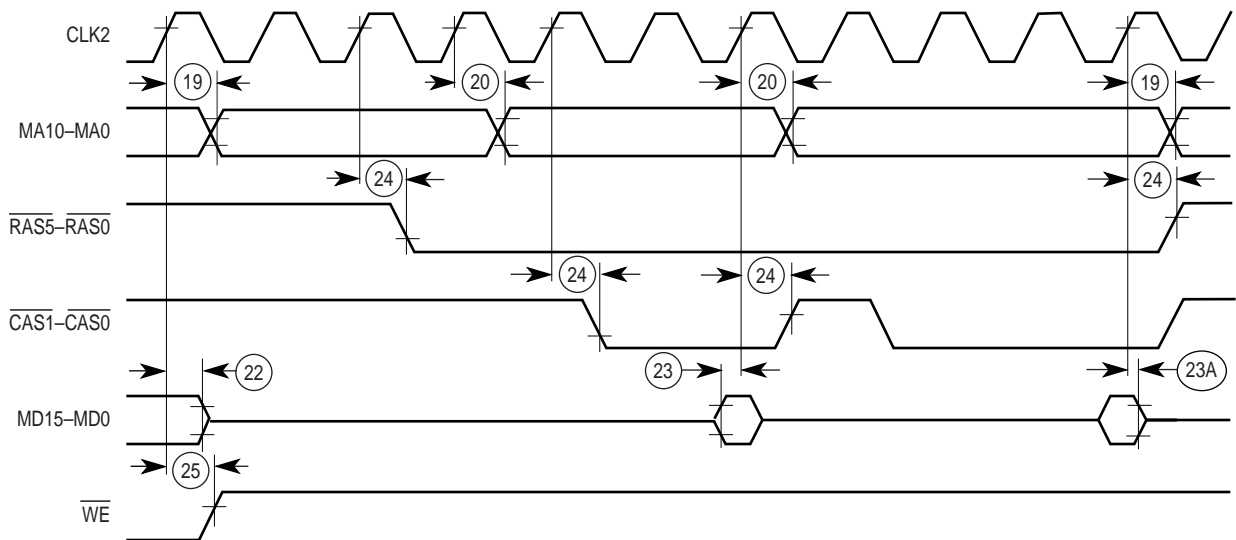


Figure 14-17. DRAM Read Cycle AC Timing

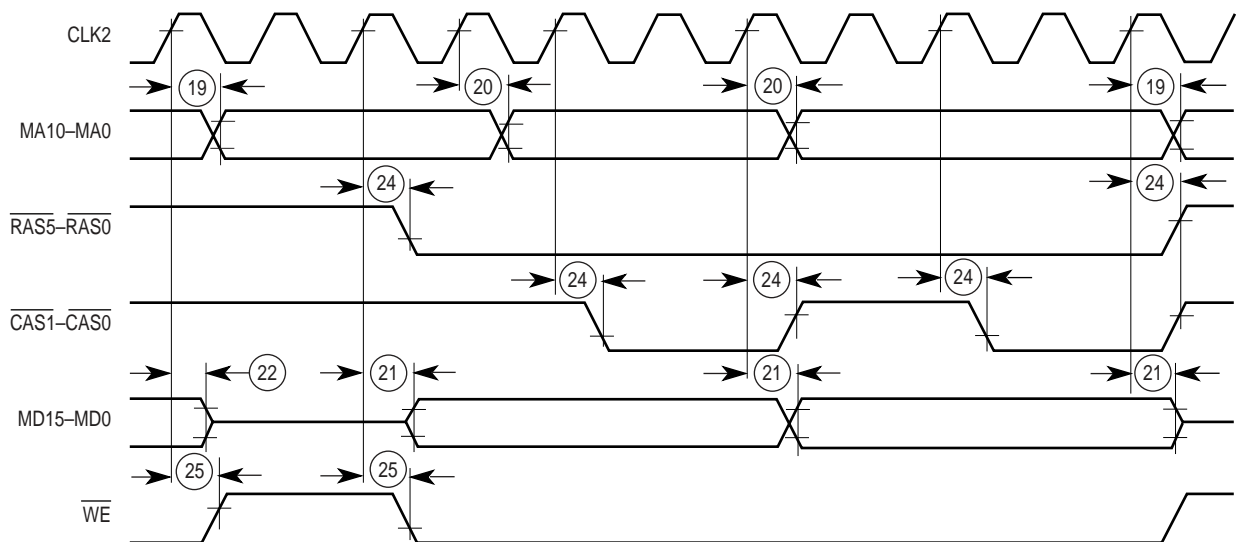
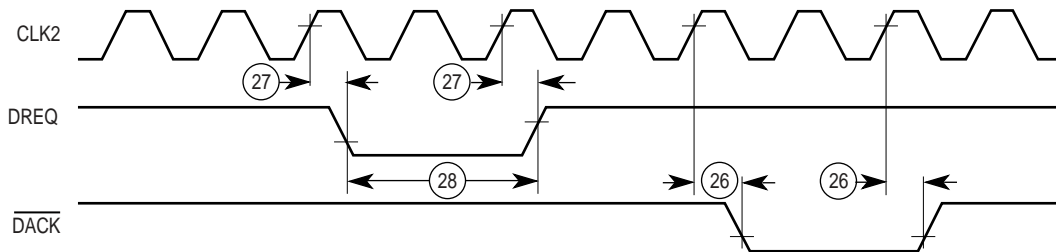


Figure 14-18. DRAM Write Cycle AC Timing

### 14.4.4 IDMA Timing

NUM	CHARACTERISTIC	MIN	MAX	UNIT
26	$\overline{\text{DACK}}$ Valid from CLK2	2	20	ns
27	DREQ Asynchronous Input Hold after CLK2 *	5	—	ns
28	DREQ Asynchronous Input Pulse Width	2	—	CLK2s

NOTE: \* Denotes that DREQ is an asynchronous input and is synchronized internally by the MC68322. It requires no setup or hold time to be recognized for proper operation. However, to guarantee recognition of the input at a certain edge of CLK2, DREQ must satisfy the hold requirement.



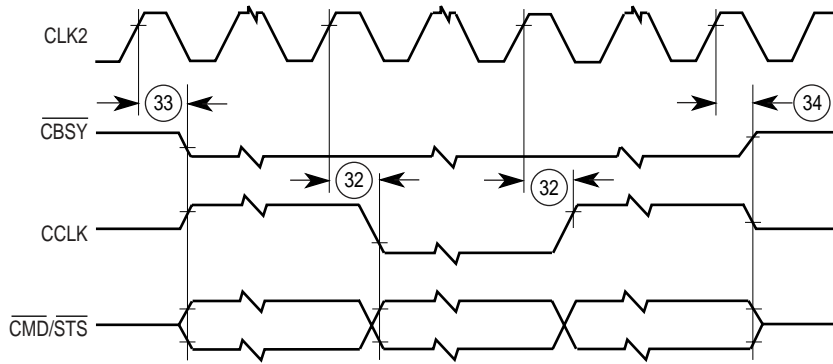
**Figure 14-19. DMA Request/Acknowledge AC Timing**

## 14.4.5 Print Engine Interface Timing

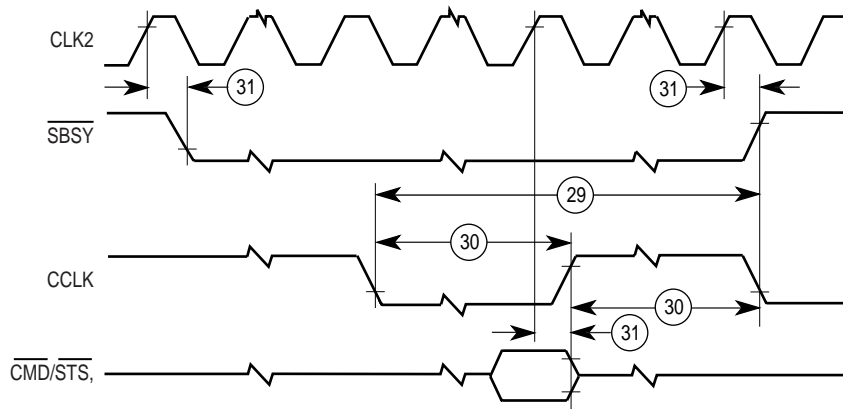
NUM	CHARACTERISTIC	MIN	MAX	UNIT
29	CCLK Period <sup>1</sup>	16	—	CLK2s
30	CCLK Pulse Width <sup>1</sup>	8	—	CLK2s
31	CCLK, $\overline{\text{CMD/STS}}$ , $\overline{\text{SBSY}}$ , $\overline{\text{STS}}$ Asynchronous Input Hold after CLK2 <sup>2</sup>	5	—	ns
32	CCLK, $\overline{\text{CBSY}}$ , $\overline{\text{CMD/STS}}$ Valid from CLK2	2	20	ns
33	CCLK, $\overline{\text{CBSY}}$ , $\overline{\text{CMD/STS}}$ Driven from CLK2	2	20	ns
34	CCLK, $\overline{\text{CBSY}}$ , $\overline{\text{CMD/STS}}$ High Impedance from CLK2	2	20	ns
	Frequency of Operation 1× Mode PLL Mode	— —	25 80	MHz
35	VCLK Period 1× Mode PLL Mode	40 12.5	— —	ns
36,37	VCLK Pulse Width 1× Mode PLL Mode	8 4	— —	ns
38, 39	VCLK Rise and Fall Times 1× Mode PLL Mode	— —	8 8	ns
40	$\overline{\text{FSYNC}}$ , $\overline{\text{LSYNC}}$ Asynchronous Input Hold after VCLK <sup>3 4</sup> 1× Mode ( $\overline{\text{FSYNC}}$ only) PLL Mode	5 5	— —	ns
41	$\overline{\text{LSYNC}}$ Setup before VCLK <sup>5 6</sup>	5	—	ns
42	$\overline{\text{LSYNC}}$ Hold after VCLK <sup>5 6</sup>	5	—	ns
43	$\overline{\text{FSYNC}}$ , $\overline{\text{LSYNC}}$ Pulse Width <sup>6</sup> 1× Mode PLL Mode	2 2	— —	dots
44	$\overline{\text{PRINT}}$ Valid from CLK2	2	20	ns

## NOTES:

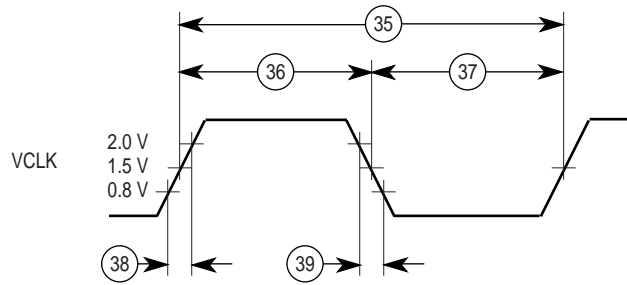
1. Applies only when CCLK is configured as an input.
2. CCLK and  $\overline{\text{CMD/STS}}$  when configured as inputs, and  $\overline{\text{SBSY}}$  and  $\overline{\text{STS}}$ , are asynchronous inputs and are synchronized internally by the MC68322. They require no setup or hold time in order to be recognized for proper operation. However, to guarantee recognition of an input at a certain edge of CLK2, the input must satisfy the hold requirement.
3.  $\overline{\text{FSYNC}}$  (in 1× or PLL mode) and  $\overline{\text{LSYNC}}$  (in PLL mode only) are asynchronous inputs and are synchronized internally by the MC68322. They require no setup or hold time to be recognized for proper operation. However, to guarantee recognition of an input at a certain edge of CLK2, the input must satisfy the hold requirement.
4. The specification is relative to the edge of VCLK selected by the VCP bit in the PVCCR.
5.  $\overline{\text{LSYNC}}$  is a synchronous input when the PVC operates in 1× mode.
6. The minimum pulse widths for  $\overline{\text{FSYNC}}$  and  $\overline{\text{LSYNC}}$  depend on the video dot rate, and is specified in video dot periods (dots). In 1× mode, the video dot period is equal to the VCLK period. In PLL mode, the video dot period is determined by the VCLK period and the configuration of the PLL.



**Figure 14-20. Print Engine Interface Input AC Timing**



**Figure 14-21. Print Engine Interface Output AC Timing**



**Figure 14-22. Video Clock AC Timing**



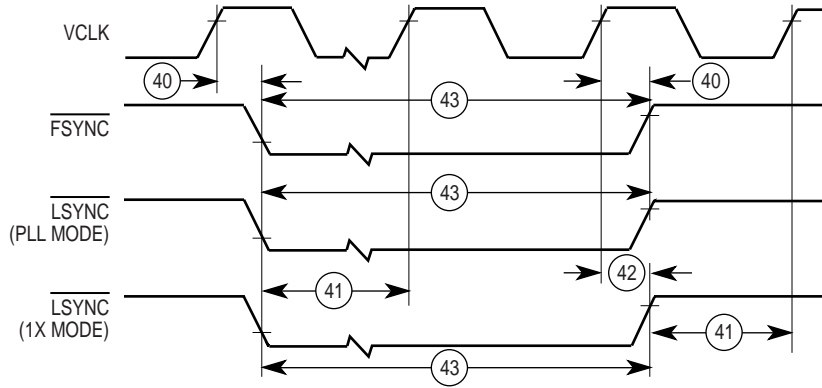


Figure 14-23. PVC AC Timing

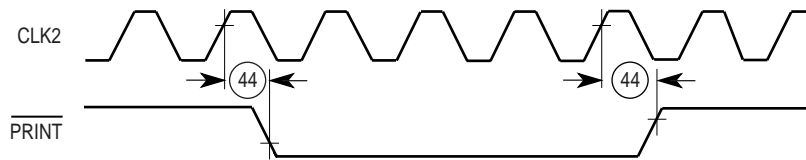


Figure 14-24. Print Engine Interface AC Timing

### 14.4.6 Interrupt Timing

NUM	CHARACTERISTIC	MIN	MAX	UNIT
45	IRQ1–IRQ0 Asynchronous Input Hold after CLK2	5	—	ns
46	IRQ1–IRQ0 Pulse Width	2	—	CLK2s

NOTE IRQ1–IRQ0 are asynchronous inputs and are synchronized internally by the MC68322. They require no setup or hold time to be recognized for proper operation. However, to guarantee recognition of an input at a certain edge of CLK2, the input must satisfy the hold requirement.

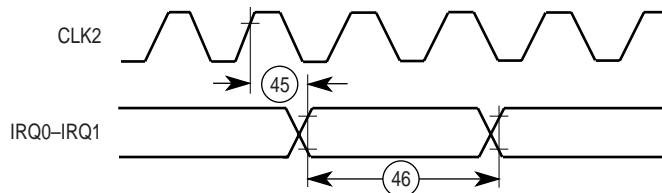


Figure 14-25. Interrupt Interface AC Timing

### 14.4.7 Parallel Port Interface Timing

NUM	CHARACTERISTIC	MIN	MAX	UNIT
47	PD7–PD0, $\overline{\text{ACK}}$ , $\text{BUSY}$ , $\text{SELECT}$ , $\text{PERROR}$ , $\overline{\text{FAULT}}$ Output Valid from CLK2	2	20	ns
48	PD7–PD0 Driven from CLK2	2	20	ns
49	PD7–PD0 High Impedance from CLK2	2	20	ns
50	PD7–PD0, $\overline{\text{SELECTIN}}$ , $\overline{\text{STROBE}}$ , $\overline{\text{AUTOFD}}$ , $\overline{\text{INIT}}$ Asynchronous Input Hold after CLK2 *	5	—	ns

NOTE: \* Denotes that PD7–PD0,  $\overline{\text{SELECTIN}}$ ,  $\overline{\text{STROBE}}$ ,  $\overline{\text{AUTOFD}}$ ,  $\overline{\text{INIT}}$  are asynchronous inputs and are synchronized internally by the MC68322. They require no setup or hold time to be recognized for proper operation. However, to guarantee recognition of an input at a certain edge of CLK2, the input must satisfy the hold requirement.

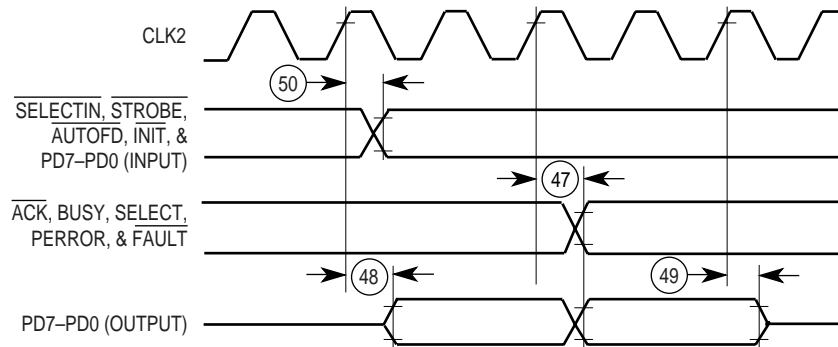


Figure 14-26. Parallel Port Interface AC Timing

### 14.4.8 External Bus Master Timing

NUM	CHARACTERISTIC	MIN	MAX	UNIT
51	$\overline{BG}$ Valid from CLK2	2	20	ns
52	Address Bus, $\overline{AS}$ , $R/\overline{W}$ Driven from CLK2	2	20	ns
53	Address Bus, $\overline{AS}$ , $R/\overline{W}$ High Impedance from CLK2	2	20	ns
54	$\overline{DTACK}$ Low from CLK2	2	20	ns
55	$\overline{DTACK}$ High from $\overline{AS}$ High	0	10	ns
56	$\overline{AS}$ , $\overline{BR}$ Asynchronous Input Hold after CLK2 *	5	—	ns
57	Address Bus, $R/\overline{W}$ Setup before $\overline{AS}$ Low	0	—	ns
58	Address Bus, $\overline{AS}$ , $R/\overline{W}$ Hold after $\overline{DTACK}$ Low	0	—	ns
59	Data Bus Valid from CLK2 (Read Cycle)	2	20	ns
60	Data Bus High Impedance after $\overline{DTACK}$ High (Read Cycle)	0	—	ns
61	Data Bus Valid after $\overline{AS}$ Low (Write Cycle)	—	4	CLK2s
62	Data Bus Hold after $\overline{DTACK}$ Low (Write Cycle)	0	—	ns
63	$\overline{BR}$ High after $\overline{AS}$ Low (Hold Time)	0	—	ns
64	$\overline{AS}$ Width High	2	—	CLK2s

NOTE: \* Denotes that  $\overline{AS}$  and  $\overline{BR}$  are asynchronous inputs and are synchronized internally by the MC68322. They require no setup or hold time to be recognized for proper operation. However, to guarantee recognition of an input at a certain edge of CLK2, the input must satisfy the hold requirement.

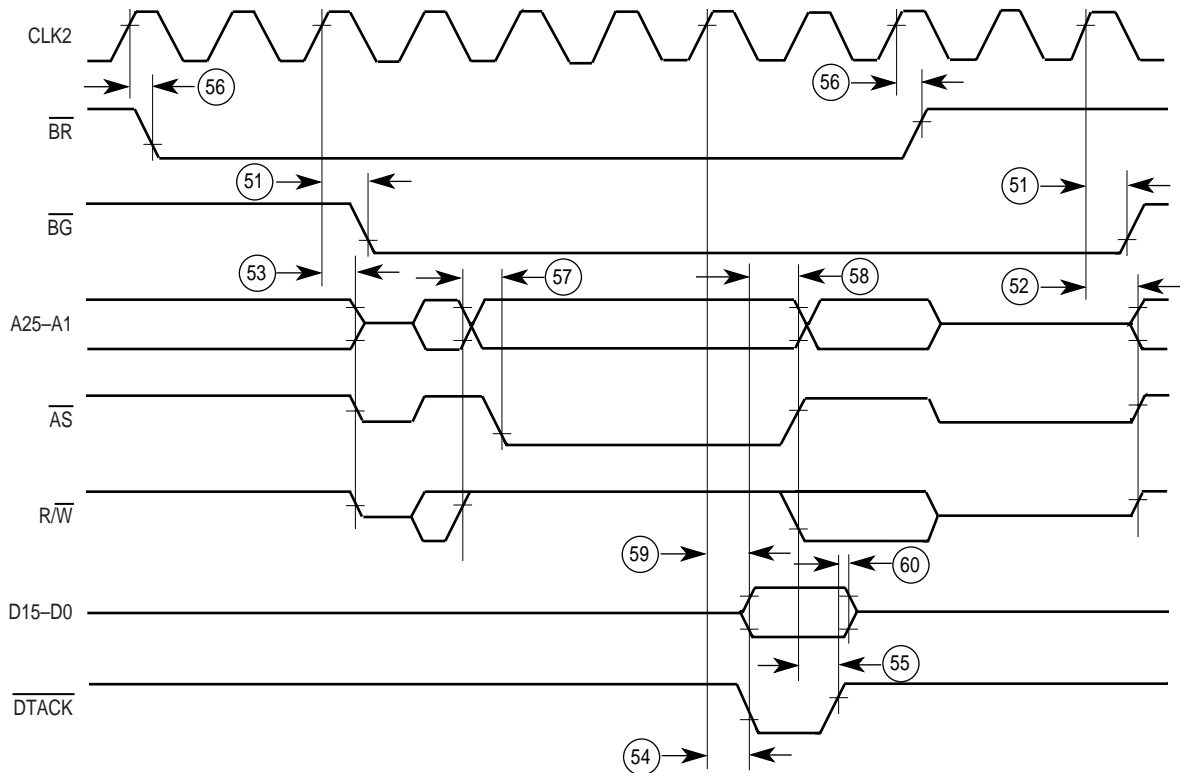
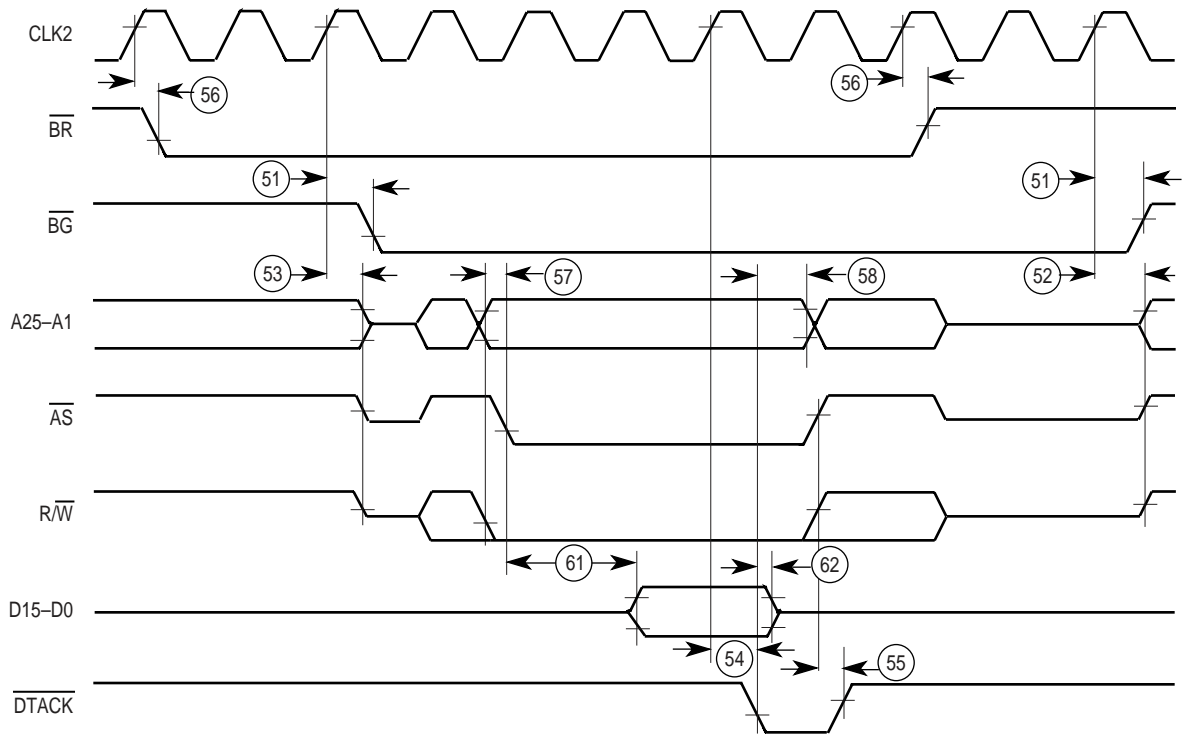
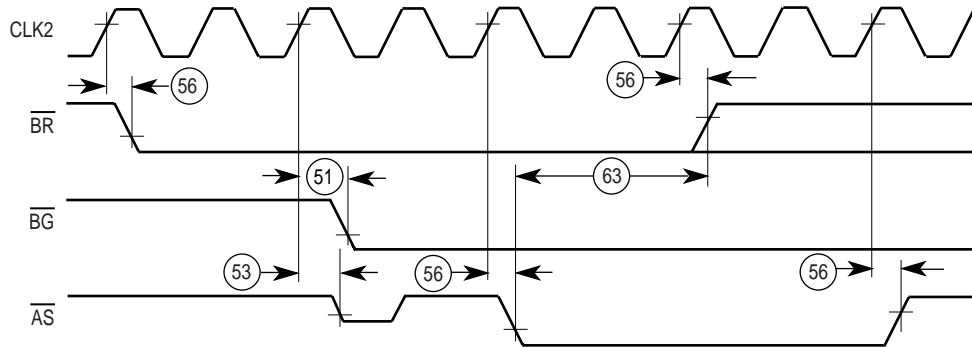


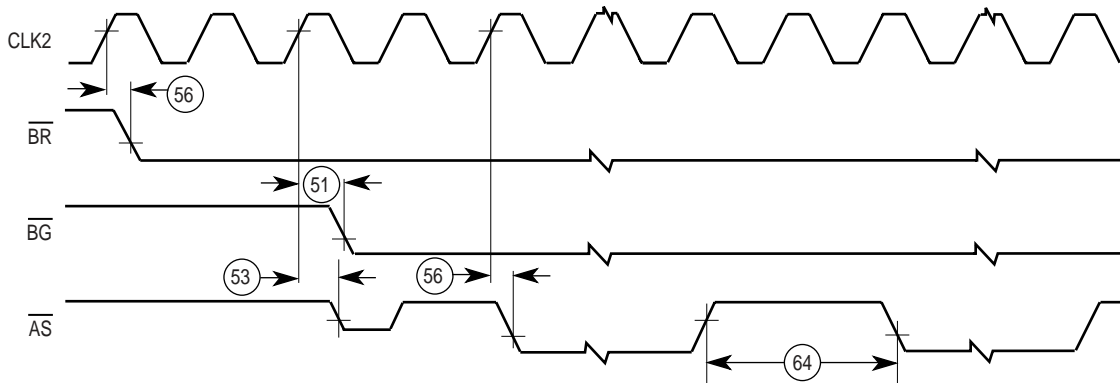
Figure 14-27. External Bus Master Read Cycle AC Timing



**Figure 14-28. External Bus Master Write Cycle AC Timing**



**Figure 14-29. External Bus Master Bus Arbitration AC Timing**



**Figure 14-30. External Bus Master Multiple Cycle AC Timing**

## SECTION 15

# ORDERING INFORMATION AND MECHANICAL DATA

This section contains MC68322 ordering information, pin assignments, and package dimensions.

### 15.1 ORDERING INFORMATION

The following table provides ordering information pertaining to the MC68322 package types, frequencies, temperatures, and Motorola order numbers.

PACKAGE TYPE	FREQUENCY	TEMPERATURE	ORDER NUMBER
160 Pin QFP FT Suffix	25 MHz 20 MHz 16.667 MHz	0 to 70° C	MC68322FT25 MC68322FT20 MC68322FT16

### 15.2 PIN ASSIGNMENT

The following illustrates the MC68322 pin assignments for the 160 PQFP package.

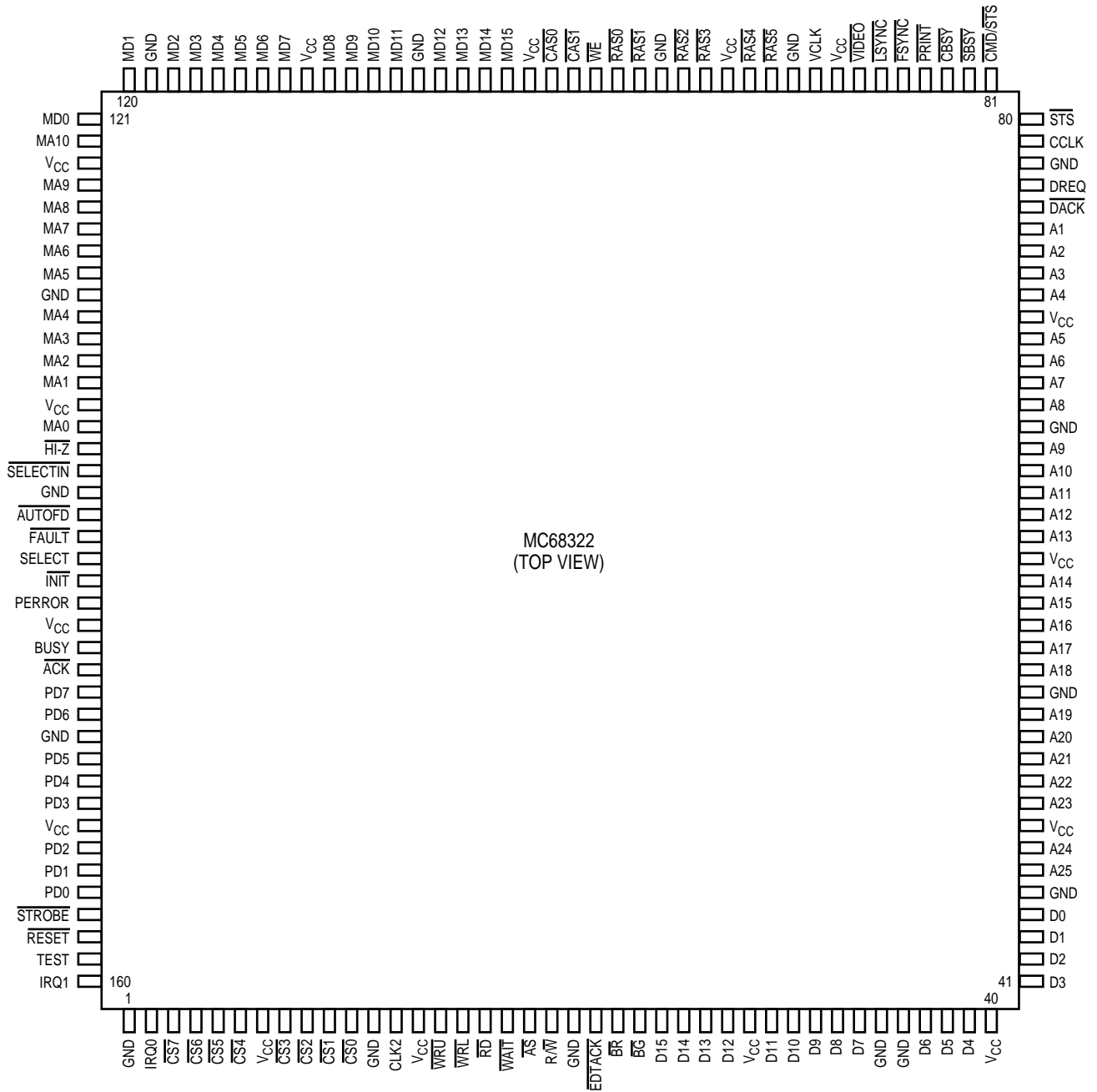
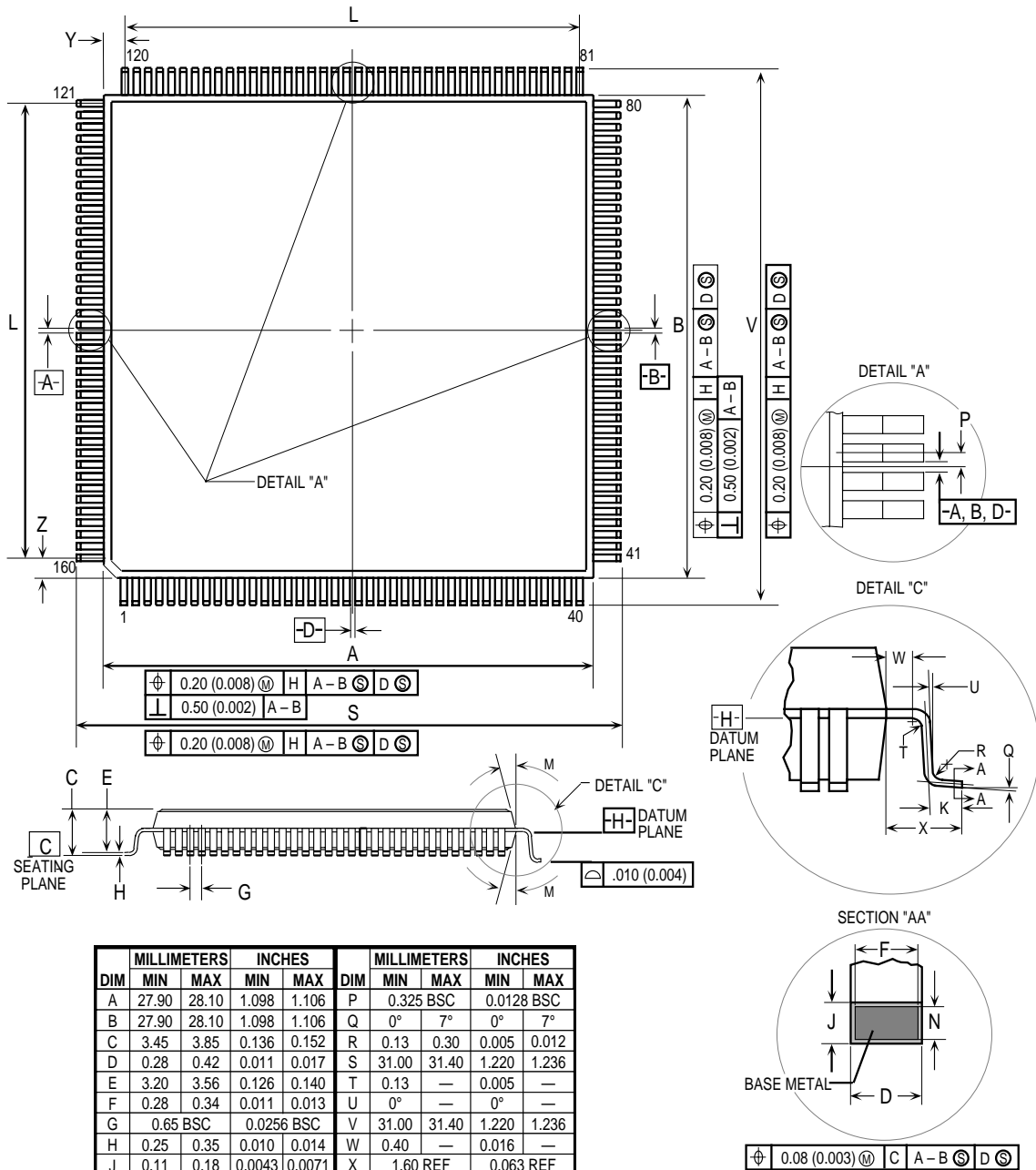


Figure 15-1. MC68322 160-Lead Plastic Quad Flat Pack (PQFP)

### 15.3 MECHANICAL DATA



NOTES:

1. Dimensioning and tolerancing per ANSI Y14.5M, 1982.
2. Controlling dimension: millimeter.
3. Datum plane –H– is located at bottom of lead and is coincident with the lead where the lead exits the plastic body at the bottom of the parting line.
4. Datums A, B, and D to be determined at datum plane –H–.
5. Dimensions S and V to be determined at seating plane –C–.
6. Dimensions A and B do not include mold protrusion. Allowable protrusion is 0.25 (.010) per side. Dimensions A and B do include mold mismatch and are determined at datum plane –H–.
7. Dimension D does not include dambar protrusion. Allowable dambar protrusion shall be 0.08 (.003) total in excess of the D dimension at maximum material condition. Dambar cannot be located on the lower radius or the foot.

Figure 15-2. 160 Pin QFP Package Dimensions

# APPENDIX A

## IN-CIRCUIT EMULATION INTERFACE

This section describes an in-circuit emulation (ICE) device and interface board for the MC68322. A special in-circuit emulation board-out package is available to provide a full-featured emulation device while using standard M68000 emulators. The ICE device uses a 208-pin grid array (PGA) package instead of the production version 160-pin plastic quad flat pack (PQFP). These extra pins, which connect to the MC68322's ICE device, interface between the logic board containing a MC68322 PQFP and a standard MC68000 ICE. The ICE interface contains a PGA connector for the logic board, the MC68322 ICE, a small amount of logic, and a connector for an MC68000 ICE with some minor modifications.

### A.1 ICE INTERFACE SIGNALS

The following pins are used to connect the MC68322 to an ICE. The ICE bond-out is for engineering use only and should not be used in production. The ICE option aids in the reuse of existing emulators for standard M68000 Family processors. Some of these signals are dedicated to this interface and only appear on the ICE bond-out package. In addition, some of these signals are used by the processor bus and an external bus master interface. The direction of the pins is from the perspective of the MC68322. Table A-1 lists a summary of the ICE interface signals.

**Table A-1. ICE Interface Signal Summary**

SIGNAL NAME	MNEMONIC	INPUT/ OUTPUT	ACTIVE STATE	ON $\overline{BG}$	ON HI-Z
Clock (Internal)	I_CLK1	Output	—	No	No
ICE Address Bits	I_A(27–26)	Input	—	—	—
ICE Auto Vector	$\overline{I\_AVEC}$	Output	Low	No	No
ICE Bus Grant	$\overline{I\_BG}$	Input	Low	—	—
ICE Bus Request	$\overline{I\_BR}$	Output	Low	No	No
ICE Enable	$\overline{I\_CEN}$	Input	Low	—	—
ICE Data Latch Enable	$\overline{I\_DLEN}$	Output	Low	No	No
ICE Data Transfer Acknowledge	$\overline{I\_DTACK}$	Output	Low	No	No
ICE Function Code	I_FC(2–0)	Input	—	—	—
ICE Processor Halt	$\overline{I\_HALT}$	Output	Low	No	No



**Table A-1. ICE Interface Signal Summary (Continued)**

SIGNAL NAME	MNEMONIC	INPUT/ OUTPUT	ACTIVE STATE	ON $\overline{BG}$	ON $\overline{HI-Z}$
ICE Lower Data Strobe	$\overline{I\_LDS}$	Input	Low	—	—
ICE Interrupt Priority Level	$\overline{I\_IPL2}$ – $\overline{I\_IPL0}$	Output	Low	No	No
ICE RESET	$\overline{I\_RESET}$	Output	Low	No	No
ICE Upper Data Strobe	$\overline{I\_UDS}$	Input	Low	—	—

### A.1.1 ICE Signal Descriptions

#### $\overline{I\_CLK1}$

**INTERNAL CLOCK**—Output. This signal is used for a 1× clock that is a buffered version of the clock to the MC68EC000 core processor. Typically, this signal is connected directly to the MC68000 ICE clock pin.

#### $\overline{I\_A27}$ – $\overline{I\_A26}$

**INTERNAL ADDRESS**—Input. These signals are address bits that expand the addressing capability of the MC68EC000 core. They are used by the internal system integration module, internal registers, and DRAM accesses. For the most part, these signals will not be used and should be pulled low.

#### $\overline{I\_UDS}$ , $\overline{I\_LDS}$

**INTERNAL UPPER AND LOWER DATA STROBE**—Input. These active low input signals control the flow of data on the data bus, as specified in Table A-2. When  $R/\overline{W}$  is high, the processor reads from the data bus as indicated and when it is low, the processor writes to the data bus. Typically, these signals are connected directly to the MC68000 ICE.

**Table A-2. Data Strobe Control of Data Bus**

$\overline{I\_UDS}$	$\overline{I\_LDS}$	R/ $\overline{W}$	D15–D8	D7–D0
High	High	—	No Valid Data	No Valid Data
Low	Low	High	Valid Data Bits (Read) 15–8	Valid Data Bits (Read) 7–0
High	Low	High	No Valid Data	Valid Data Bits (Read) 7–0
Low	High	High	Valid Data Bits (Read) 15–8	No Valid Data
Low	Low	Low	Valid Data Bits (Write) 15–8	Valid Data Bits (Write) 7–0
High	Low	Low	Valid Data Bits (Write) 7–0*	Valid Data Bits (Write) 7–0
Low	High	Low	Valid Data Bits (Write) 15–8	Valid Data Bits (Write) 15–8*

NOTE: \* Denotes That These Conditions Are A Result Of Current Implementation And May Not Appear On Future Devices.

**I<sub>IPL2</sub>–I<sub>IPL0</sub>**

**INTERNAL INTERRUPT PRIORITY LEVEL**—Output. These three active low outputs indicate the encoded priority level of the device requesting an interrupt. Level 7 is the highest priority while level 0 indicates that no interrupts are requested. Level 7 cannot be masked. The least-significant bit is provided in I<sub>IPL0</sub> and the most-significant bit is contained in I<sub>IPL2</sub>. Typically, these signals are connected directly to the MC68000 ICE.

**I<sub>FC2</sub>–I<sub>FC0</sub>**

**INTERNAL FUNCTION CODE**—Input. These function code input signals indicate the processing mode, user or supervisor, and the cycle type currently being executed (see Table A-3). The information indicated by the function code outputs is valid whenever  $\overline{AS}$  is active. The function code signals are internal signals used to decode an interrupt acknowledge bus cycle. Typically, these signals are connected directly to the MC68000 ICE.

**Table A-3. Function Code Outputs**

FUNCTION CODE OUTPUT			CYCLE TIME
I <sub>FC2</sub>	I <sub>FC1</sub>	I <sub>FC0</sub>	
Low	Low	Low	Undefined, Reserved
Low	Low	High	User Data
Low	High	Low	User Program
Low	High	High	Undefined, Reserved
High	Low	Low	Undefined, Reserved
High	Low	High	Supervisor Data
High	High	Low	Supervisor Program
High	High	High	Interrupt Acknowledge

**I<sub>BR</sub>**

**INTERNAL BUS REQUEST**—Output. This signal is used by the internal bus arbitration control logic. It should be connected directly to the M68000 emulator  $\overline{BR}$  signal.

**I<sub>BG</sub>**

**INTERNAL BUS GRANT**—Input. This signal is used by the internal bus arbitration control logic. It should be connected directly to the M68000 emulator  $\overline{BG}$  signal.

**I<sub>AVEC</sub>**

**INTERNAL AUTOVECTOR**—Output. This signal is used by an emulator to assert the I<sub>AVEC</sub> signal during an interrupt acknowledge cycle. It should be connected directly to the M68000 emulator  $\overline{AVEC}$  or  $\overline{VPA}$  signal.

**$\overline{I\_RESET}$** 

**INTERNAL RESET**—Output. This signal is used by an emulator to assert the  $\overline{RESET}$  signal. Typically this signal is connected directly to the MC68000 ICE.

 **$\overline{I\_HALT}$** 

**INTERNAL HALT**—Output. This signal is used to assert the emulator  $\overline{HALT}$  signal. It should be connected directly to the M68000 emulator  $\overline{HALT}$  signal.

 **$\overline{I\_DLEN}$** 

**ICE DATA LATCH ENABLE**—Output. This signal is used by external latches to latch incoming data before being sent to the emulator.

 **$\overline{ICEN}$** 

**ICE ENABLE**—Input. This signal should be asserted during in circuit emulation because it will force the MC68322 to act as a slave device to an external M68000 processor. It must be asserted or negated before the negation of  $\overline{RESET}$  and remain that way during operation.

## A.2 ICE ADAPTOR BOARD DESIGN

In a typical system, the logic board contains an MC68322 and a PGA connector around the 160-pin QFP MC68322. The PGA connects directly to all the signals of the MC68322. The user plugs the ICE interface into this PGA connector, which asserts the  $\overline{HI-Z}$  input pin on the MC68322 on the logic board, thereby forcing it to three-state. With the MC68322 three-stated, the MC68322 ICE on the ICE interface has control of all the signals on the logic board. The MC68322 ICE takes the place of the MC68322 in the system when the ICE interface is installed.

Normally, the  $\overline{ICEN}$  input pin on the MC68322 ICE is connected to ground, which disables the internal core and uses an MC68000 ICE for the processor. For diagnostic purposes, if  $\overline{ICEN}$  is connected high on the ICE module, then the internal core is used. When an MC68000 ICE is used, IA27–IA26 to the MC68322 ICE are tied to ground. An MC68000 ICE can only access the lower 64M of memory, but an MC68020 ICE can be used instead. If an MC68020 ICE is used, then IA27–IA26 are driven by the MC68020 ICE, which can access the entire 256M of memory. Figure A-1 illustrates the connection between the ICE interface and the logic board.

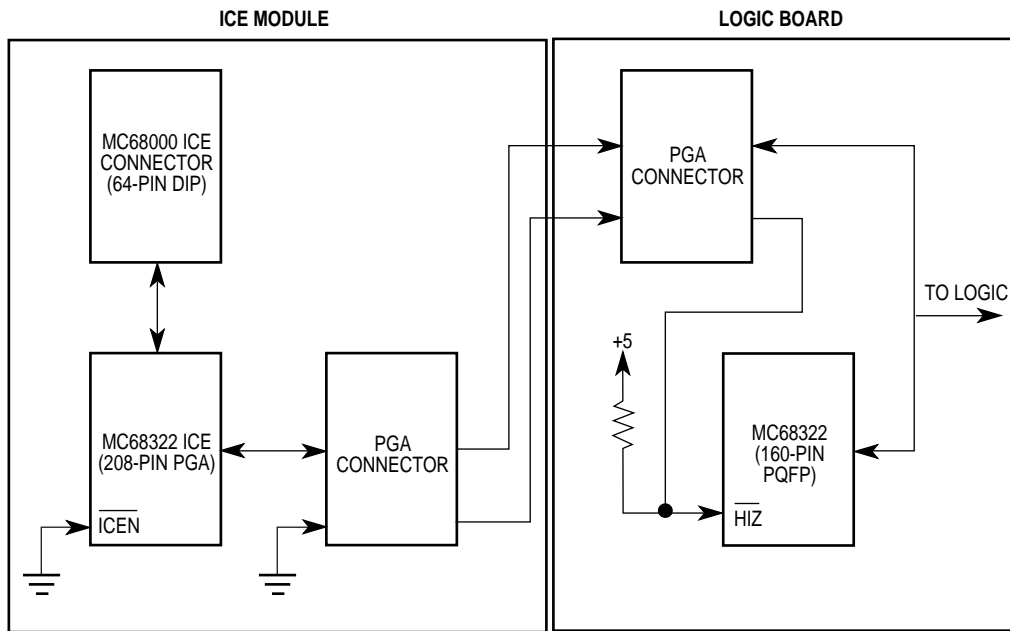


Figure A-1. ICE Interface Block Diagram

### A.3 ICE ADAPTOR BOARD SCHEMATICS

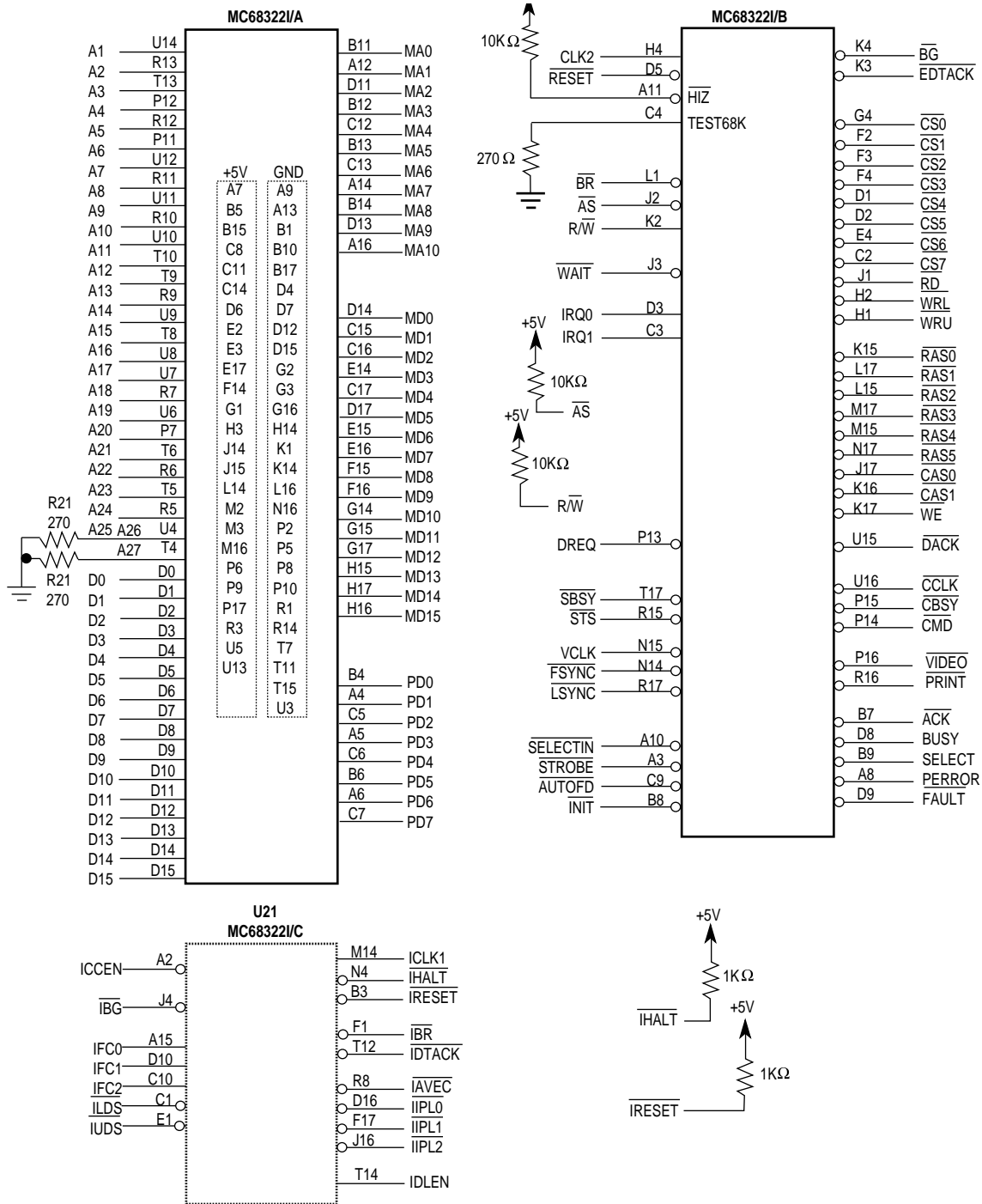


Figure A-2. MC68322 PGA Pinout

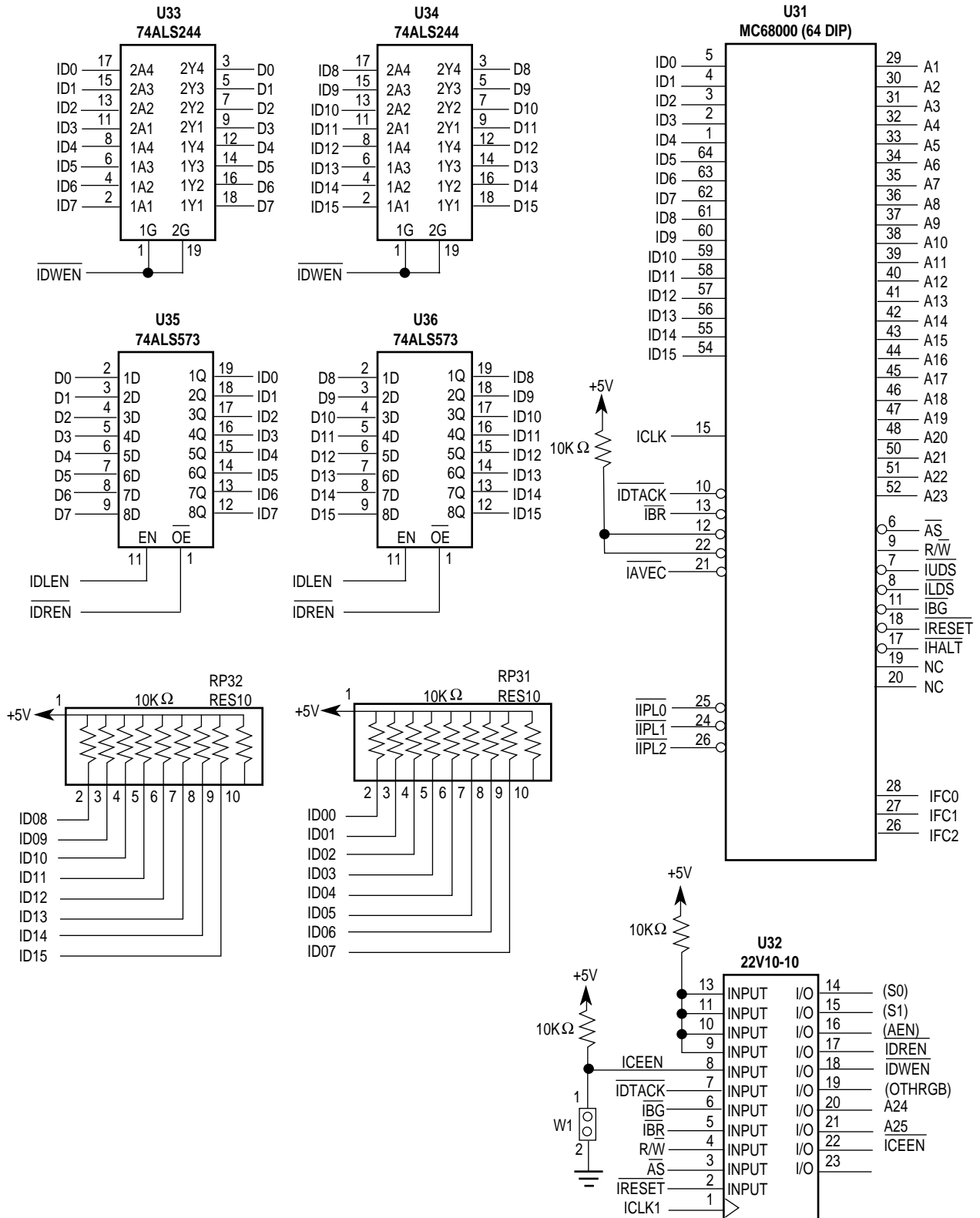


Figure A-3. MC68000 Emulator Connection

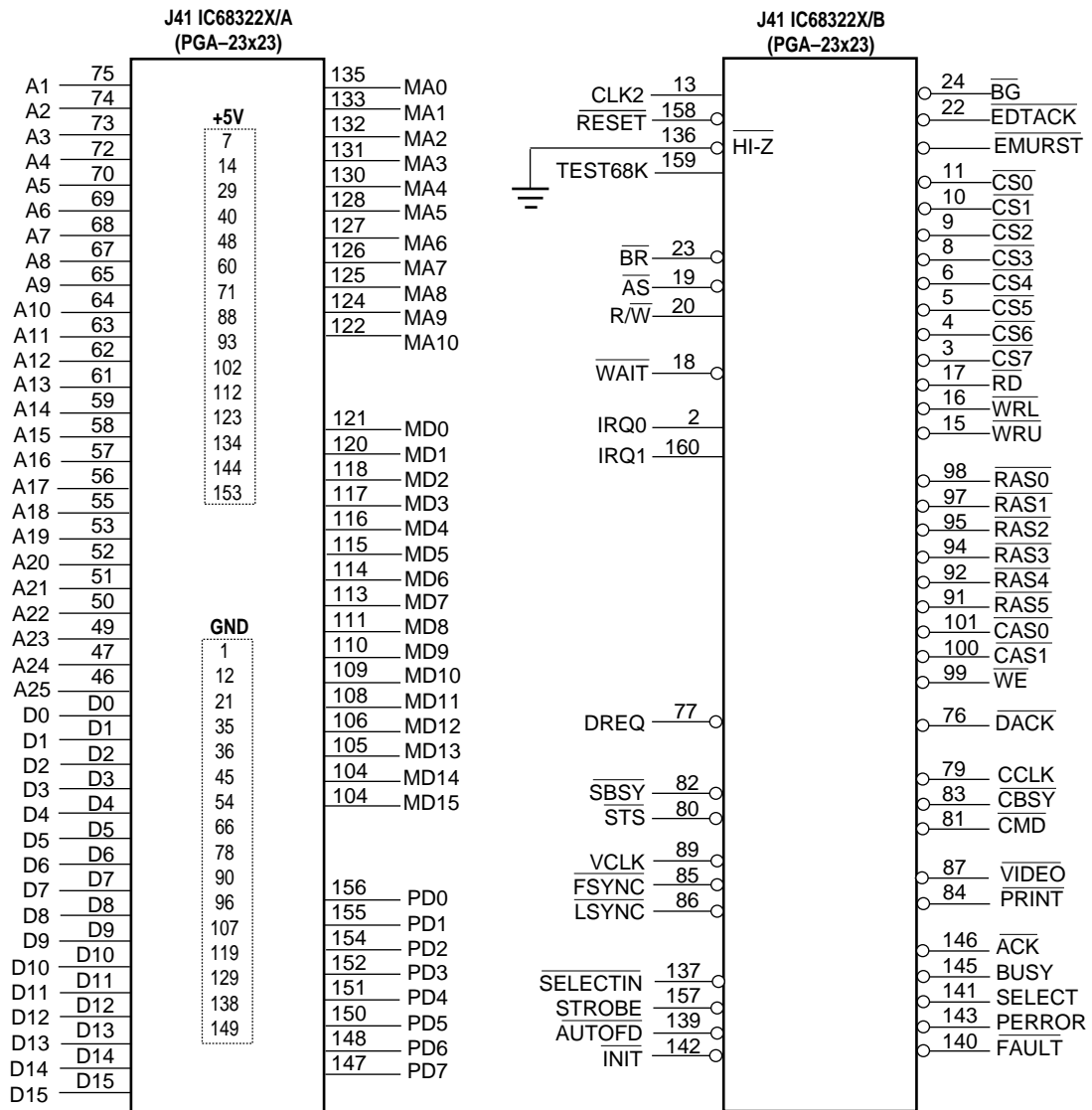


Figure A-4. PGA Connector

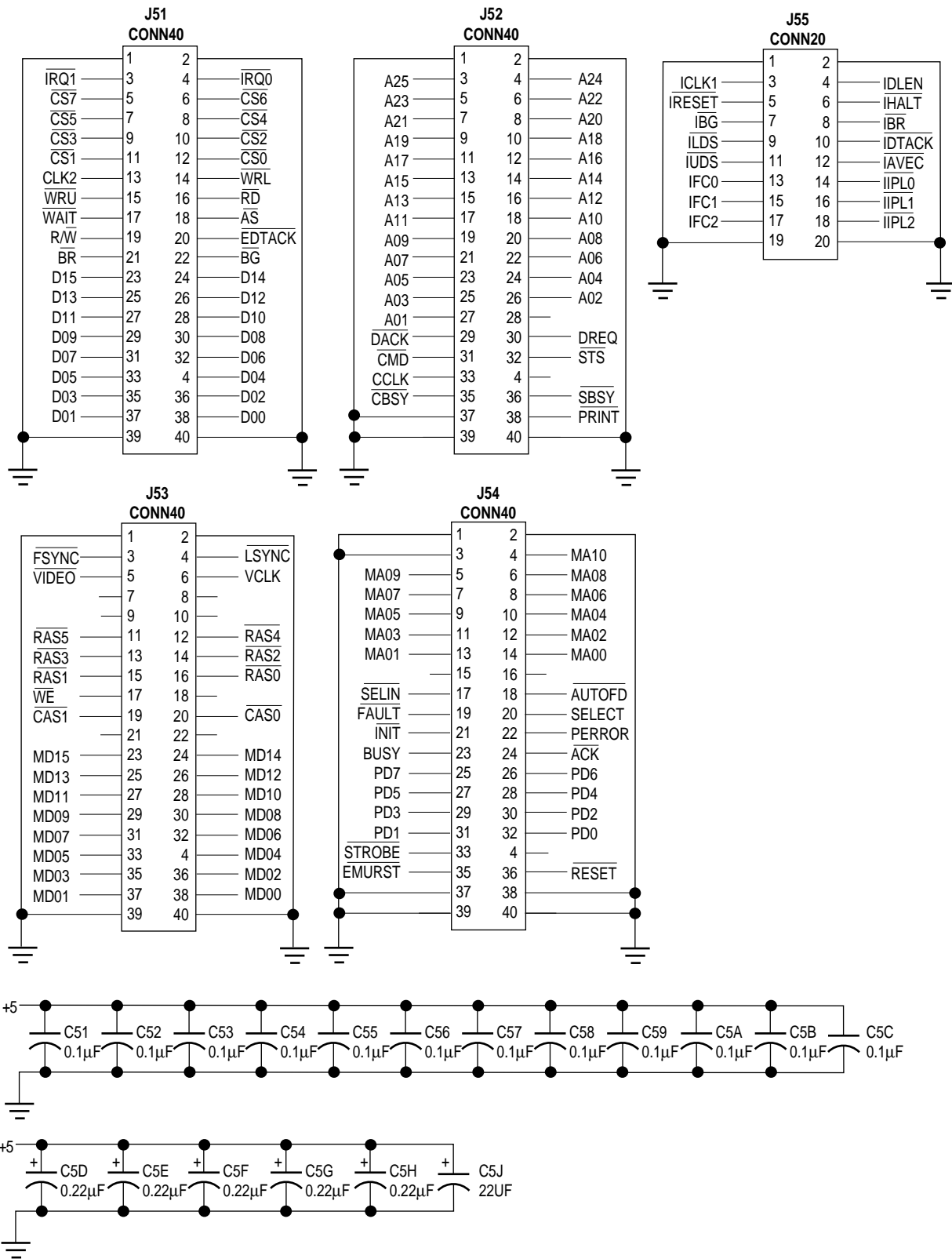
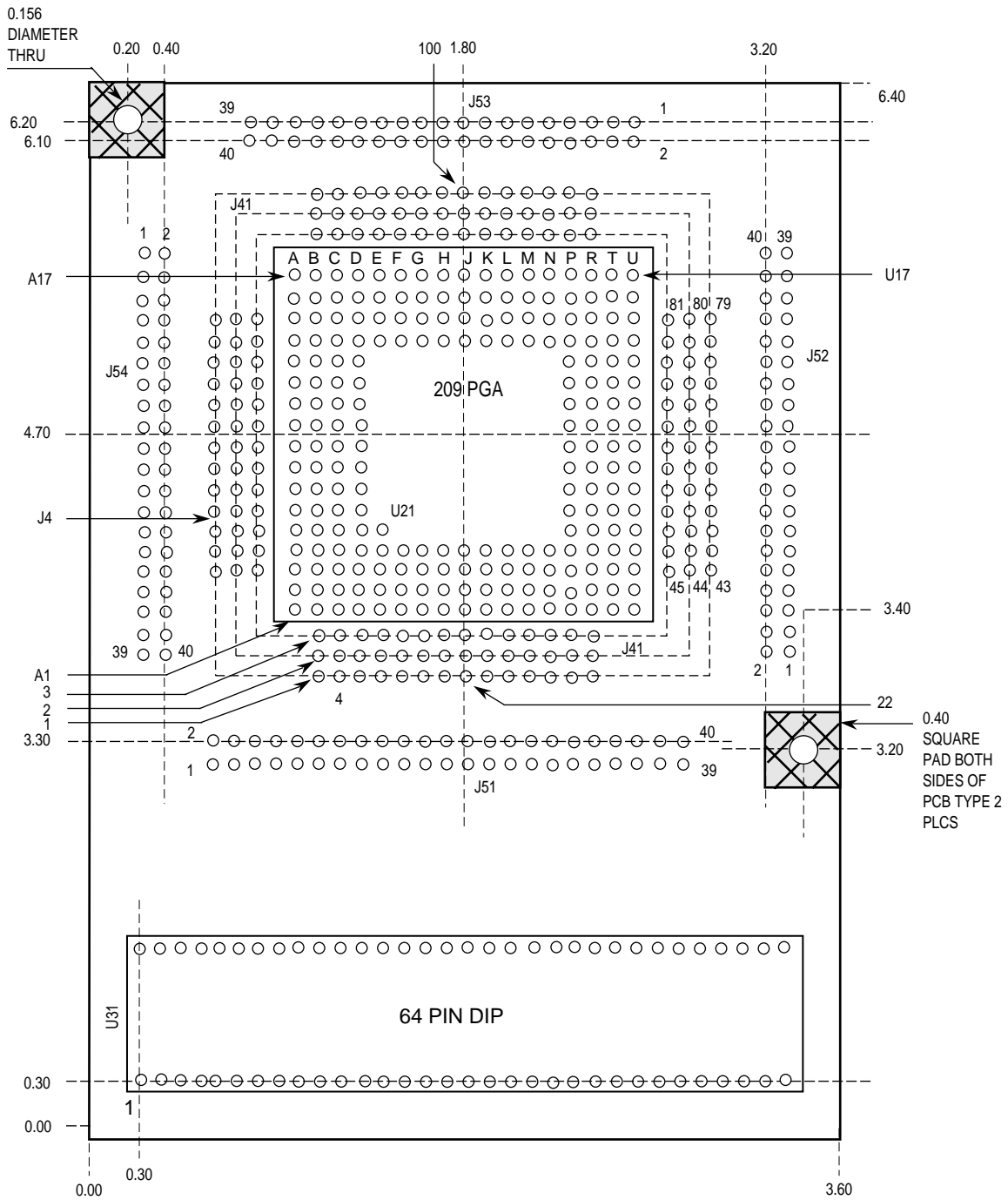


Figure A-5. Test Points





NOTES:

1. Component side is shown.
2. Dimensions are in inches unless otherwise specified.
3. Tolerances: Hole location and diameter:  $\pm 0.010$ .
4. Connectors J51, J53, J54, and J55 are located on the component side of the PCB.
5. Connector J41 is located on the solder side of the PCB.

**Figure A-6. ICE Adaptor Board**

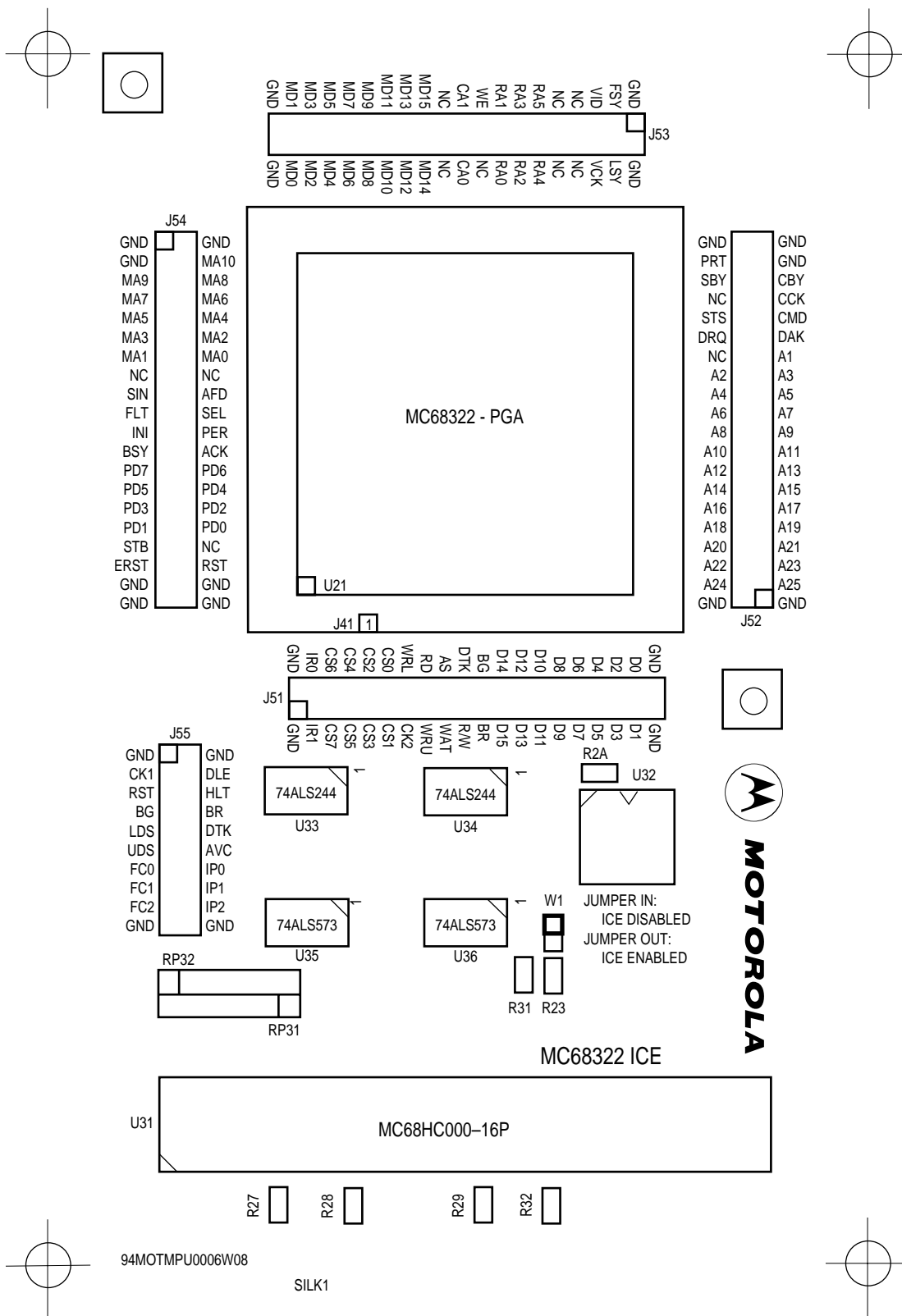


Figure A-7. ICE Adaptor Board—Silkscreen

### A.3.1 In-Circuit Emulation Interface

NUM	CHARACTERISTIC	MIN	MAX	UNIT
65	I_CLK1 Valid From CLK2	2	15	ns
66	I_RESET, I_HALT Valid From CLK1	1	10	ns
67	I_FC2-I_FC0 Set Up Before CLK1	20	—	ns
68	Address Bus, I_A27-I_A26 Set Up Before CLK1	20	—	ns
69	AS, I_UDS, I_LDS, R/W Set Up Before CLK1	20	—	ns
70	I_FC2-I_FC0, Address Bus, I_A27-I_A26, R/W Hold After I_CLK1	0	—	ns
71	CS7-CS0 Valid From AS	—	10	ns
72	DTACK, DATALEN Valid From I_CLK1	1	10	ns
73	Data Bus Driven From I_CLK1 (Read Cycle)	1	10	ns
74	Data Bus Valid From I_CLK1 (Read Cycle)*	1	10	ns
75	Data Bus High Impedance From I_CLK1 (Read Cycle)	1	10	ns
76	Data Bus Set Up Before I_CLK1 (Write Cycle)	20	—	ns
77	Data Bus Hold After I_CLK1 (Write Cycle)	0	—	ns
78	I_AVEC Valid From AS	—	10	ns
79	I_IPL2-I_IPL0 Valid From I_CLK1	1	10	ns
80	I_BR Valid From I_CLK1	1	10	ns
81	I_BG Set Up Before I_CLK1	20	—	ns

NOTE: \* Denotes That This Specification Only Applies To Register And DRAM Read Cycles. During Core Cycles, Read Data Propagates Directly From The Core To The ICE And Is Unaffected By The MC68322.

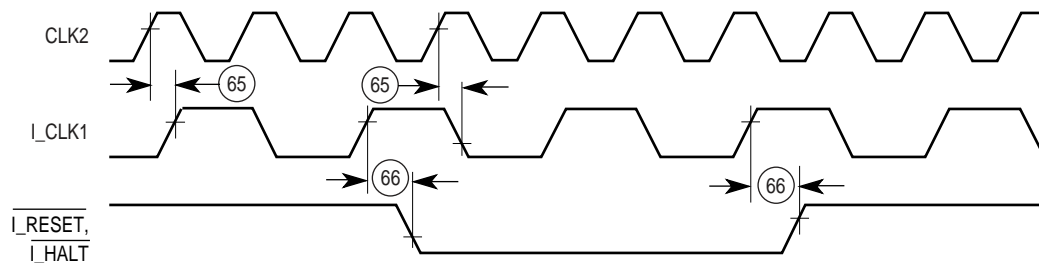


Figure A-8. ICE Reset AC Timing

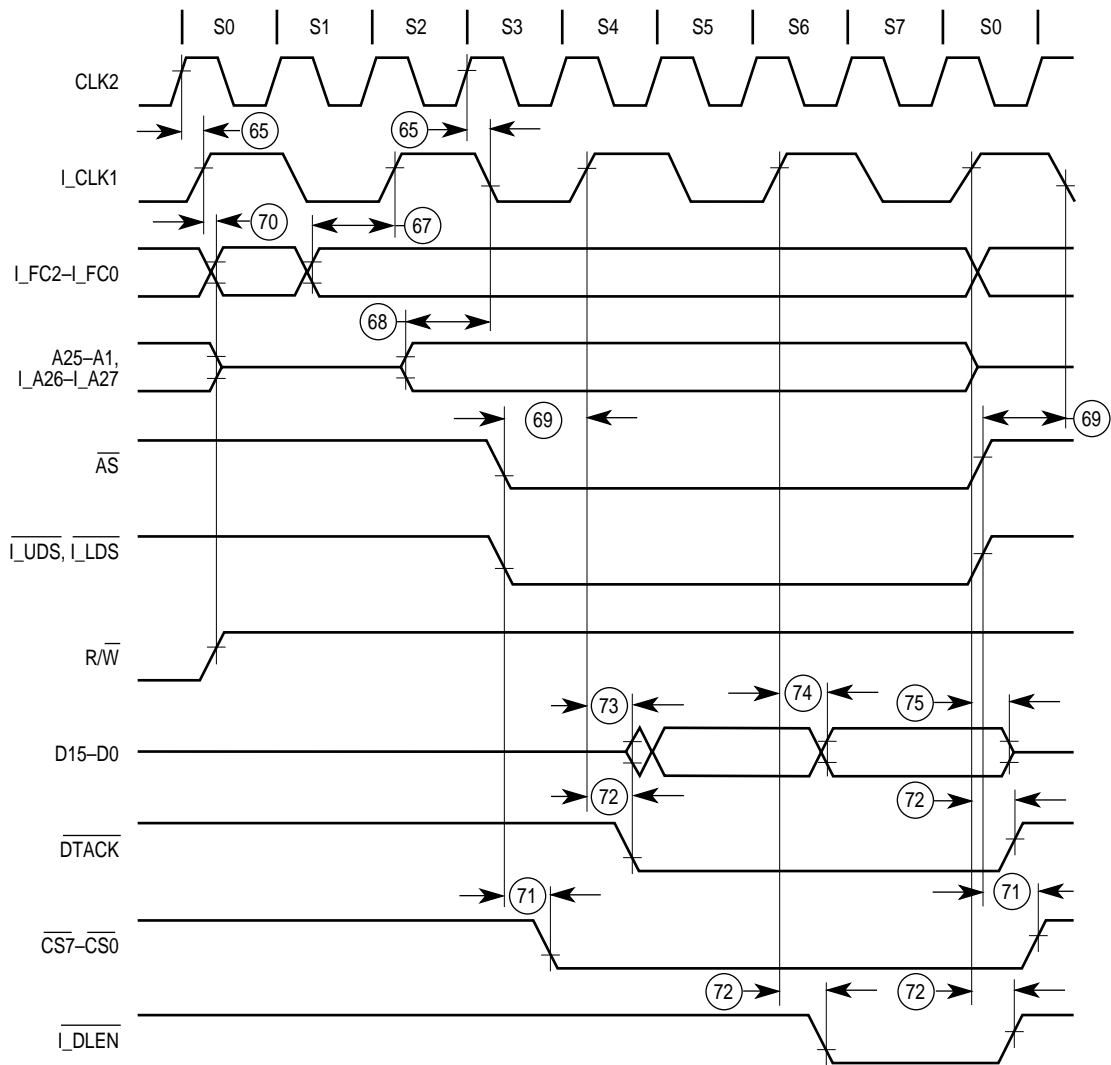


Figure A-9. ICE Read Cycle AC Timing

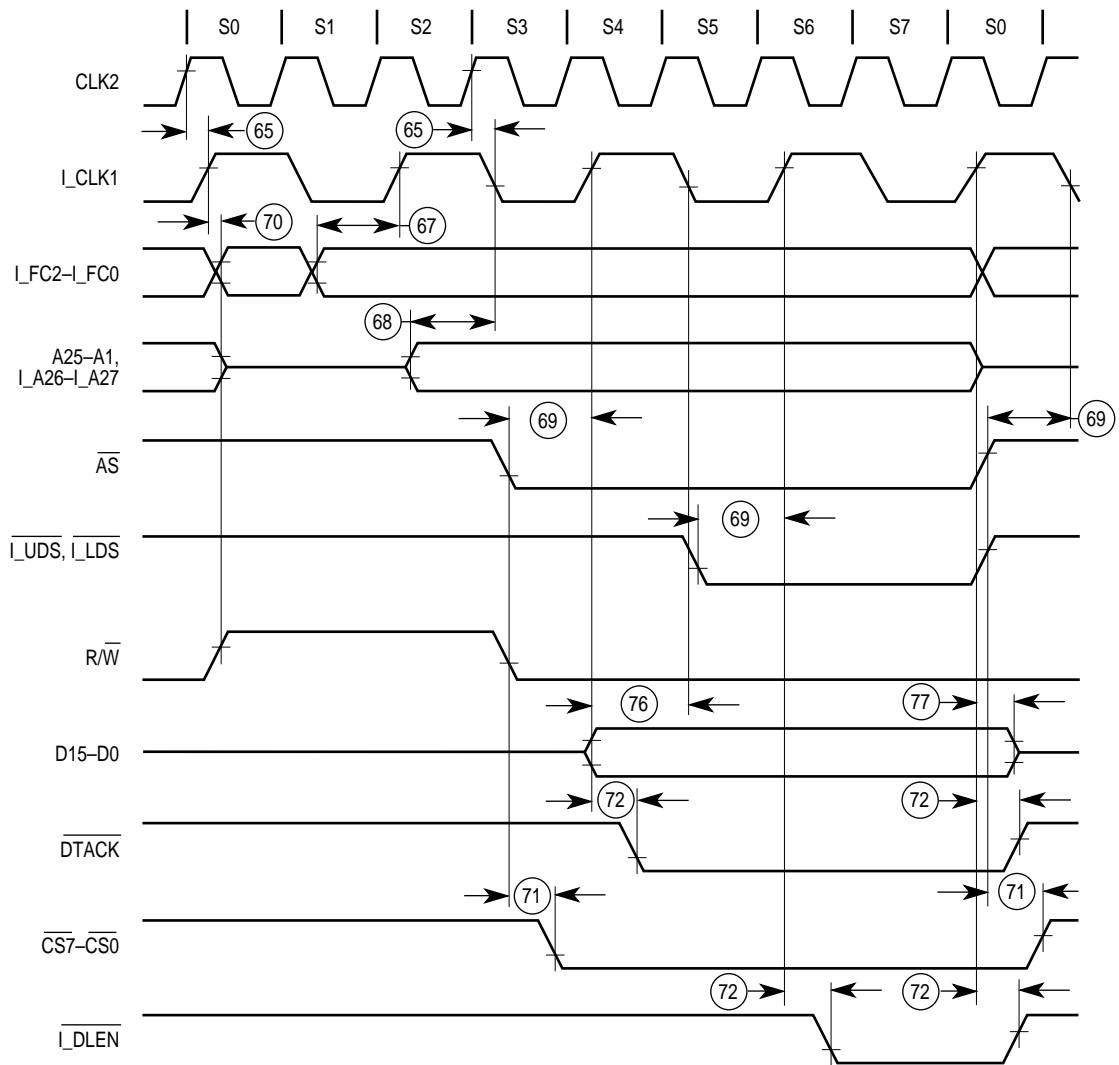


Figure A-10. ICE Write Cycle AC Timing

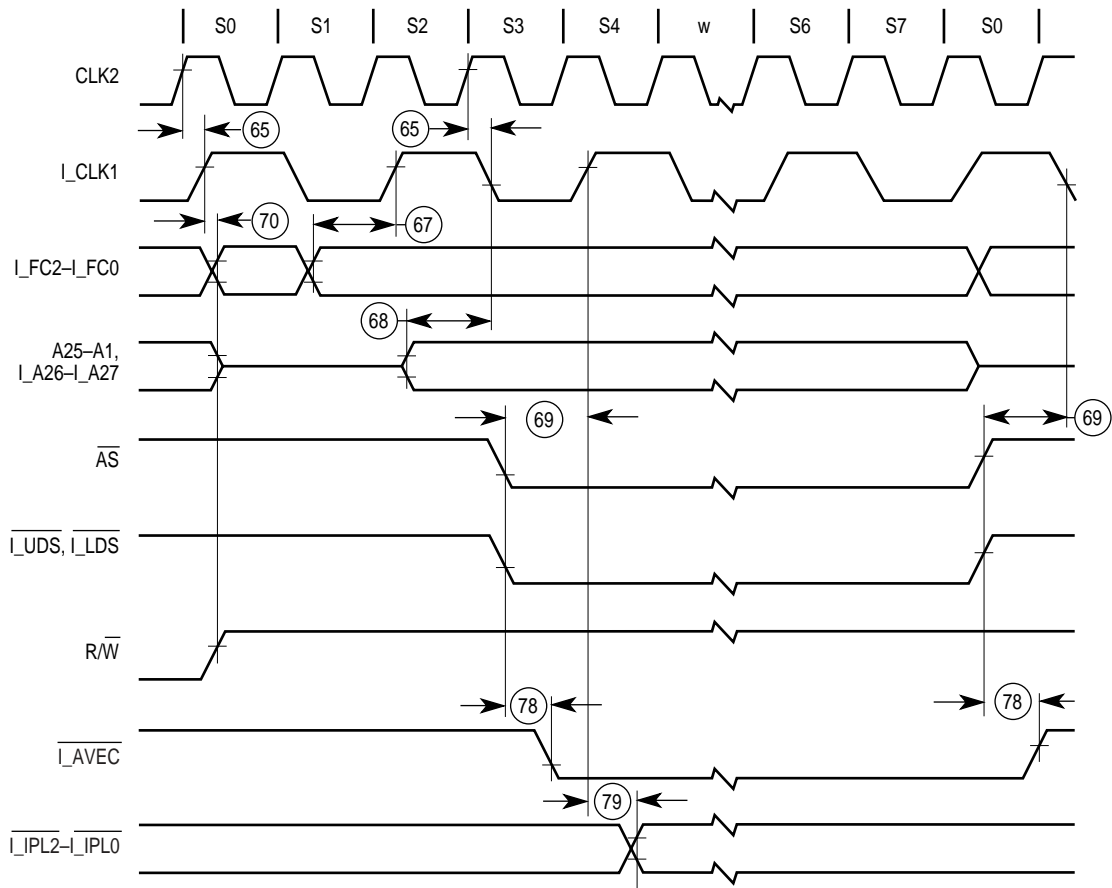


Figure A-11. ICE Interrupt Acknowledge Cycle AC Timing

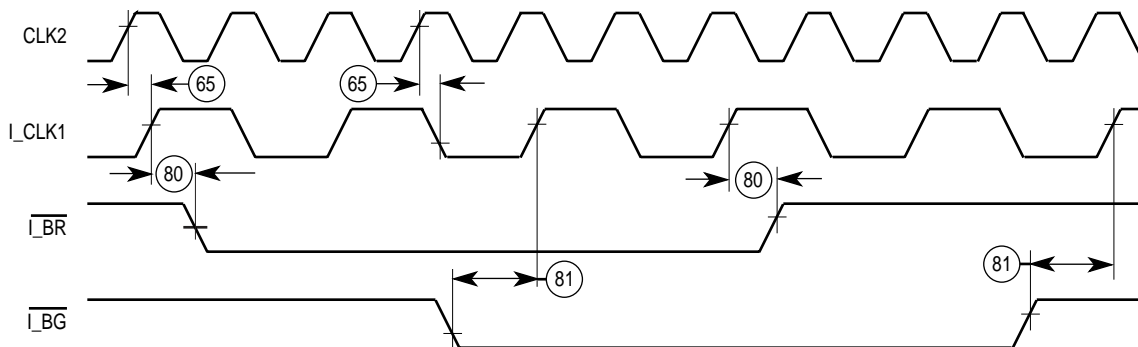
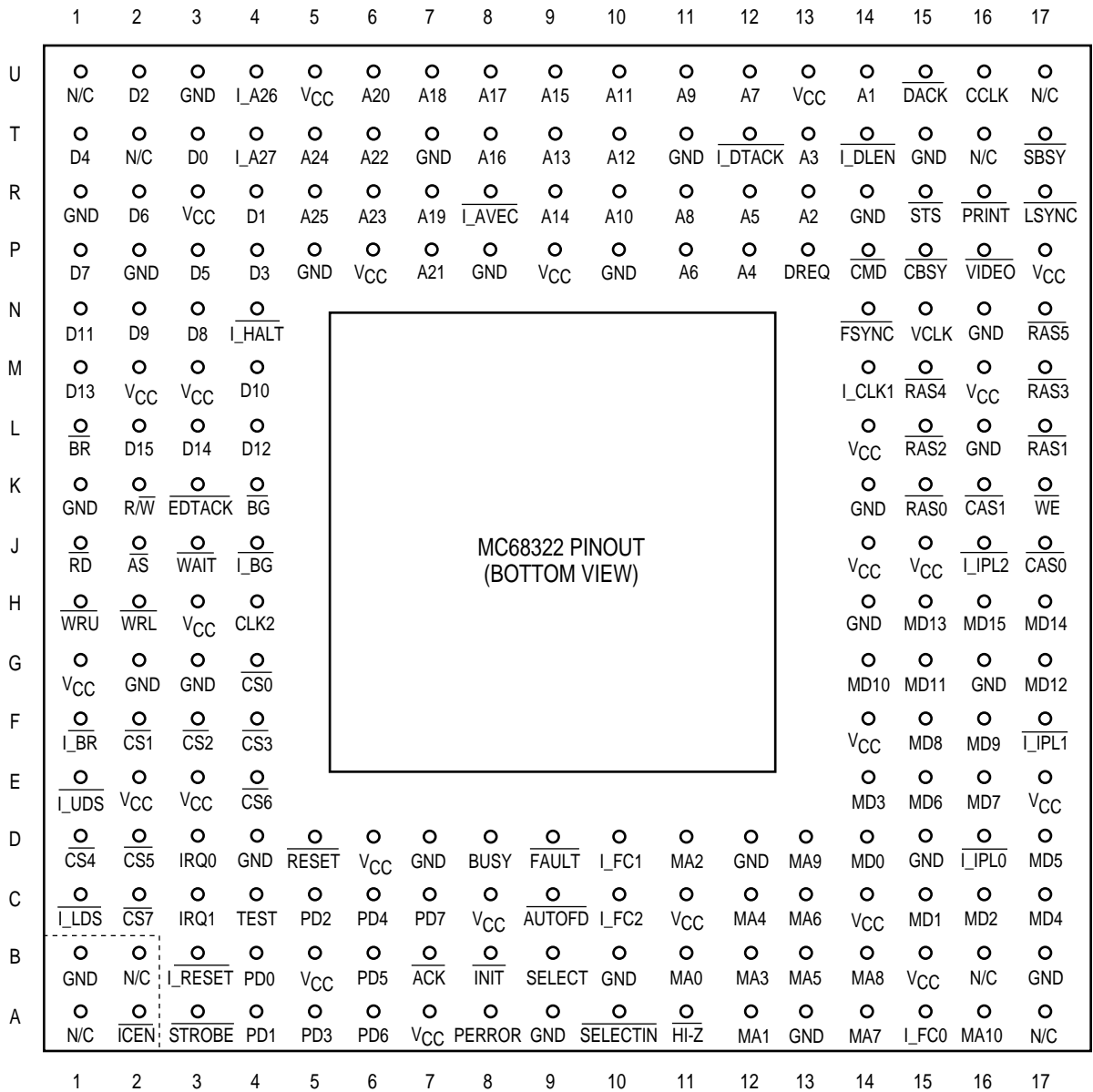


Figure A-12. ICE Bus Arbitration AC Timing

## A.4 ICE PIN ASSIGNMENT

The following figure illustrates the MC68322 ICE pin assignments and case drawing for the 208 PGA package.



# APPENDIX B

## APPLICATIONS

The MC68322 provides a wide variety of configuration possibilities. The sample design described in this section demonstrates the detailed components required to design and build an MC68322-based laser printer. This design demonstrates the typical configuration of printing systems for laser printers, ink jet printers, fax machines, copiers, and many other paper and nonpaper handling applications. Also included in this section is a sample code stub to set up a memory map.

The following schematics describe the required connections for the MC68322, a 512K DRAM, DRAM SIMM, flash EPROM, random control logic, serial EEPROM, in-circuit emulation option, parallel port, and generic print engine interface.

### B.1 CONFIGURING THE MC68322

The MC68322 requires up to two clock sources to properly clock the device. Some designs take advantage of a single oscillator to clock the MC68322. For example, if the video rate is 7.8MHz, then a 31.2MHz source could be used to clock CLK2 and VCLK. Several pull-up resistors may be required to properly negate unused options on the MC68322. Figure B-1 illustrates the MC68322 connection.



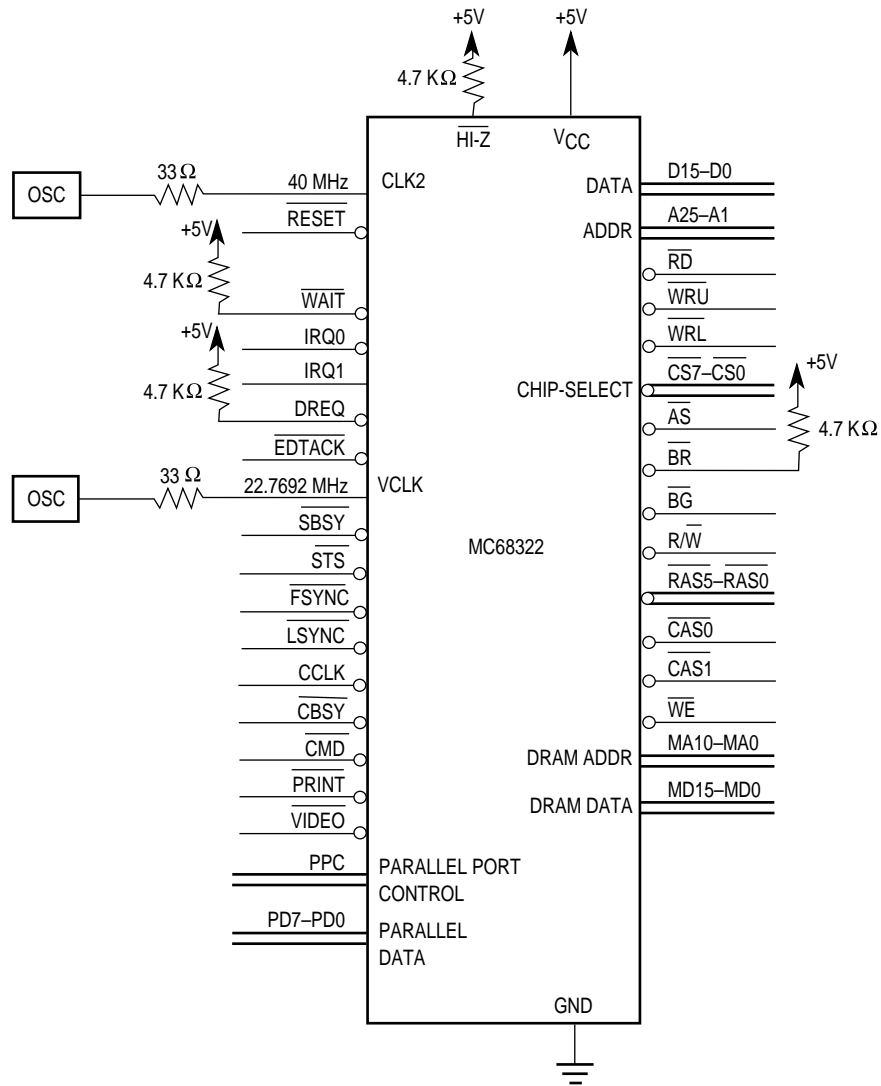
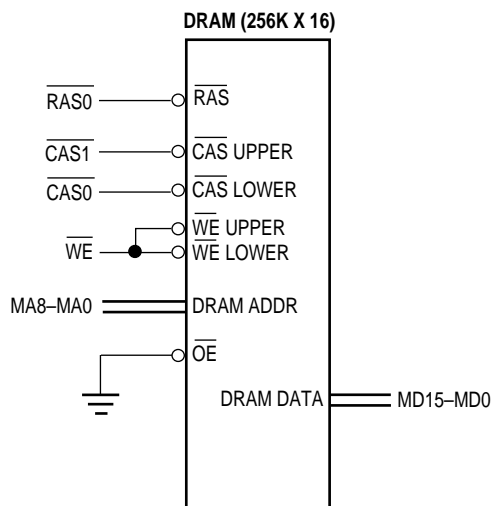


Figure B-1. MC68322 Connection

## B.2 CONFIGURING THE DRAM AND DRAM SIMM

The DRAM can be mounted directly onto the printed circuit board. The following design demonstrates a 512K memory option, which is a minimal memory configuration for the MC68322. However, additional memory can be connected to the MC68322's DRAM controller. Figure B-2 illustrates the DRAM connection.



**Figure B-2. DRAM Connection**

DRAM SIMM modules may be added to provide more memory, which is necessary in some designs. The DRAM SIMM module does not have to be buffered. However, using buffers to isolate the main memory from a memory module can improve design reliability and prevent field failure(s). Due to the unknown nature of the SIMM modules inserted into the design, isolation resistors are helpful to reduce undershoot and other electrical problems resulting from driving a large capacitive load. Figure B-3 illustrates the DRAM SIMM connection.

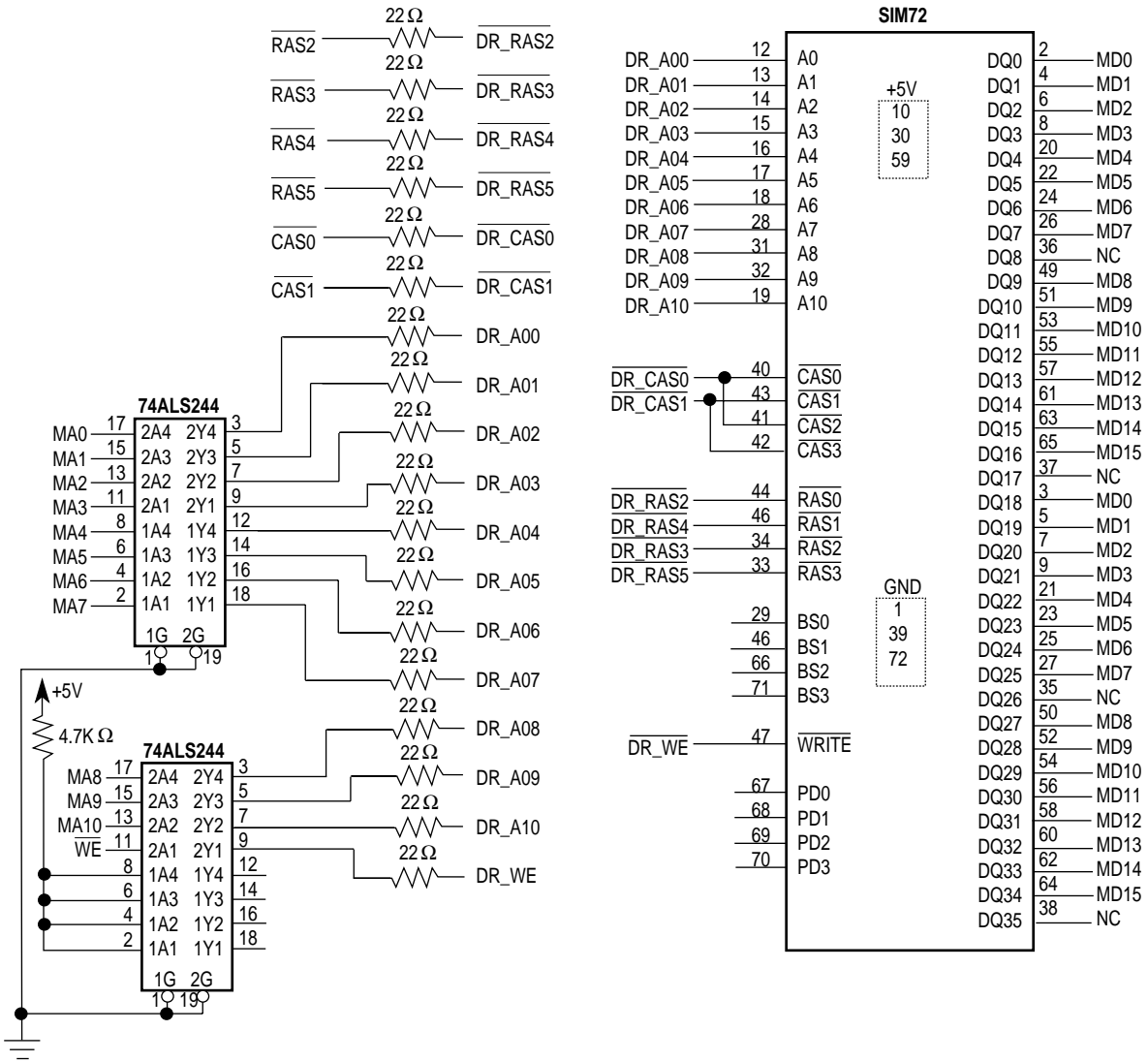


Figure B-3. DRAM SIMM Connection

### B.3 CONFIGURING THE FLASH EPROM

The flash EPROM in this design demonstrates the methodology used to connect a typical PROM, ROM or EPROM interface. Different EPROM devices can be used with minimal or no changes required. This design demonstrates four banks of flash memory connected to the MC68322, which provides 4M of main memory to support the core with instruction memory, built-in fonts, and permanent data structures. Figure B-4 illustrates the flash EPROM connection.

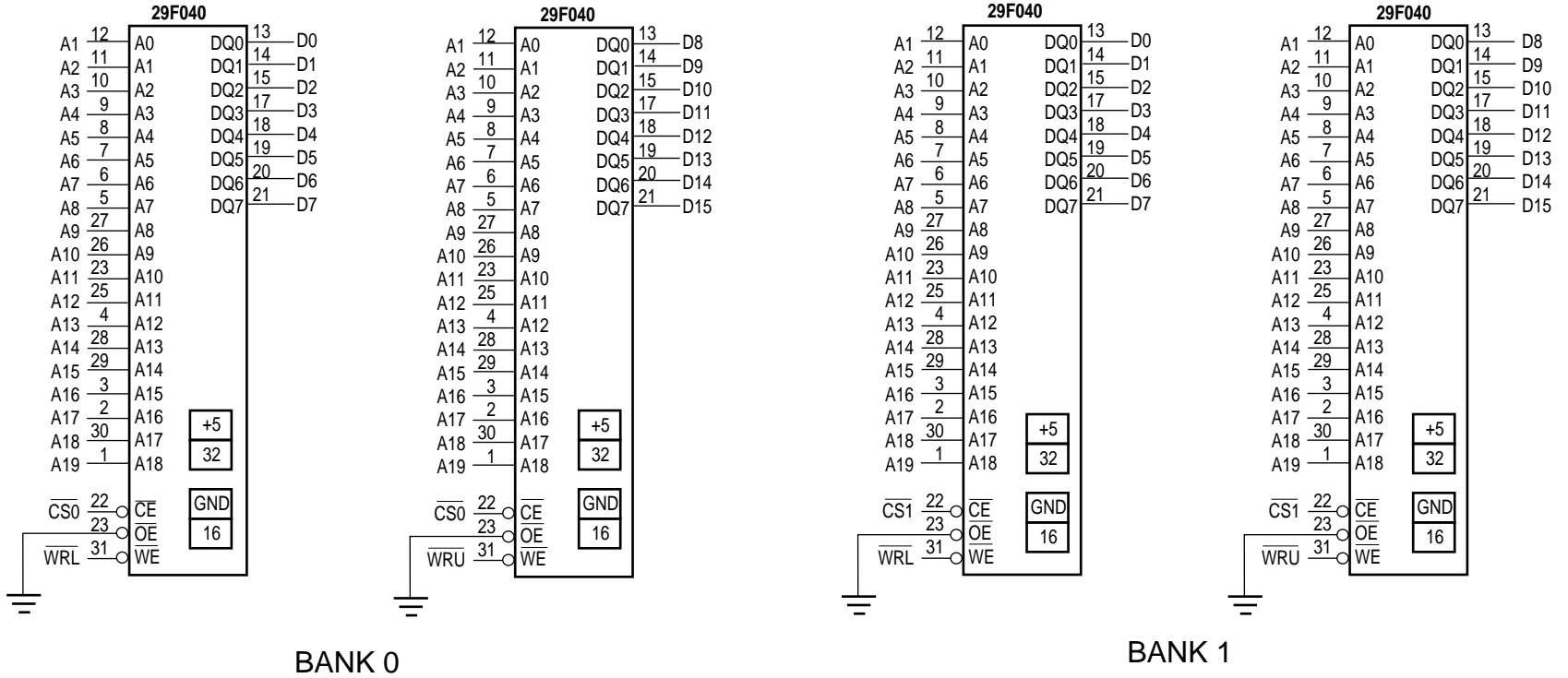


Figure B-4. Flash EEPROM Connection (1 of 2)

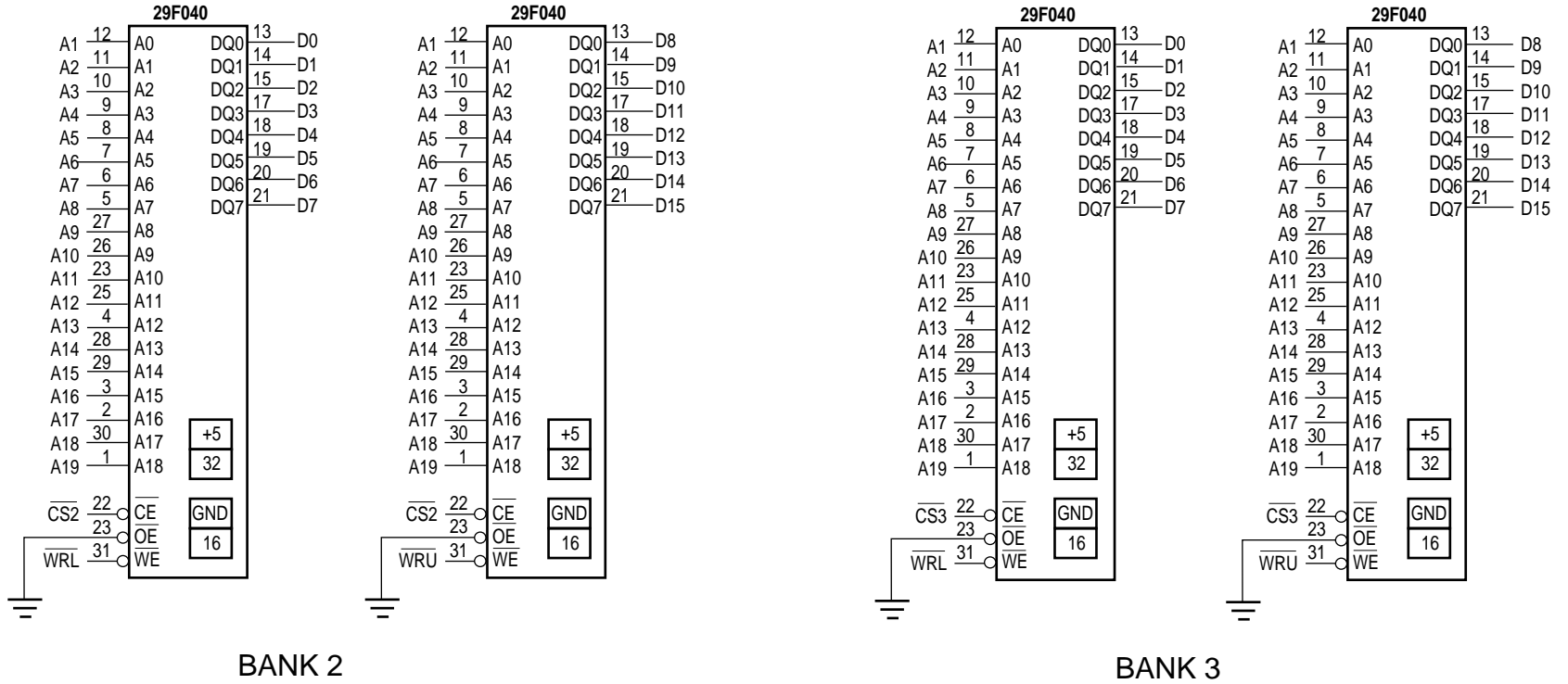
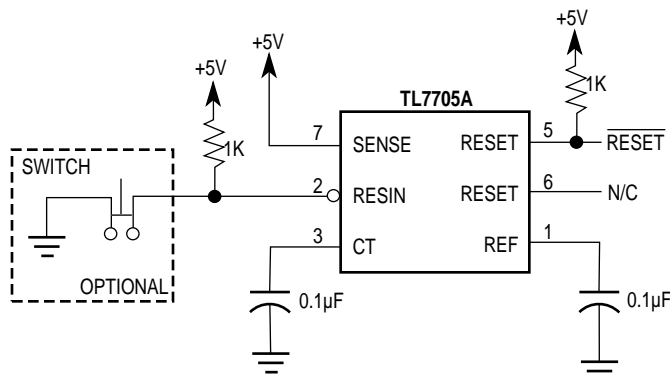


Figure B-4. Flash EEPROM Connection (2 of 2)

## B.4 CONFIGURING THE RANDOM CONTROL LOGIC

The MC68322 requires a minimal amount of external control logic, but should have a reliable reset circuit to sense power-up, low voltage, and push button resets. This sample design includes a Texas Instruments device (TL7705A) to supply a reliable reset and Figure B-5 illustrates reset circuit usage. Other devices from Motorola and Dallas Semiconductor have suitable reset circuits for the MC68322. A resistor-capacitor reset circuit is not recommended for reliable power-up and low-voltage resets.



**Figure B-5. Reset Circuit**

Other random logic, such as external latches or buffers, may be required to interface with the print engine. This sample design has two latches to interface with the front panel and serial EEPROM. Two buffers are used to filter and clean up the incoming signals from the print engine. Figure B-6 illustrates the front panel buffers and latches.

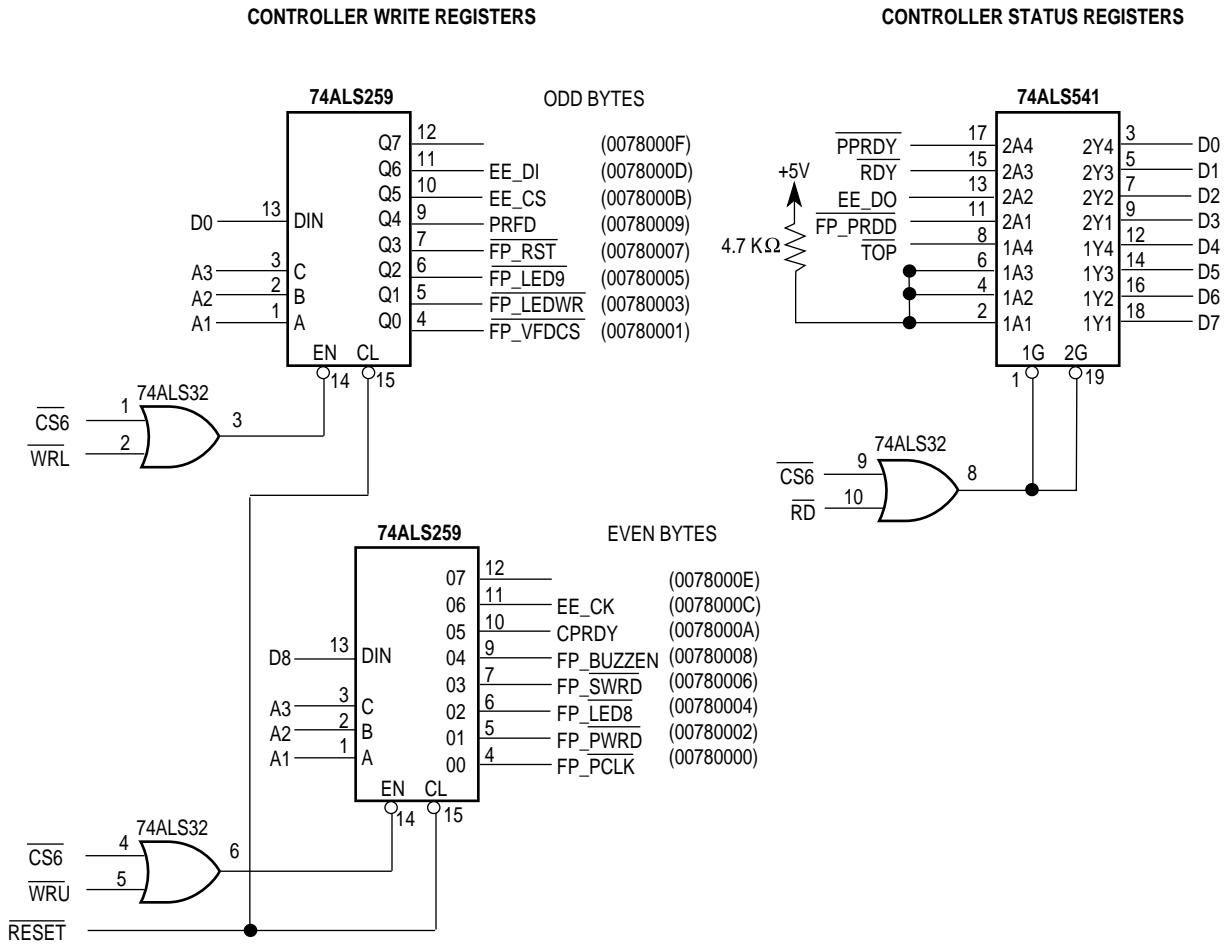


Figure B-6. Front Panel Buffers and Latches

## B.5 CONFIGURING THE SERIAL EEPROM

Often a serial EEPROM is used to store field programmable or default information, such as page count, printer name, Ethernet address, and resolution. This interface uses four signals from the random logic interface. The serial interface utilizes the latches and buffers that are shared with the front panel interface. Figure B-7 illustrates the serial EEPROM connection.

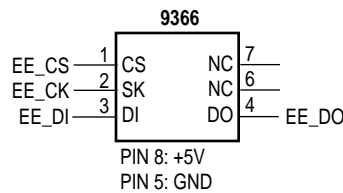


Figure B-7. 4-Kbit Serial EEPROM Connection

## B.6 CONFIGURING THE IN-CIRCUIT EMULATION

The MC68322 provides an ICE option, which is included here. The ICE connection is a simple pin-grid-array to an ICE board available from Motorola. The ICE option can be implemented in the prototype stages of designing a printing system and can remain available through production as a method of testing, debugging, and field service. See **Appendix A In-Circuit Emulation Interface** for ICE board specifications.

To provide an ICE option, the  $\overline{\text{HI-Z}}$  signal should be connected to a pull-up resistor and not directly to  $V_{CC}$ . The ICE PGA pattern should be placed on the PCB and typically the MC68322 can be placed inside the cut-out section of the PGA. All signals from the MC68322 160-pin quad flat pack (QFP) must be connected to the ICE PGA connector. The special version of the MC68322, along with the ICE board, allow a standard MC68000 DIP format emulator to be used. This type of interface allows existing emulators to be reused with virtually no additional investment. Figure B-8 illustrates the MC68322 ICE interface.

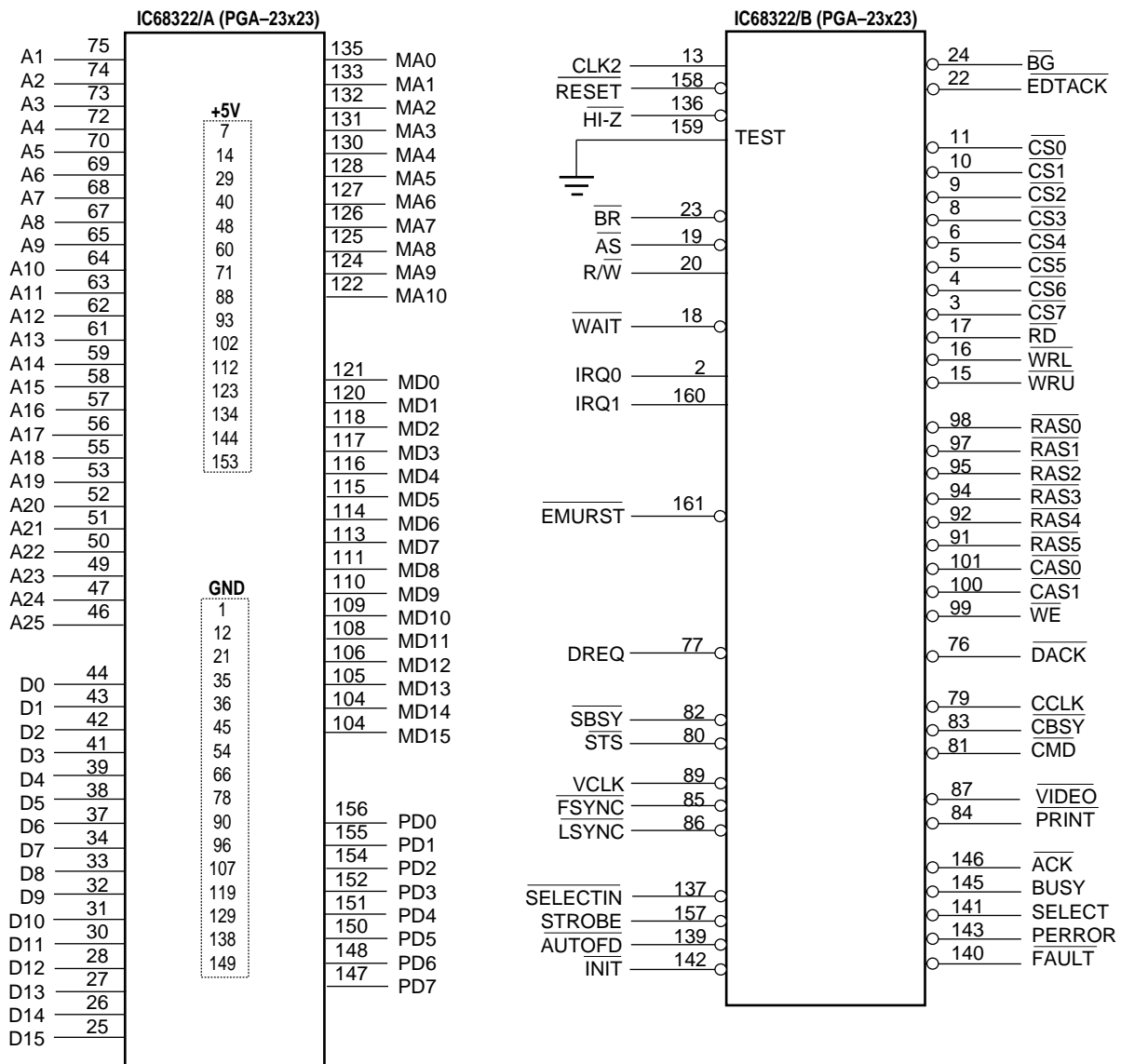


Figure B-8. MC68322 ICE Interface



## B.7 CONFIGURING THE PARALLEL PORT

The MC68322 provides a direct connection, with no external buffers or latches, to virtually all standard computer parallel ports available. The MC68322 can support 2M/sec and higher data rates when the internal DMA is utilized. Additionally, the parallel port supports the IEEE-1284 interface. Figure B-9 demonstrates the interface to a parallel port connector. External resistors are needed in many implementations to avoid ringing, RF noise, and a powered down host situation. The powered down host situation exists when the host computer is turned off while the printer is still powered up or vice-versa. The resistors help to avoid damage to the printer or host computer in the event that this situation exists for extended amounts of time.

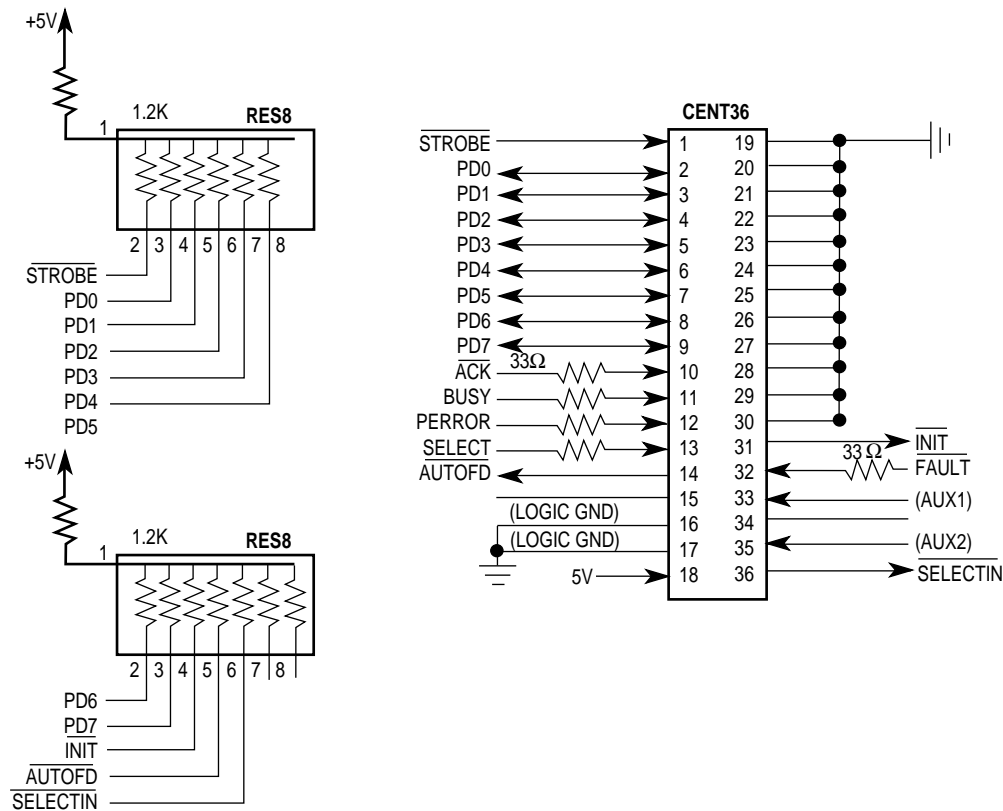


Figure B-9. Parallel Port Connector Interface

## B.8 CONFIGURING THE GENERIC PRINT ENGINE INTERFACE

The print engine interface requires a minimal amount of external logic to connect to most print engines. Due to the different front panel interfaces available from print engine manufactures, the MC68322 may require an external latch or two as demonstrated here. However, the primary handshake and video signals can be connected directly to the print engine in most cases. Figure B-10 illustrates the print engine interface. Notice that the external 555 timer can be eliminated if the  $\overline{BG}$  signal is used in its alternate function (see **Appendix D Alternate Pin Functions**).

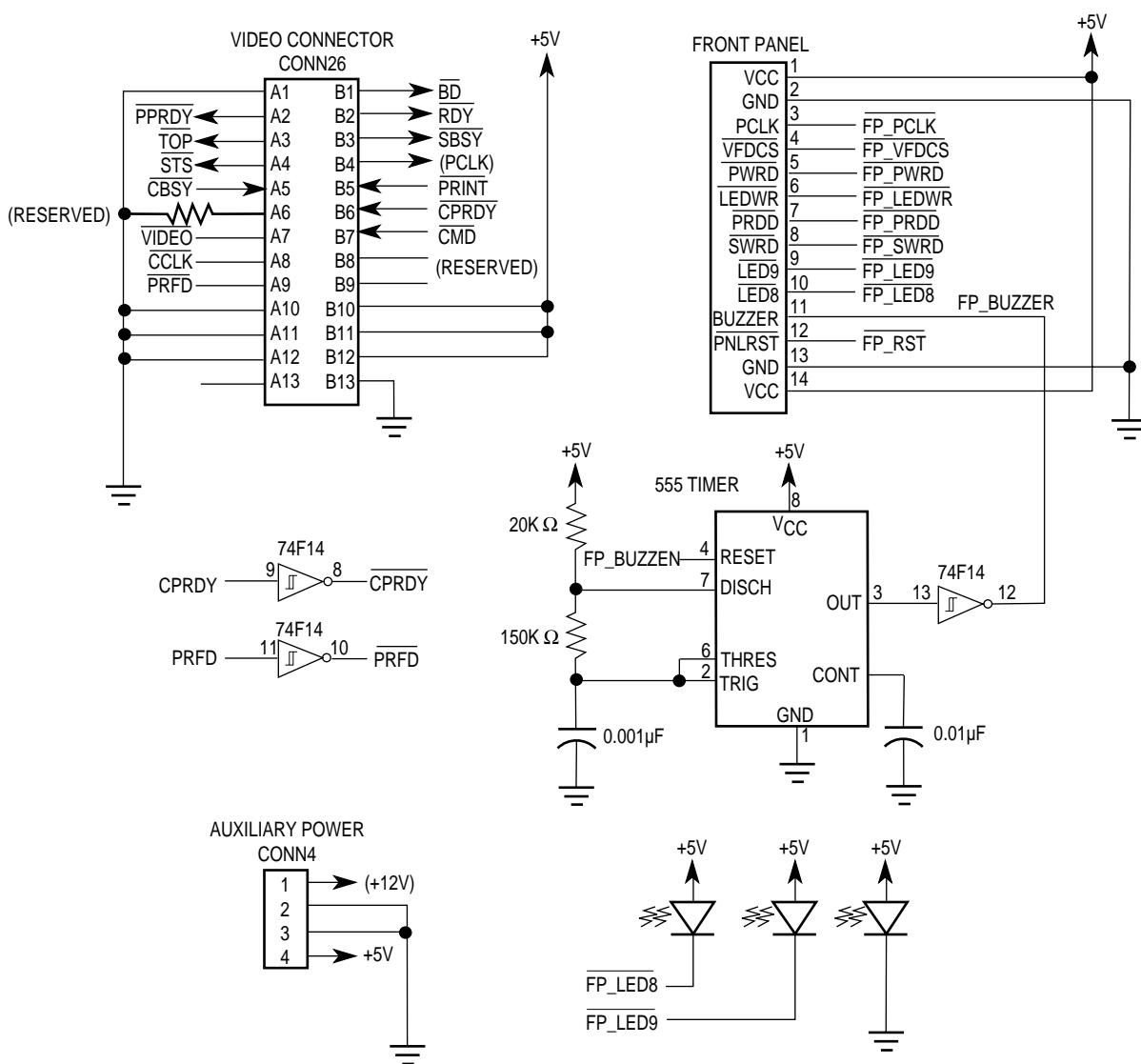
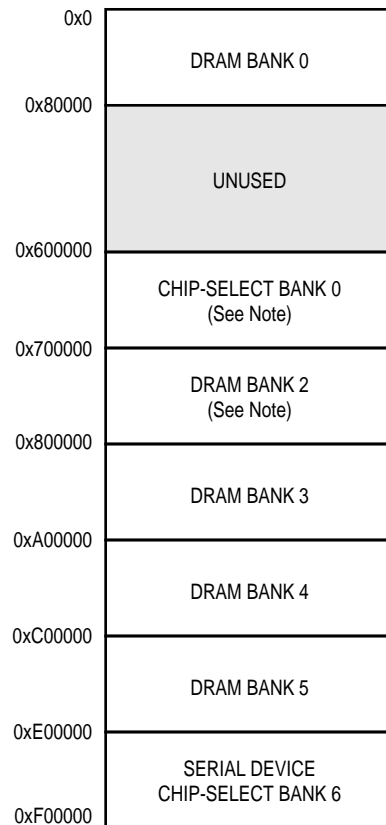


Figure B-10. Print Engine Interface

## B.9 MC68322 MEMORY MAP INITIALIZATION EXAMPLE

The following code example sets up the memory map illustrated in Figure B-11.



NOTE: The chip-select banks have priority over the DRAM banks.

**Figure B-11. Initialized Memory Map From Code Example**

## B.10 MC68322 INTERNAL REGISTERS SAMPLE CODE

```

BASE322      EQU      $00fff000
ADDR0        EQU      $0
TIMING0      EQU      $2
ADDR1        EQU      $10
TIMING1      EQU      $12
ADDR2        EQU      $20
TIMING2      EQU      $22
ADDR3        EQU      $30
TIMING3      EQU      $32
ADDR4        EQU      $40
TIMING4      EQU      $42
ADDR5        EQU      $50
TIMING5      EQU      $52
ADDR6        EQU      $60
TIMING6      EQU      $62
ADDR7        EQU      $70
TIMING7      EQU      $72
RECOVERY     EQU      $82
DRAM0        EQU      $100
DRAM1        EQU      $110
DRAM2        EQU      $120
DRAM3        EQU      $130
DRAM4        EQU      $140
DRAM5        EQU      $150
DRAM_CTRL    EQU      $160
DRAM_REF     EQU      $162

```

SECTION code

```

XDEF          start
XREF          main

```

```

start:  move.l   #$0,a5          * Clear a5
        move.l   #$0,a6          * Clear frame pointer
        move.w   #$2700,sr       * Set up status register
        lea     BASE322,a0       * Set a0 to base address of 68322
        move.w   #$0c18,ADDR0(a0) * Set CS0 to 0x600000-0x6fffff
        move.w   #$0003,TIMING0(a0)
        move.w   #$1010,ADDR1(a0) * Set CS1 to 0x400000-0x5fffff
        move.w   #$0003,TIMING1(a0)
        move.w   #$1008,ADDR2(a0) * Set CS2 to 0x200000-0x3fffff
        move.w   #$0003,TIMING2(a0)
        move.w   #$0c38,ADDR6(a0) * Set CS6 to 0xe00000-0xefffff
        move.w   #$3434,TIMING6(a0)
        move.w   #$0c3c,ADDR7(a0) * Set CS7 to 0xf00000-0xfffff
        move.w   #$3434,TIMING7(a0)
        move.w   #$1540,RECOVERY(a0) * Set recovery for back to back access
        move.w   #$0200,DRAM0(a0) * DRAM0 = 0x0 - 0x80000
        move.w   #$0000,DRAM1(a0) * Disabled
        move.w   #$040c,DRAM2(a0) * DRAM2 = 0x600000 - 0x7fffff
        move.w   #$0410,DRAM3(a0) * DRAM3 = 0x800000 - 0x9fffff
        move.w   #$0414,DRAM4(a0) * DRAM4 = 0xa00000 - 0xbfffff
        move.w   #$0418,DRAM5(a0) * DRAM5 = 0xc00000 - 0xdfffff
        move.w   #$0000,DRAM_CTRL(a0)
        move.w   #$0020,DRAM_REF(a0)
        bra     main          * Jump to Main()
end

```

# APPENDIX C

## MEMORY-MAPPED REGISTER SUMMARY

This section summarizes the memory-mapped registers for the MC68322. The following table contains the name of the register, an image of the register in memory with all of its associated fields, the address of the register at startup, the register's encoding at startup, and the page number where the register is described.



**Note:** All shaded areas are reserved for future use and should always be written as zero.

**Table C-1. Memory-Mapped Register Set**

REG	MEMORY MAP															ADDRESS	VALUE AT RESET	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1		0	HIGH BYTE
CSR0	SIZE			BASE ADDRESS (A27–A18)												00FFF000	XX011000	00000000
	WHLD	WSET	WACC			RHLD	RSET	RACC			00FFF002	11001111	11001111					
CSR1	SIZE			BASE ADDRESS (A27–A18)												00FFF010	XX000000	00000000
	WHLD	WSET	WACC			RHLD	RSET	RACC			00FFF012	00000000	00000000					
CSR2	SIZE			BASE ADDRESS (A27–A18)												00FFF020	XX000000	00000000
	WHLD	WSET	WACC			RHLD	RSET	RACC			00FFF022	00000000	00000000					
CSR3	SIZE			BASE ADDRESS (A27–A18)												00FFF030	XX000000	00000000
	WHLD	WSET	WACC			RHLD	RSET	RACC			00FFF032	00000000	00000000					
CSR4	SIZE			BASE ADDRESS (A27–A18)												00FFF040	XX000000	00000000
	WHLD	WSET	WACC			RHLD	RSET	RACC			00FFF042	00000000	00000000					
CSR5	SIZE			BASE ADDRESS (A27–A18)												00FFF050	XX000000	00000000
	WHLD	WSET	WACC			RHLD	RSET	RACC			00FFF052	00000000	00000000					
CSR6	SIZE			BASE ADDRESS (A27–A18)												00FFF060	XX000000	00000000
	WHLD	WSET	WACC			RHLD	RSET	RACC			00FFF062	00000000	00000000					
CSR7	SIZE			BASE ADDRESS (A27–A18)												00FFF070	XX000000	00000000
	WHLD	WSET	WACC			RHLD	RSET	RACC			00FFF072	00000000	00000000					
CSDTR	WHLD	WSET	WACC			RHLD	RSET	RACC			00FFF080	00000000	00000000					
CSRR	REC			RECOVERY SELECT												00FFF082	XX000000	00000000
DRAM0	RM		SIZE	BASE ADDRESS (A27–A19)												00FFF100	XXXX0000	00000000
DRAM1	RM		SIZE	BASE ADDRESS (A27–A19)												00FFF110	XXXX0000	00000000
DRAM2	RM		SIZE	BASE ADDRESS (A27–A19)												00FFF120	XXXX0000	00000000
DRAM3	RM		SIZE	BASE ADDRESS (A27–A19)												00FFF130	XXXX0000	00000000
DRAM4	RM		SIZE	BASE ADDRESS (A27–A19)												00FFF140	XXXX0000	00000000
DRAM5	RM		SIZE	BASE ADDRESS (A27–A19)												00FFF150	XXXX0000	00000000

**Table C-1. Memory-Mapped Register Set (Continued)**

REG	MEMORY MAP														ADDRESS	VALUE AT RESET		
	15	14	13	12	11	10	9	8	7	6	5	4	3	2		1	0	HIGH BYTE
DRMCR															TS	00FFF160	XXXXXXXX	XXXXXXXX00
	REFRESH INTERVAL COUNT (RIC)															00FFF162	XXXXXXXX00	01000000
PDMA CONFIG. REG.	0000			PDMA DRAM TRANSFER ADDRESS (HIGH WORD)												00FFF200	00000000	00000000
	PDMA DRAM TRANSFER ADDRESS (LOW WORD)															00FFF202	00000000	00000000
	PDMA TRANSFER COUNT															00FFF204	XX000000	00000000
															FR	00FFF206	XXXXXXXXXX	XXXXXXXXX0
GDMA CONFIG. REG.	0000			GDMA DRAM TRANSFER ADDRESS (HIGH WORD)												00FFF210	00000000	00000000
	GDMA DRAM TRANSFER ADDRESS (LOW WORD)															00FFF212	00000000	00000000
	0000			GDMA CHIP-SELECT TRANSFER ADDRESS (HIGH WORD)												00FFF214	00000000	00000000
	GDMA CHIP-SELECT TRANSFER ADDRESS (LOW WORD)															00FFF216	00000000	0000000X
	GDMA TRANSFER COUNT															00FFF218	XX000000	00000000
															FR	00FFF21A	XXXXXXXXXX	XXXXXXXXX0
GDMCR															DM DS W D	00FFF21C	XXXXXXXXXX	XXX00000
PPCR															FLL RLD ABT PDE ERC MODE DFE RST	00FFF300	XXXXXXXXX0	00000000
PPIR				INT	AFD	STR	SIN	ACK 1	BSY 1	ACK 2	BSY 2	PER	SEL	FLT	00FFF302	XXXXX000	01101000	
				CMD		DATA										00FFF304	XXXXXXXXX0	00000000
	ACK PULSE WIDTH (ACKW)															00FFF306	XXXXXXXXXX	00000000
PVCCR															PRT	00FFF400	XXXXXXXXXX	XXXXXXXXX0
	SDN	SLC	SRC	PLE	PLD	VDP	BDP	PRP	FSP	LSP	VCP	VCS	00FFF402	X0000000	00000000			
PCB REG. SET	VERTICAL MARGIN															00FFF404	00000000	00000000
	HORIZONTAL MARGIN															00FFF406	00000000	00000000
	PAGE IMAGE HEIGHT															00FFF408	00000000	00000000
	PAGE IMAGE WIDTH															00FFF40A	00000000	00000000
															SME B2T BND	00FFF40C	XXXXXXXXXX	XXXXXXXX00
	0	PAGE IMAGE BIT ADDRESS (HIGH WORD)														00FFF410	00000000	00000000
	PAGE IMAGE BIT ADDRESS (LOW WORD)															00FFF412	00000000	00000000
PCOMR	PRINTER COMMAND															00FFF500	XXXXXXXXXX	00000000
	PRINTER STATUS															00FFF502	XXXXXXXXXX	00000000
	CCLK DIVISOR										CSB SRC CRC	00FFF504	X0000000	00000000				
TIMER REG.	TIMER INTERVAL (HIGH BYTE)															00FFF600	XXXXXXXXXX	00000000
	TIMER INTERVAL (LOW WORD)															00FFF602	00000000	00000000
	TIMER COUNT (HIGH BYTE)															00FFF604	XXXXXXXXXX	00000000
	TIMER COUNT (LOW WORD)															00FFF606	00000000	00000000
PVCIR	ENABLE															00FFF700	XXXXXXXXXX	XXX00000
								BSY	PFL	PBB	PGE	BUD	VUD	ILA	00FFF702	XXXXXXXXXX	X0000000	
	INT. LEVEL															00FFF704	XXXXXXXXXX	XXXXX000
RIER	ENABLE															00FFF710	XXXXXXXXXX	XXXXXX00
												RBY	DLF	RDN	RER	00FFF712	XXXXXXXXXX	XXXX0000
	INT. LEVEL															00FFF714	XXXXXXXXXX	XXXXX000
PCIER	ENABLE															00FFF720	XXXXXXXXXX	XXXXXX00
	CMS STR															00FFF722	XXXXXXXXXX	XXXXXX00
	INT. LEVEL															00FFF724	XXXXXXXXXX	XXXXX000
TIMER INT. REG.															ENA	00FFF730	XXXXXXXXXX	XXXXXXXXX0
															INT	00FFF732	XXXXXXXXXX	XXXXXXXXX0
															INT. LEVEL	00FFF734	XXXXXXXXXX	XXXXX000

Table C-1. Memory-Mapped Register Set (Continued)

REG	MEMORY MAP														ADDRESS	VALUE AT RESET	
	15	14	13	12	11	10	9	8	7	6	5	4	3	2		1	0
GDMA INT. REG.	ENABLE														00FFF740	XXXXXXXX	XXXXX000
	BSY FLL TCR CMP ILA														00FFF742	XXXXXXXX	XXXX0000
	INT.LEVEL														00FFF744	XXXXXXXX	XXXXX000
PDMA INT. REG.	ENABLE														00FFF750	XXXXXXXX	XXXXX000
	BSY FLL TCR CMP ILA														00FFF752	XXXXXXXX	XXXX0000
	INT.LEVEL														00FFF754	XXXXXXXX	XXXXX000
PIER	ENABLE														00FFF760	XXXXX000	00000000
	IVD CRD DRD IHL INH AFL AFH STL STH SNL SNH														00FFF762	XXXXX000	00000000
	INT.LEVEL														00FFF764	XXXXXXXX	XXXXX000
EXIRO	ENB2 ENB0														00FFF770	XXXXXXXX	XXXXXX00
	STS2 REQ2 STS0 REQ0														00FFF772	XXXXXXXX	XXXX0000
	INT.LEVEL														00FFF774	XXXXXXXX	XXXXX000
	MODE2 MODE0														00FFF776	XXXXXXXX	XXXX0000
	SEN2 SEN0														00FFF778	XXXXXXXX	XXXXXX00
EXIR1	ENB3 ENB1														00FFF780	XXXXXXXX	XXXXXX00
	STS3 REQ3 STS1 REQ1														00FFF782	XXXXXXXX	XXXX0000
	INT.LEVEL														00FFF784	XXXXXXXX	XXXXX000
	MODE3 MODE1														00FFF786	XXXXXXXX	XXXX0000
	SEN3 SEN1														00FFF788	XXXXXXXX	XXXXXX00
SIER	ENABLE														00FFF790	XXXXXXXX0	00000000
	EIA CIA EVENT														00FFF792	XXXXXXXX0	00000000
	SET														00FFF794	XXXXXXXX	X0000000
MSRR	GDR PDR RGP PVC														00FFF7A0	XXXXXXXX	XXXX0000
RSR	0000		HIGHWORD												00FFF800	00000000	00000000
	LOWWORD												0	00FFF802	00000000	00000000	
RDR	0000		HIGHWORD												00FFF804	00000000	00000000
	LOWWORD												0	00FFF806	00000000	00000000	
MMAR	BASE ADDRESS														00FFF900	XXXXXXXX	XXXX0000
322MSK	REV														00FFF902	00000000	00000000
DMASP	SPD														00FFF904	00000000	00000000
TEST REG.	RGPBUSACTIVITY														00FFFA00	00000000	00000000
	DMA IPL MD														00FFFA02	XXXXXXXX	XXXXX000
	PDMA		GDMA		TMR	PCOM		RGP		PVC				00FFFA04	00000000	00000000	
	SFT		EX1	EX0	PPI										00FFFA06	XXX00000	00000000

## C.1 MC68322 MASK REGISTER

The MC68322 mask (322MSK) register reflects the revision number of the MC68322 chip.

## C.2 TEST REGISTER

The test register contains fields that should not be read or written.

### RGP Bus Activity

This field indicates the current bus activity while the RGP is processing a display list.

#### DMA—DMA Test

When this bit is set with WAIT asserted, a DRAM cycle will be delayed. This bit is used for testing the DMA cycles.

#### IPL—IPL Test

When this bit is set, the  $\overline{\text{IPLx}}$  signals output on MA2–MA0. This bit is used with the test registers interrupt set fields (PDMA, GDMA, TMR, PCOM, etc) to test the interrupt logic in the MC68322.

#### MD—Mode Test

When this bit is set with WAIT asserted, the MC68322 goes into test mode.

#### Set Interrupt

The test register contains 10 fields, collectively called the set interrupt fields, which allow the capability to set interrupt events for each module. Writing a 1 to a bit will set the corresponding bit in that module's interrupt event register. Writing a zero to a bit has no effect. These fields are write-only.



# APPENDIX D

## ALTERNATE PIN FUNCTIONS

This section describes the MC68322 alternate pin functions available in Mask Set G59B. In small systems where many of the pins are not used, these pins may be individually programmed as general-purpose inputs and outputs, thereby providing as many as four I/O pins, 12 output pins, and two interrupt/input pins.

### D.1 PINS

The following 18 pins have alternate functions. Each of these pins powers up with their normal function. Software written for previous revisions of the MC68322 will operate identically on Mask Set G59B provided users write zeroes to all unused register bits. To alternately define these pins, the software may program the ALTPIN SEL register to assign an alternate function to each pin. The following table shows the pins and pin type when the ALTPIN SEL register bit is set. Note that three of the alternate bus master output signals share a register bit.

**Table D-1. ALTPIN SEL Bit Descriptions**

PIN	PIN TYPE ALTPIN SEL =1	00FFF910 ALTPIN SEL REGISTER BIT	00FFF912 ALTPIN DIR REGISTER BIT	00FFF914 ALTPIN IN REGISTER BIT	00FFF916 ALTPIN OUT REGISTER BIT
A (22)	Input / Output	0	0	0	0
A (23)	Input / Output	1	1	1	1
A(24)	Input / Output	2	2	2	2
A(25)	Input / Output	3	3	3	3
A (21)	Output	4	—	—	4
CS* (4)	Output	5	—	—	5
CS* (5)	Output	6	—	—	6
CS* (6)	Output	7	—	—	7
CS* (7)	Output	8	—	—	8
RAS* (4)	Output	9	—	—	9
RAS* (5)	Output	10	—	—	10
WRU*	Output	11	—	—	11
DACK*	Output	12	—	—	12
AS*	Output	13	—	—	13
DTACK*	Output	13	—	—	14
BG*	Output / Buzzer	13	—	—	—
BR*	Input / Interrupt	14	—	—	—
DREQ*	Input / Interrupt	15	—	—	—

## D.2 STATE DURING RESET

The output pins are in the normal deasserted state during reset and the first bus cycles, except for A[21:25],  $\overline{AS}$ , and  $\overline{RAS}$ [4:5]. Pins A[22:25] may be used as inputs on systems that require 4M or less of ROM by using the input pin mode as described below.

## D.3 REGISTERS

The new registers in the table below control the alternate pin function.

REGISTER	ADDRESS	BITS	R/W	DESCRIPTION
ALTPIN SEL	00FFF910	16	R/W	Select Alternate Function For Each Pin.
ALTPIN DIR	00FFF912	4	R/W	Direction Control For A[22:25], 0=In, 1=Out.
ALTPIN IN	00FFF914	4	R	Input Status For A[22:25].
ALTPIN OUT	00FFF916	15	R/W	Output Control For Output Bits.

The following bits have been added to the existing external interrupt registers—DREQ (EXIR2) bits were added to the EXIR0 registers and  $\overline{BR}$  (EXIR3) bits were added to the EXIR1 registers. The new bits are located immediately adjacent to the bits already contained in those registers. For example, the EXIR0 enable bit is bit 0 in the EXIR0 ENB register and the EXIR2 enable bit is bit 1. Similarly, the EXIR0 status and request bits are bit 1 and 0 of the EXIR0 EVENT register, respectively, and the EXIR2 status and request bits are bit 3 and 2, respectively. EXIR0 and EXIR2 will share the same interrupt level as EXIR1 and EXIR3. If  $\overline{BR}$  or DREQ are to be used as simple inputs rather than interrupts, the software should keep the interrupts disabled. Users can check the input pin status by reading the status bit for the corresponding pin in the appropriate external interrupt event register. In a similar way, the IRQ0 and IRQ1 pins have always been available as input pins.

REGISTER	ADDRESS	BITS	R/W	DESCRIPTION
EXIR0 ENB	00FFF770	1	R/W	Interrupt EXIR2 Enable.
EXIR0 EVENT	00FFF772	2	R/W	Interrupt EXIR2 Request And Status.
EXIR0 MODE	00FFF776	2	R/W	Interrupt EXIR2 Mode.
EXIR0 SEN	00FFF778	1	R/W	Interrupt EXIR2 Software Enable Bit.
EXIR1 ENB	00FFF780	1	R/W	Interrupt EXIR3 Enable.
EXIR1 EVENT	00FFF782	2	R/W	Interrupt EXIR3 Request And Status.
EXIR1 MODE	00FFF786	2	R/W	Interrupt EXIR3 Mode.
EXIR1 SEN	00FFF788	1	R/W	Interrupt EXIR3 Software Enable Bit.

The registers in the following table have been added for buzzer control.

REGISTER	ADDRESS	BITS	R/W	DESCRIPTION
BUZZER INT	00FFF608	8	R/W	Buzzer Interval.
BUZZER ENB	00FFF60A	1	R/W	Buzzer Enable.

## D.4 INPUT PIN MODE

For A[22:25] to be programmed as input pins, they must be three-stated on power-up to prevent the processor from driving them during the initial bus cycles, which could cause bus contention on these lines. To accommodate this condition, A[22:25] will power-up in a three-state mode when a special state of the  $\overline{\text{HI-Z}}$  and TEST pins exists. The normal functions for  $\overline{\text{BR}}$  and DREQ will also be disabled in input pin mode (ALTPIN SEL register bits 14,15 are set). The table below shows the modes for these pins.

HI-Z	TEST	FUNCTION
0	0	Three-State All Outputs
0	1	Input Pin Mode, A[22:25] Three-Stated ALTPIN SEL Register Bits 0-3, 14-15 = 1
1	0	Normal Mode, A[22:25] Enabled ALTPIN SEL Register Bits 0-15 = 0
1	1	Special Test Mode (Do Not Use)

## D.5 BUZZER

The  $\overline{\text{BG}}$  output pin now has a buzzer control function, which consists of an 8-bit interval control register, a 1-bit enable register, and a 16-bit free-running counter. The eight bits in the BUZZER INT register control the value to be loaded into the upper byte of the 16-bit counter after it reaches a count of zero. The lower byte will always be loaded with \$FF. The enable bit in the BUZZER ENB register causes the output from the  $\overline{\text{BG}}$  pin to toggle every time the counter reaches zero. The buzzer function drives a buzzer on the board or front panel. On products that need this feature, the buzzer control function saves a 555 chip with its resistors and capacitors.

## D.6 IN-CIRCUIT EMULATION

If the alternate pin function is to be used in a system, then an in-circuit emulator (ICE) cannot be used for system development. This feature of the G59B Mask Set is not available in an ICE version. If the alternate pin function will not be used in the system, then the original ICE version of the MC68322 can be used with an in-circuit emulator.

## D.7 OPERATION EXAMPLE

If the alternate pin functions are not used or if none of them are be used as inputs, then  $\overline{HI-Z}$  should be connected to  $V_{CC}$ , and TEST (pin 159 of the PQFP package) should be connected to ground. If some of the input pins will be used, then the input pin mode should be selected by connecting  $\overline{HI-Z}$  to ground and TEST to  $V_{CC}$ . This connection powers up the A[22:25] pins to function as input pins and disables the normal  $\overline{BR}$  and DREQ input pin functions. The software loads the ALTPIN OUT register with the initial values, loads the ALTPIN DIR register to control the configuration of the A[22:25] pins, and loads the ALTPIN SEL register to control the pin configurations.

# APPENDIX E

## GLOSSARY

This section provides the definitions to some of the key terms used in this document.

### **address demunging**

Reversal of a munge operation on an address. This has the effect of restoring the original address.

### **address munging**

Address modification in a way that the low-order three bits of the address are exclusive-OR'ed with a three-bit value that depends on the length of the operand (refer to the *PowerPC™ Microprocessor Family: The Programming Environments (MPGFPE/AD)*).

### **atomic cycle**

If multiple bus transactions by a bus master occur in a sequence where the master retains ownership of the bus during the duration of the sequence, thus preventing other master(s) from transferring in the middle of the sequence, the sequence is considered atomic.

### **autobaud**

The process of determining a serial data rate by timing the width of a single bit.

### **big-endian**

An ordering of the bytes within a word where the least-significant byte is at the highest address and vice versa. For example, a 32-bit wide data bus with big-endian, the least-significant byte is located on data bus bits 24-31 and the most-significant byte is on bits 0-7.

### **blockage**

The interval from the time an instruction begins execution until its execution unit is available for a subsequent instruction (AND has 1 clock latency and 1 clock blockage).

### **boundedly undefined**

Results of a given (not defined) instruction are boundedly undefined if they could have been achieved by executing an arbitrary sequence of defined instructions in valid form starting in the state the machine was before the attempt was made to execute the given instruction.

**breakpoint**

An event that forces the machine to branch into a breakpoint exception routine.

**bubble**

A number inside a circle that is used to identify specific terms in AC timing diagrams.

**burst**

A bus transfer that has more than one piece of data associated with it.

**burst length**

The number of data associated with a burst cycle. For example, a burst length of four has four data pieces (four beats) associated with it.

**bus park**

Keeps the bus granted to a bus master although it has completed the bus cycle. This allows the same master to make the next transfer without having to re-arbitrate for the bus.

**copyback**

Updates to external memory are delayed until forced by the user program or a transfer of bus control to an external master. At the time of forced update or relinquishment of the bus, all changes to the cache are written to external memory. Until that time, cache and external memory are not coherent.

**critical-data first**

This feature allows the data transferred during the burst cycle to be organized where the word or data needed first is the first one to transfer within the burst-data block. The order of transferring can be sequential and usually wraps back to the word (or data) zero. For example, 1 → 2 → 3 → 0 for a sequence of four data with data 1 as the critical data.

**datastream**

A sequence of information to be processed by the CPU.

**early termination**

Some burst protocols specify the burst length at the beginning of the transfer. Early termination allows the burst to be terminated before all data beats are transferred.

**exception**

An error, unusual condition, or external signal that can set a status bit. It may or may not cause an interrupt, depending on whether or not the corresponding interrupt is enabled.

**execution serialization**

Instruction issue is halted until all instructions that are currently in progress complete execution (all internal pipeline stages and instruction buffers have emptied and all outstanding memory transactions are completed).

**execution stream**

The combination of instructions and data on which the CPU operates.

**fetch serialization**

Instruction fetch is halted until all instructions currently in the processor have completed execution (all issued instructions as well as the prefetched instructions waiting to be issued). The machine after fetch serialization is said to be completely synchronized.

**fixed transaction**

A bus transaction that combines the address and data phase of the bus cycle into a single event.

**flow control instruction**

One of: B BR BCR BCC RFI SC and sometimes ISYNC.

**half-word**

A half-word consists of 2 bytes or 16 bits.

**instruction done**

Execution finished and results written back.

**instruction execution time**

All the time between taken and done.

**instruction fetch**

Reading the instruction data received from the instruction memory (I-Cache, Flash).

**instruction issue**

Driving valid instruction bits inside the core.

**instruction retire**

The instruction and all preceding instructions in the program finished execution with no errors. Retired instructions are said to be architecturally executed.

**instruction stream**

A sequence of operands to be executed by the CPU.

**instruction taken**

All resources to perform the instruction are ready and the core begins executing it.

**internal bus**

The bus connecting the CPU and SIU.

**interrupt**

The act of changing the machine state register and other parts of the machine state in response to an exception.

**latency**

The interval from the time an instruction begins execution until it produces a result that is available for use by a subsequent instruction.

**little-endian**

Byte ordering that assigns the lowest address to the lowest-order 8 bits of the scalar.

**master**

A device on the bus that requests bus ownership and initiates the bus cycles.

**memory controller**

A functional logic section of the . It's primary function is to provide the controls for the external bus memories and I/O devices.

**no operation (NOP)**

An instruction whose sole function is to increment the Program Counter, but which affects no changes to any registers or memory.

**pace control**

Controls the rate of the data flow between the master and slave. The burst mechanisms allow this to be controlled by the slave and is useful in slowing down the data transfer rate. Slave delay can be used in place of pace control. It means the data pace can be slowed down by the slave.

**pipeline**

The act of initiating a bus cycle while another bus cycle is in progress. Thus, the bus can have multiple bus cycles pending at a time.

**scan chain**

The peripheral buffers of a device, linked in JTAG test mode, that are addressed in a shift register fashion.

**scoreboard**

A register tracking system that ensures that values are not pulled from a register before they are updated by a previous instruction.

**sequential instruction**

Any instruction that is not a flow control instruction and not ISYNC.



**slave**

A device that responds to the master's address. A slave receives data on a write cycle and gives data to the master on a read cycle.

**snoop**

The act of monitoring external bus activity by alternate bus masters. By snooping these external accesses, a CPU can identify accesses to memory locations that contain dirty data and possibly halt activity to supply correct data.

**swap**

Four byte lanes, reversing (lane 0 to lane 3, lane 1 to lane 2, lane 2 to lane 1 and lane 3 to lane 0).

**tablewalk**

An index value is used to identify an entry point in a tree structure that is traversed until a pointer is found. The system 'walks' through a table of pointers to it's end.

**transaction**

A bus transaction consists of an address transfer (address phase) and data transfers (data phase).

**time-division multiplex (TDM)**

Any serial channel that is divided into channels separated by time.

**watchpoint**

An event that is reported, but does not change the timing of the machine.

**word**

A word consists of 4 bytes or 32 bits.

**writethrough**

Continuous updates, as they occur, of external memory so that cache and memory maintain coherency at all times.

# INDEX

## Numerics

- 0° pages
  - bottom-to-top (B2T) parameter 13-6
  - scanline run 12-7, 12-8
- 0° pages
  - definition, 1-11
- 180° pages
  - band fault, during 11-4
  - bottom-to-top (B2T) parameter 13-6
  - definition, 1-11
  - scanline run, 12-7, 12-8
- 322MSK, C-3

## A

- accessing memory, 4-9
- address
  - bit, defined 13-5
  - byte, defined 13-5
- address boundary, DRAM registers, 7-2
- address bus, 2-3, 4-1
- address constraints, 12-15
- address error exception 5-12
- address error, 5-12
- address, physical vs logical, 13-6
- addresses
  - physical/logical translation, 13-48
- addresses, duplex, 13-6
- addresses, graphic orders, 13-5
- addressing modes, 3-3
- ALTPIN DIR, D-2
- ALTPIN IN, D-2
- ALTPIN OUT, D-2
- ALTPIN SEL, D-1
- applications
  - configuring a generic print engine interface, B-11
  - configuring the DRAM and DRAM SIMM, B-2
  - configuring the flash EPROM, B-4
  - configuring the in-circuit emulation, B-9
  - configuring the parallel port, B-10
  - configuring the random control logic, B-7
  - configuring the serial EEPROM, B-8
  - MC68322 internal registers sample code, B-13
  - memory map initialization example, B-12

- applications, B-1
- asynchronous operation, 10-10

## B

- band
  - bit map type described 12-1
  - buffer, reuse, 10-7
  - display list defined 11-3
  - fault, 11-4
  - number parameter, 13-7
  - underrun, interrupt event, 10-7
- band buffer, 10-9
- band faults, 13-8
- band image
  - starting address 10-6
- band numbers, 13-7
- banded
  - duplex operation, 13-7
- banded bitmap, 12-1
- banding
  - definition, 1-10
- bit address, definition, 13-5
- bit block order execution, 12-14
- bit block transfer
  - expanded operation, 12-5
- bit block transfer graphic orders, 13-3
- bit block transfers, 12-5
- bit map
  - expanded described 12-1
- bit maps
  - expanded
    - clipping 13-3
- bit string specifiers, 12-6
- bitBLT
  - rendering direction, 10-5
- bitBLT, 12-5
- bit-granular, control of clipping 13-3
- bitmap
  - banded, 12-1
  - definition, 1-9
  - expanded, 12-1
  - frame, 12-2
  - halftone, 12-2
  - unbanded, 12-1
  - unexpanded, 12-1
- bitmap types, 12-1
- block diagrams
  - parallel port interface controller, 9-1

BLT2BB\_D, 13-9  
 BLT2BB\_SD, 13-11  
 BLT2BB\_SHD, 13-13  
 BLT2BB\_XD, 13-17  
 BLT2BB\_XHD, 13-22  
 BLT2F\_D, 13-27  
 BLT2F\_SD, 13-28  
 BLT2F\_SHD, 13-29  
 BLT2F\_XD, 13-31  
 BLT2F\_XHD, 13-33  
 BLT2UB\_D, 13-36  
 BLT2UB\_SD, 13-37  
 BLT2UB\_SHD, 13-38  
 BLT2UB\_XD, 13-40  
 BLT2UB\_XHD, 13-43  
 Boolean  
     code calculating example 12-4  
     specifying a graphic operatin transfer, 12-3  
 Boolean codes, 12-3  
 Boolean logic unit, 11-1  
 bottom-to-top (B2T), definition, 13-6  
 BR, asserting, 4-10  
 burst accesses, DRAM, 7-10  
 burst cycles  
     DRAM access 7-1  
 bus 4-1  
     address, 2-3  
     data, 2-3  
 bus arbitration  
     DRAM, 7-9  
     signals, asserting, 4-9  
 bus arbitration, 4-9  
 bus cycle, exception, 5-13  
 bus cycles  
     DRAM read cycles 7-7  
     DRAM write cycles 7-8  
 bus interface unit  
     GDMA read cycles, performing 8-7  
 bus mastership, exchanging, 4-9  
 bus operation  
     core read cycle, 4-1  
     core write cycle, 4-4  
     external bus master, 4-9  
     interrupt acknowledge bus cycle, 4-6  
     reset, 4-8  
 bus operation, 4-1  
 BUZZER ENB, D-3  
 BUZZER INT, D-3  
 byte address, defined 13-5  
 byte operation, 4-1, 4-4

## C

CCLK supplied by MC68322, 10-11, 10-13  
 CCLK supplied by print engine, 10-12, 10-14  
 channel address  
     command received interrupt 9-7  
 chip-select  
     recovery value, 6-4  
 chip-select DMA timing register, 6-3  
 chip-selects  
     active read and write times 6-2  
     banks  
         DMA access timing, 8-8  
         location priority 6-3  
     banks, chip-select registers 6-1  
     data transfers  
         synchronous timing values, 6-4  
     minimum value timings, 6-2  
     registers  
         chip-select DMA timing register 6-3  
         chip-select recovery, 6-4  
         DMA access timing 8-8  
         location, 6-3  
     registers at reset, 6-3  
     size encodings, 6-2  
     timing characteristics, 6-2  
 clipping expanded bit maps 13-3  
 clocks  
     command, supplying 10-2  
     status, supplying, 10-2  
     video, divisor operation, 10-15  
 command byte detection, 9-9  
 command bytes  
     during ECP mode 9-3  
 commands to the print engine, 10-11  
 communications modes, 9-1  
 compatibility mode (see handshaking)  
 core  
     data types and addressing modes, 3-3  
     DRAM write cycle, 7-9  
     DRAMcontroller  
         accesses and refresh cycle, 7-6  
     instruction set summary, 3-4, 3-6  
     notational conventions, 3-4  
     programming model, 3-1  
 core read cycle, 4-1  
 core write cycle, 4-4  
 core, 3-1  
 CSDTR, 6-3  
 CSR, 6-1  
 CSR, (see chip-selects, registers)  
 CSRR, 6-4

**D**

- DA (destination address)
  - defined 13-3
- data bus (D15–D0)
  - DMA transfers, during 8-6
- data bus, 2-3, 4-1
- data formats, 3-3
- data latch
  - DMA
    - during DRAM transfers 8-6
    - status indication 8-5
- data latching
  - parallel port 9-13
- data transfer rate, 9-14
- data transfers(see also print engine video controller (PVC))
  - DMA termination, indication of 8-5
- data turnaround time, 4-10
- DDL, 1-8
- default interface, 10-16
- definitions, E-1
- destination
  - operand type described 12-2
- destination address
  - duplex operation, 13-6
- digital filtering, 9-11
- direct memory access (DMA)
  - DMA accesses 7-1
- display list
  - banded defined, 11-3
  - errors during execution 11-4
  - example format 13-5
- display lists
  - address convention, 12-15
- DMA
  - active channel indication 8-5
  - arbitration 8-8
  - chip-select bank
    - access timing 8-8
  - DRAM bus transfers 8-8
  - error condition 8-6
  - external device request 8-4
  - flush request, described 8-3
  - flush request during operation 8-6
  - GDMA
    - read cycle, termination 8-7
  - GDMA configuration registers, described 8-2
  - GDMA write cycle 8-7
  - illegal address interrupt 8-6
  - MC68322 bus cycles, CSx during 8-4
  - PDMA
    - during compatibility mode 9-8
  - PDMA configuration registers, described 8-2
  - soft-reset register 5-14
  - transfers
    - DMA initiated 8-6
    - DREQ and DACK during 8-7
    - size 8-6
- DMA interface
  - channel status indication, 8-5
  - channels, 8-1
  - data latch
    - DRAM transfers, 8-6
  - data latch status indication, 8-5
  - DRAM bus, control, 8-8
  - error condition, 8-10
  - GDMA
    - CSx during read cycle, 8-7
    - handshaking, 8-7
    - read cycle request, 8-7
  - initiating an operation, 8-6
  - invalid address, accessing, 8-6
  - MC68322 address bus, incrementing, 8-2
  - operation, 8-1
  - reallocating resource, 8-6
  - transfer count field, 8-3
  - transfers
    - count, 8-3
    - direction, programmed, 8-4
    - flush request, 8-3
    - termination indication, 8-5
    - width, programmed, 8-4
- DMA interface signals, 2-8
- DMA interface, 8-1
- DMASP, 8-4
- DRAM
  - bank
    - size 7-1
  - banks
    - described 7-1
    - location 7-1
    - reset values 7-10
  - burst access
    - page boundary, crossing 7-10
  - burst accesses 7-10
  - burst cycles 7-1
  - bus
    - DMA control of 8-8
  - bus transfers
    - DMA 8-8
  - devices
    - operation speed 7-5
    - pre-charge 7-10
  - DMA accesses 7-1
  - DRAM control
    - TS field encodings 7-5
  - EC000 Core accesses 7-6, 7-9
  - fast-page mode 7-1
  - fast-page mode, burst accesses 7-10
  - nibble mode 7-1

- registers
  - described 7-1
  - DRAM control 7-5
- reset 7-10
- static column 7-1
- transfers
  - DMA data latch during 8-6
  - WE, RAS, and CAS during DRAM read 7-7
- DRAM bus
  - (see memory data and address bus)
- DRAM controller
  - accesses
    - burst, 7-10
    - bus arbitration, 7-9
  - accesses, 7-6
  - banks
    - location, 7-2
  - devices, 7-1
  - power-up, 7-10
  - read cycle, 7-7
  - registers
    - reset values, 7-10
  - registers and banks
    - base address and size fields, 7-1
    - ROM mode, 7-2
  - registers and banks, 7-1
  - timing mode, 7-5
  - timing modes, 7-5
  - write cycles, 7-8
- DRAM controller, 7-1
- DRAM interface signals, 2-7
- DRMCR, 7-5
- duplex operation
  - bottom-to-top (B2T) parameter 13-6
  - destination address (DA) during 13-6
- duplex printing
  - explanation, 1-11

## E

- EC000 Core
  - DMA accesses 7-1
  - exceptions
    - address error 5-12
    - illegal 5-11
    - privileged violations 5-11
    - tracing 5-12
    - unimplemented 5-11
  - exceptions (see also exceptions)
  - flush request, DMA 8-3
  - illegal memory address access 5-3
    - status register during exception processing 5-7
- electrical characteristics, 14-1
- enhanced capabilities mode, 9-1
- error condition

- display list execution, during 11-4
- DMA transfers 8-10
  - RGP operation, during 11-3
- error condition, DMA during 8-6
- error cycles, 9-12
- exception handling, 5-1
- exception processing
  - stack frame during 5-8
  - status register during 5-7
- exception vector number 5-7
- exceptions
  - address error, 5-12
  - bus cycles, 5-13
  - causes, 5-11
  - how processing occurs, 5-6
  - illegal 5-11
  - illegal and unimplemented instructions, 5-10
  - multiple 5-13
  - multiple, 5-13
  - operation, 5-6
  - priority 5-13
  - privilege violations, 5-11
  - processing-specific, 5-9
  - tracing, 5-12
  - types
    - instruction traps, 5-10
    - interrupt, 5-10
    - reset, 5-9
  - types, 5-9
  - unimplemented 5-11
- exceptions, 5-6
- EXIR0 ENB, D-2
- EXIR0 EVENT, D-2
- EXIR0 MODE, D-2
- EXIR0 SEN, D-2
- EXIR1 ENB, D-2
- EXIR1 EVENT, D-2
- EXIR1 MODE, D-2
- EXIR1 SEN, D-2
- expanded bit block graphic orders, 13-3
- expanded bitmap, 12-1
- expanded bitmaps
  - clipping, 13-3
- external bus master
  - illegal memory address access 5-3
- external bus master read cycle, 4-10
- external bus master signals, 2-6
- external bus master write cycle, 4-11
- external bus master, 4-9

## F

- fast-page mode, 7-1
- features
  - core, 3-1
  - MC68322, 1-2
- FH (frame height), defined 13-4
- flush request
  - DMA, during operation, 8-6
- flush request, DMA 8-3
- frame bit map described 12-2
- frame bitmap, 12-2
- FW (frame width), defined 13-3

## G

- GDMA (general purpose DMA), (see DMA)
- GDMA, 8-2
- GDMCR, 8-3
- glossary, E-1
- graphic operations
  - bit block and scanline order execution, 12-14
  - bit block transfer, 12-5
  - bitmap types, 12-1
  - Boolean codes, 12-3
  - graphic operands
    - types, 12-3
  - graphic operands, 12-2
  - location and address constraints, 12-15
  - scanline and halftone table example, 12-13
  - scanline transfer
    - executing during banded applications, 12-9
    - halftone companion tables, 12-10
    - run operation, 12-8
  - scanline transfer, 12-5
- graphic operations, 12-1
- graphic order
  - interrupted 11-4
- graphic orders
  - address alignment, 13-5
  - addresses, 13-5
  - band number and fault, 13-7
  - bitBLT during duplex 13-6
  - defined 11-1
  - descriptions, 13-8
  - display list sequence, 13-5
  - duplex operation 13-6
  - example display list format 13-5
  - execution unit, 11-1
  - operand types, 12-2
  - parser, 11-1
  - scanline during duplex, 13-6
  - types
    - bit block transfer, 13-3
    - expanded bit block transfer, 13-3

- initialization, 13-1
- program flow control, 13-3
- scanline transfer, 13-4
- types, 13-1
- graphic orders, 13-1
- graphics unit
  - shut down 12-15

## H

- HA (halftone address), defined 13-3
- halftone
  - 32-bit specifier, 12-10
  - 48-bit specifier described 12-11
  - bit map type described 12-2
  - operand type described 12-2
  - specifier boundary conditions 12-12
  - table
    - address convention 12-15
    - table address (HTTA) during duplex operation 13-6
- halftone bitmap, 12-2
- halftone companion table, 12-10
- halftoning
  - definition, 1-10
- handshaking
  - compatibility
    - error cycles, 9-12
- ECP
  - enabling with RLE, 9-9
  - enabling without RLE 9-9
  - operation 9-9
  - with RLE, 9-10
  - without RLE, 9-10
- hardware, disabling, 9-10
- resetting controller, 9-14
- software-controlled, 9-11
- handshaking modes
  - compatibility
    - enabling, 9-8
    - operation 9-8
    - PDMA during 9-8
- ECP
  - command bytes during, 9-3
  - types, 9-8
- hardware handshaking, 9-7
- horizontal margin (see page, margins)
- HTTA (see halftone table address)
- HXR (halftone X remainder), defined 13-3
- HYR (halftone Y remainder), defined 13-4

**I**

- ICE interface signals, A-1
- ICE, A-1
- illegal address
  - DMA access to 8-6
- illegal address interrupt, 4-12
- illegal and unimplemented instructions, 5-10
- illegal instruction exception 5-11
- in-circuit emulation interface
  - adaptor board design, A-4
  - adaptor board schematics, A-6
  - pin assignment, A-16
  - signals, A-1
- in-circuit emulation interface, A-1
- initialization graphic orders, 13-1
- instruction set summary, 3-4, 3-6
- instructions
  - illegal and unimplemented, 5-11
  - privileged listed 5-11
  - privileged, 5-11
  - tracing 5-12
  - unimplemented emulation 5-11
- interrupt acknowledge bus cycle, 4-6
- interrupt events
  - command sent, 10-11
  - RGP error, 12-15
- interrupt handling, 5-1
- interrupt, generating, 4-6
- interrupts
  - DTACK, during illegal memory address access 5-3
  - error during RGP operation, 11-3
  - events
    - band underrun 10-7
    - command receive, 9-7
    - command sent status, 10-8
    - data received, 9-7
    - DMA illegal address 8-6
    - EC000 Core request 9-8
    - illegal address, PVC, 10-7
    - page end 10-7
    - page/band begin, 10-7
    - PDMA request, 9-8
    - RGP busy, 11-3
    - RGP error 11-4
    - status receive, 10-8
    - video underrun 10-7
  - external, 5-4
  - externally initiated 5-4
  - hardware, 5-1
  - illegal memory address access, 5-3
  - internal, 5-1
  - IRQx, during external interrupts 5-4
  - priority level
    - external, 5-4

- registers
  - external described 5-4
  - software event described 5-3
- software
  - clearing, 5-3
  - setting priority level, 5-3
- software, 5-3
- interrupts, 5-1

**J**

- JUMP, 13-46

**L**

- languages
  - printer, 1-8
- latching, 9-13
- location constraints, 12-15

**M**

- mask register, C-3
- mastership, assuming, 4-9
- MC68322
  - alternate pin functions, D-1
  - applications, B-1
  - bus arbitration, 4-9
  - bus operation, 4-1
  - configuration, B-1
  - core, 3-1
  - DMA interface, 8-1
  - DRAM controller, 7-1
  - electrical and thermal characteristics, 14-1
  - explanation, 1-8
  - features, 1-2
  - graphic operations, 12-1
  - graphic orders, 13-1
  - in-circuit emulation interface, A-1
  - interrupt and exception handling, 5-1
  - introduction, 1-1
  - memory-mapped register summary, C-1
  - parallel port interface, 9-1
  - print engine interface, 10-1
  - RISC graphics processor, 11-1
  - signal descriptions, 2-1
  - system integration module, 6-1
- MC68322 bus
  - cycles
    - CSx during DMA generated, 8-4
    - DMA incrementing address 8-2
    - DMA read and write cycles 8-7
- MC68322 reset, differences during, 10-16
- mechanicals, 15-1

memory addresses, 3-3  
 memory data bus  
   DRAM write cycles 7-8  
 memory-mapped registers, C-1  
 modes of operation  
   exception processing described 5-6  
 module soft-reset register (MSRR), described 5-14  
 module soft-reset register, 5-14  
 MSRR, 5-14  
 multiple exceptions, 5-13

## N

nibble mode, 7-1  
 notational conventions, 3-4

## O

operands  
   types of graphic order 12-2  
 operation, bus, 4-1  
 ordering information, 15-1

## P

page  
   buffer reuse 10-7  
   height of image, 10-5  
   image  
     dimension and location 10-9  
     height values counted down 10-9  
     width values counted down, 10-9  
   width 10-5  
 page boundary  
   DRAM burst access 7-10  
 parallel interface port signals, 2-9  
 parallel port  
   control signals 9-8  
   digital filtering, operation 9-11  
   error cycles (see error cycles)  
   hardware handshaking (see handshaking)  
 parallel port interface  
   block diagram, 9-1  
   data latching operation, 9-13  
   data transfer rate, 9-14  
   digital filtering, 9-11  
   error cycles, 9-12  
   hardware handshaking  
     disabling, 9-10  
     ECP, 9-8  
   hardware handshaking, 9-7  
   interrupt events, 9-6  
   PDMA, 9-8  
   PDMA, (see also DMA)

registers, 9-2  
 RESET, 9-14  
   software controlled handshaking, 9-11  
 parallel port interface, 9-1  
 PCB, 10-5  
 PCIER, 10-8  
 PCL, 1-8  
 PCOMR, 10-2  
 PDL, 1-8  
 PDMA (parallel port DMA), (see DMA)  
 PDMA, 8-2  
 performance, improving, 4-9  
 phase 10-1  
 phase lock loop  
   control of video clock, 10-3  
 phase-locked loop, 10-1  
 PIER, 9-6  
 pins (alternate)  
   functions, D-1  
 pixel, definition, 12-1  
 PLL video clock divisor, 10-15  
 PLL, 10-1  
 PPCR, 9-4  
 PPIR, 9-2  
 prescaler, setting resolution 10-15  
 print engine  
   band image starting address, 10-6  
   clocks  
     supplying command and status 10-2  
   command operation  
     end of operation 10-11  
   interface  
     operation, described 10-9  
     reset operation 10-11  
     synchronous operation 10-10  
   transmitting data, 10-2  
   video clock divisor operation 10-15  
   video rate supplied 10-2  
 print engine interface  
   operation  
     command, 10-11  
     PLL video clock divisor, 10-15  
     status, 10-13  
   operation, 10-9  
 printer communication protocol, 10-8  
 PVC on reset, 10-15  
 registers  
   printer communication interrupt event, 10-8  
   printer communication, 10-2  
   printer control block register set, 10-5  
   PVC control, 10-3  
   PVC interrupt event, 10-6  
 registers, 10-2  
 synchronous/asynchronous PVC  
   operation, 10-10



print engine interface, 10-1  
 print engine video controller  
     data transfer complete, 10-9  
     soft-reset, 10-16  
 print engine video controller (PVC)  
     soft-reset 10-7  
     starting 10-9  
 print engine video controller interface signals, 2-8  
 print engine video controller, 10-1  
 printer communication interface signals, 2-8  
 printer communication protocol, 10-8  
 printer control block  
     new print operation indication, 10-6  
 printer languages, 1-8  
 printer video controller  
     soft-reset register 5-14  
 printer video controller (PVC)  
     burst cycles 7-1  
     DMA accesses 7-1  
 privilege violation exception 5-11  
 privileged instructions listed 5-11  
 program flow control graphic orders, 13-3  
 PVC reset interrupt event, 10-15  
 PVC, 10-1  
 PVCCR, 10-3  
 PVCIR, 10-6

## R

RDR, 11-2  
 reallocating DMA resource 8-6  
 refresh cycle  
     CAS before RAS, timing 7-5  
     EC000 Core accesses 7-6  
     timing 7-6  
     timing parameters, 7-5  
 refresh cycle, 7-6  
 refresh cycles  
     WE during 7-6  
 registers  
     alternate pin, D-2  
     chip-select DMA timing, 6-3  
     chip-select recovery, 6-4  
     chip-select related, 6-1  
     chip-select, 6-1  
     DMA speed, 8-4  
     DRAM (see DRAM registers)  
     DRAM control, 7-5  
     DRAM, 7-1  
     external interrupt, 5-4  
     external interrupt, described 5-4  
     GDMA configuration  
         described 8-2  
     GDMA configuration, 8-2  
     GDMA control, 8-3

internal status, 5-1  
 interrupt event, 4-6, 5-1  
 interrupt level, 5-1  
 location overlap priority 6-3  
 mask, C-3  
 memory-mapped, C-1  
 module soft-reset (MSRR), described 5-14  
 parallel port control, 9-4  
 parallel port interface, 9-2  
 PDMA configuration, 8-2  
 PPI interrupt event, 9-6  
 printer communication interrupt event, 10-8  
 printer communication, 10-2  
 printer control block, 10-5  
 PVC control, 10-3  
 PVC interrupt event, 10-6  
 RGP diagnostic, 11-2  
 RGP interrupt event, 11-2  
 RGP start, 11-2  
 software interrupt event  
     described 5-3  
 software interrupt event, 4-12, 5-3  
 status (see EC000 Core)  
 status, 4-8, 5-7  
 test, C-3  
 timer interrupt event, 5-5  
 timer, 5-5  
 rendering direction, programmed 10-5  
 RESET  
     asserting, 9-14  
 reset exception, 5-9  
 reset instruction  
     executing, 4-8  
 reset operation  
     PVC during, 10-11  
     PVC, 10-15  
 reset operation, 4-8  
 resolution 10-15  
 resolution, 10-1  
 RGP, 10-9, 11-1  
 RICSC graphics processor (RGP)  
     burst cycles 7-1  
 RIER, 11-2  
 RISC graphics processor  
     DMA accesses, 7-1  
     errors during display list execution, 11-4  
     operation, 11-3  
     registers  
         RGP diagnostic, 11-2  
         RGP interrupt event, 11-2  
         RGP start, 11-2  
     registers, 11-2  
     soft-reset register 5-14  
 RISC graphics processor, 11-1  
 ROM mode, 7-2

RSR, 11-2  
 run-length  
   decompression  
     resetting logic 9-14  
 decompression, 9-10  
 run-length encoding  
   command received interrupt 9-7

## S

sample code, B-13  
 scanline  
   table  
     address convention 12-15  
     table address (STLA) during duplex operation 13-6  
 scanline and halftone table example, 12-13  
 scanline graphic orders, 13-4  
 scanline order execution, 12-14  
 scanline table, 12-6  
 scanline, definition, 12-1  
 scource  
   operand type described 12-2  
 SET\_BBMAP, 13-47  
 SET\_BOOL\_D, 13-50  
 SET\_HTBMAP, 13-51  
 SET\_SBMAP, 13-53  
 SET\_UBMAP, 13-54  
 signal descriptions, 2-1  
 signal summary, 2-2  
 signals  
   ACK\*  
     during compatibility mode transfer 9-8  
     software control of 9-4  
   AUTOFD\*  
     during compatibility mode transfer 9-8  
   bus arbitration, 4-9  
   bus control, 4-1  
   BUSY  
     software control of 9-4  
   CAS  
     DRAM read cycles 7-7  
   CASx  
     DRAM burst accesses 7-10  
   CMD/STS\*  
     direction of 10-2  
   CSx  
     DMA access timing 8-8  
     GDMA read cycle, during 8-7  
   DACK  
     DMA access timing 8-8  
     GDMA handshaking 8-7  
     GDMA read cycle 8-7  
   DMA interface, 2-8  
   DRAM interface, 2-7

DREQ  
   default configuration 8-7  
   GDMA handshaking 8-7  
 DTACK  
   asynchronous EC000 Core timing 6-5  
   illegal memory address access 5-3  
 external bus master, 2-6  
 external interrupts, 5-4  
 FAULT\*  
   driving high level on 9-4  
 FSYNC\*  
   page/band begin interrupt, during 10-7  
 LDS (lower data strobe) 7-8  
 parallel port interface, 2-9  
 print engine video controller interface, 2-8  
 printer communication interface, 2-8  
 RAS  
   DRAM read cycles 7-7  
 SELECT\*  
   driving high level on 9-3  
 STROBE\*  
   during compatibility mode transfer 9-8  
   hardware handshaking disabled 9-10  
 system interface, 2-4  
 WAIT  
   asynchronous EC000 Core timing 6-5  
 WE  
   DRAM read cycles 7-7  
   DRAM write cycles 7-8  
   during refresh cycles 7-6  
 WRL  
   DMA transfers, during 8-6  
   GDMA write cycle 8-7  
 WRU  
   DMA transfers, during 8-6  
   GDMA write cycle 8-7  
 SIM, 6-1  
 SL2BB\_D, 13-55  
 SL2BB\_HD, 13-57  
 SL2F\_D, 13-60  
 SL2F\_HD, 13-61  
 SL2UB\_D, 13-63  
 SL2UB\_HD, 13-64  
 SLTA (see scanline table address)  
 SmartToner  
   enabled, 10-5  
 soft-reset  
   PVC 10-7  
   RGP 11-4  
 specifications, 14-1  
 SR, 5-7  
 stack frame  
   exception 5-8, 5-11  
 static column, 7-1  
 STOP, 13-66

supplying valid data, 4-11  
 system integration module, 6-1  
 system interface signals, 2-4

## T

termination  
     bad address, 8-10  
     core-forced, 8-10  
     normal, 8-9  
 terminology, E-1  
 test register, C-3  
 thermal characteristics, 14-1  
 TIER, 5-5  
 timer  
     count, 5-6  
     interval 5-6  
 timer module, 5-5  
 tracing 5-12  
 tracing, 5-12  
 transfer  
     bit block, 12-5  
     scanline, 12-5  
 transfer count field, DMA described 8-3  
 transfer, initiating, 4-4  
 transferring data, 4-1

## U

unbanded  
     duplex operation during 13-7  
 unbanded bit map described 12-1  
 unbanded bitmap, 12-1  
 unexpanded bit map described 12-1  
 unexpanded bitmap, 12-1  
 unimplemented instruction exception 5-11  
 unimplemented instruction emulation 5-11

## V

vector  
     numbers, listed 5-8  
     table 5-8  
 vertical margin, 10-5  
 video  
     underrun, interrupt event, 10-7

## W

word operation, 4-4  
 word-sized read cycle support, 4-10  
 word-sized write cycle, 4-11

## X

X dimension  
     definition, 1-9  
 X dimension, definition, 12-1  
 XOFF, defined 13-3  
 XON, definition, 13-4

## Y

Y dimension  
     definition, 1-9  
 Y dimension, definition, 12-1

# MC68340

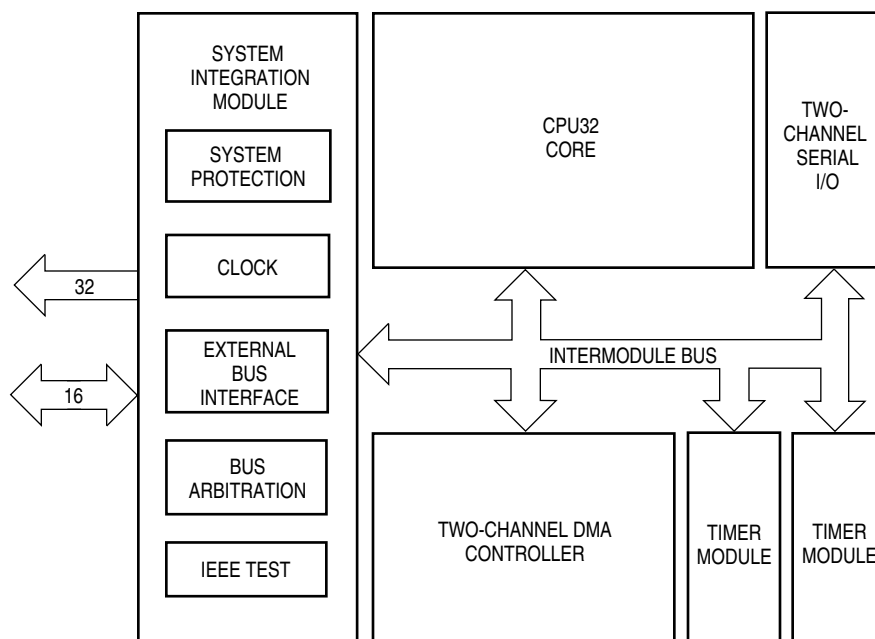
# MC68340V

## Product Brief

### Integrated Processor With DMA

The MC68340 is a high-performance 32-bit integrated processor with direct memory access (DMA), combining an enhanced M68000-compatible processor, 32-bit DMA, and other peripheral subsystems on a single integrated circuit. The MC68340 CPU32 delivers 32-bit CISC processor performance from a lower cost 16-bit memory system. The combination of peripherals offered in the MC68340 can be found in a diverse range of microprocessor-based systems, including embedded control and general computing. Systems requiring very high-speed block transfers of data can especially benefit from the MC68340's DMA.

The MC68340's high level of functional integration results in significant reductions in component count, power consumption, board space, and cost while yielding much higher system reliability and shorter design time. The 3.3-V MC68340V is particularly attractive to applications requiring a very tight power budget. Complete code compatibility with the MC68000 affords the designer access to a broad base of established real-time kernels, operating systems, languages, applications, and development tools—many oriented towards embedded control. Figure 1 shows a block diagram of the MC68340.



**Figure 1. MC68340 Simplified Block Diagram**

This document contains information on a product under development. Motorola reserves the right to change or discontinue this product without notice.



The primary features of the MC68340 are as follows:

- High Functional Integration on a Single Piece of Silicon
- CPU32—MC68020-Derived 32-Bit Central Processor Unit
  - Upward Object-Code Compatible with MC68000 and MC68010
  - Additional 32-Bit MC68020 Instructions and Addressing Modes
  - Unique Embedded Control Instructions
  - Fast Two-Clock Register Instructions—10,045 Dhrystones/Second
- Two-Channel Low-Latency DMA Controller for High-Speed Memory Transfers
  - Single- or Dual-Address Transfers
  - 32-Bit Addresses and Counters
  - 8-, 16-, and 32-Bit Data Transfers
  - 50 Mbyte/Sec Sustained Transfers (12.5 Mbyte/Sec Memory-to-Memory)
- Two-Channel Universal Synchronous/Asynchronous Receiver/Transmitter (USART)
  - Baud Rate Generators
  - Modem Control
  - MC68681/MC2681 Compatible
  - 9.8 Mbits/Sec Maximum Transfer Rate
- Two Independent Counter/Timers
  - 16-Bit Counter
  - Up to 8-Bit Prescaler
  - Multimode Operation
  - 80-ns Resolution
- System Integration Module Incorporates Many Functions Typically Relegated to External PALs, TTL, and ASIC, such as:
  - System Configuration
  - System Protection
  - Chip Select and Wait State Generation
  - Clock Generation
  - Dynamic Bus Sizing
  - Up to 16 Discrete I/O Lines
  - External Bus Interface
  - Periodic Interrupt Timer
  - Interrupt Response
  - Bus Arbitration
  - IEEE 1149.1 Boundary Scan (JTAG)
  - Power-On Reset
- 32 Address Lines, 16 Data Lines
- Power Consumption Control
  - Static HCMOS Technology Reduces Power in Normal Operation
  - Low Voltage Operation at 3.3 V  $\pm$ 0.3 V (MC68340V only)
  - Programmable Clock Generator Throttles Frequency
  - Unused Peripherals Can Be Turned Off
  - LPSTOP Provides an Idle State for Lowest Standby Current
- 0–16.78 MHz or 0–25.16 MHz Operation
- 144-Pin Ceramic Quad Flat Pack (CQFP) or 145-Pin Plastic Pin Grid Array (PGA)

As a low voltage part, the MC68340V can operate with a 3.3-V power supply. MC68340 is used throughout this document to refer to both the low voltage and standard 5-V parts since both are functionally equivalent.

# M68300 FAMILY

The MC68340 is one of a series of components in Motorola's M68300 Family. Other members of the family include the MC68302, MC68330, MC68331, MC68332, and MC68F333.

## ORGANIZATION

The M68300 family of integrated processors and controllers is built on an M68000 core processor, an on-chip bus, and a selection of intelligent peripherals appropriate for a set of applications. The CPU32 is a powerful central processor with nearly the performance of the MC68020. A system integration module incorporates the external bus interface and many of the smaller circuits that typically surround a microprocessor for address decoding, wait-state insertion, interrupt prioritization, clock generation, arbitration, watchdog timing, and power-on reset timing.

Each member of the M68300 family is distinguished by its selection of peripherals. Peripherals are chosen to address specific applications but are often useful in a wide variety of applications. The peripherals may be highly sophisticated timing or protocol engines that have their own processors, or they may be more traditional peripheral functions, such as UARTs and timers. Since each major function is designed in a standalone module, each module might be found in many different M68300 family parts. Driver software written for a module on one M68300 part can be used to run the same module that appears on another part.

## ADVANTAGES

By incorporating so many major features into a single M68300 family chip, a system designer can realize significant savings in design time, power consumption, cost, board space, pin count, and programming. The equivalent functionality can easily require 20 separate components. Each component might have 16–64 pins, totaling over 350 connections. Most of these connections require interconnects or are duplications. Each connection is a candidate for a bad solder joint or misrouted trace. Each component is another part to qualify, purchase, inventory, and maintain. Each component requires a share of the printed circuit board. Each component draws power—often to drive large buffers to get the signal to another chip. The cumulative power consumption of all the components must be available from the power supply. The signals between the CPU and a peripheral might not be compatible nor run from the same clock, requiring time delays or other special design considerations.

In a M68300 family component, the major functions and glue logic are all properly connected internally, timed with the same clock, fully tested, and uniformly documented. Power consumption stays well under a watt, and a special standby mode drops current well under a milliamp during idle periods. Only essential signals are brought out to pins. The primary package is the surface-mount quad flat pack for the smallest possible footprint; pin grid arrays are also available.

## MC68340 SIGNALS

Figure 2 shows the components and signals.

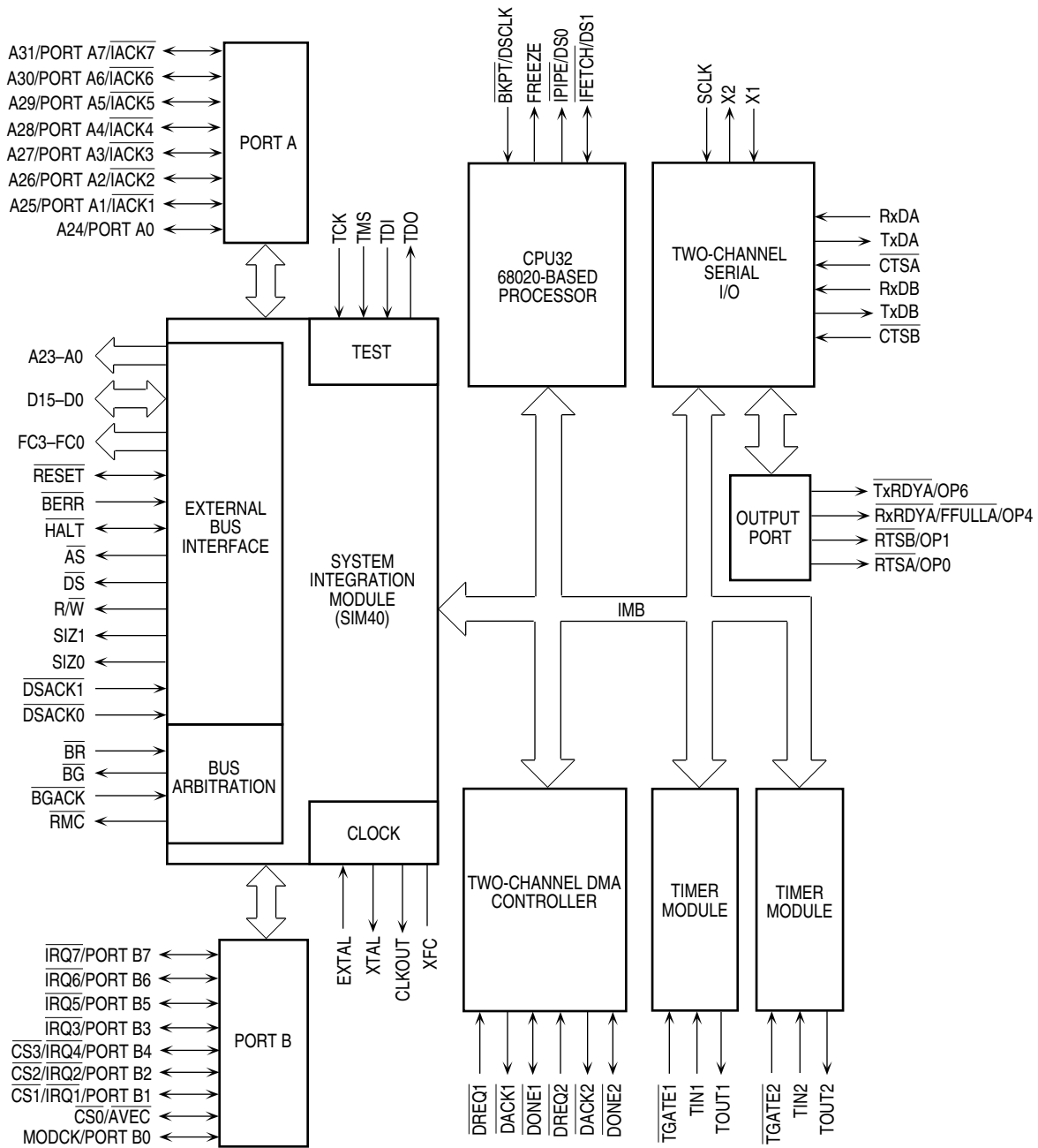


Figure 2. MC68340 Detailed Block Diagram

# CENTRAL PROCESSOR UNIT

The CPU32 is a powerful central processor that supervises system functions, makes decisions, manipulates data, and directs I/O. A special debugging mode simplifies processor emulation during system debug.

## CPU32

The CPU32 is an M68000 family processor specially designed for use as a 32-bit core processor and for operation over the intermodule bus (IMB). Designers used the MC68020 as a model and included advances of the later M68000 family processors, resulting in an instruction execution performance of 4 MIPS (VAX-equivalent) at 25.16 MHz.

The powerful and flexible M68000 architecture is the basis of the CPU32. MC68000 (including the MC68HC000 and the MC68EC000) and MC68010 user programs will run unmodified on the CPU32. The programmer can use any of the eight 32-bit data registers for fast manipulation of data and any of the eight 32-bit address registers for indexing data in memory. The CPU32 can operate on data types of single bits, binary-coded decimal (BCD) digits, and 8, 16, and 32 bits. Peripherals and data in memory can reside anywhere in the 4-Gbyte linear address space. A supervisor operating mode protects system-level resources from the more restricted user mode, allowing a true virtual environment to be developed.

Flexible instructions for data movement, arithmetic functions, logical operations, shifts and rotates, bit set and clear, conditional and unconditional program branches, and overall system control are supported, including a fast  $32 \times 32$  multiply and 32-bit conditional branches. Instructions, such as table lookup and interpolate and low power stop, support specific requirements of embedded control applications. Many addressing modes complement these instructions, including predecrement and postincrement, which allow simple stack and queue maintenance and scaled indexed for efficient table accesses. Data types and addressing modes are supported orthogonally by all data operations and with all appropriate addressing modes. Position-independent code is easily written.

The CPU32 is specially optimized to run with the MC68340's 16-bit data bus. Most instructions execute in one-half the number of clocks compared to the original MC68000, yielding an overall 1.6 times the performance of the same-speed MC68000 and measuring 10,045 Dhrystones/sec @ 25.16 MHz (6,742 Dhrystones/sec @ 16.78 MHz).

Like all M68000 family processors, the CPU32 recognizes interrupts of seven different priority levels and allows the peripheral to vector the processor to the desired service routine. Internal trap exceptions ensure proper instruction execution with good addresses and data, allow operating system intervention in special situations, and permit instruction tracing. Hardware signals can either terminate or rerun bad memory accesses before instructions process data incorrectly.

The CPU32 offers the programmer full 32-bit data processing performance with complete M68000 compatibility, yet with more compact code than is available with RISC processors. The CPU32 is identical in all CPU32-based M68300 family products.

## BACKGROUND DEBUG MODE

A special operating mode is available in the CPU32 in which normal instruction execution is suspended while special on-chip microcode performs the functions of a debugger. Commands are received over a dedicated, high-speed, full-duplex serial interface. Commands allow the manual reading or writing of CPU32 registers, reading or writing of external memory locations, and diversion to user-specified patch code. This background debug mode permits a much simpler emulation environment while leaving the processor chip in the target system, running its own debugging operations.



# ON-CHIP PERIPHERALS

To improve total system throughput and reduce part count, board size, and cost of system implementation, the M68300 family integrates on-chip, intelligent peripheral modules and typical glue logic. These functions on the MC68340 include the SIM40, a DMA controller, a serial module, and two timers.

The processor communicates with these modules over the on-chip intermodule bus (IMB). This backbone of the chip is similar to traditional external buses with address, data, clock, interrupt, arbitration, and handshake signals. Because bus masters (like the CPU32 and DMA), peripherals, and the SIM40 are all on the chip, the IMB ensures that communication between these modules is fully synchronized and that arbitration and interrupts can be handled in parallel with data transfers, greatly improving system performance. Internal accesses across the IMB may be monitored from outside of the chip, if desired.

Each module operates independently. No direct connections between peripheral modules are made inside the chip; however, external connections could, for instance, link a serial output to a DMA control line. Modules and their registers are accessed in the memory map of the CPU32 (and DMA) for easy access by general M68000 instructions and are relocatable. Each module may be assigned its own interrupt level, response vector, and arbitration priority. Since each module is a self-contained design and adheres to the IMB interface specifications, the modules may appear on other M68300 family products, retaining the investment in the software drivers for the module.

## SYSTEM INTEGRATION MODULE

The MC68340 SIM40 provides the external bus interface for both the CPU32 and the DMA. It also eliminates much of the glue logic that typically supports the microprocessor and its interface with the peripheral and memory system. The SIM40 provides programmable circuits to perform address decoding and chip selects, wait-state insertion, interrupt handling, clock generation, bus arbitration, watchdog timing, discrete I/O, and power-on reset timing. A boundary scan test capability is also provided.

### External Bus Interface

The external bus interface (EBI) handles the transfer of information between the internal CPU32 or DMA controller and memory, peripherals, or other processing elements in the external address space. Based on the MC68030 bus, the external bus provides up to 32 address lines and 16 data lines. Address extensions identify each bus cycle as CPU32 or DMA initiated, supervisor or user privilege level, and instruction or data access. The data bus allows dynamic sizing for 8- or 16-bit bus accesses (plus 32 bits for DMA). Synchronous transfers for the CPU32 or the DMA can be made in as little as two clock cycles. Asynchronous transfers allow the memory system to signal the CPU32 or DMA when the transfer is complete and to note the number of bits in the transfer. An external master can arbitrate for the bus using a three-line handshaking interface.

### System Configuration And Protection

The M68000 family of processors is designed with the concept of providing maximum system safeguards. System configuration and various monitors and timers are provided in the MC68340. Power-on reset circuitry is a part of the SIM40. A bus monitor ensures that the system does not lock up when there is no response to a memory access. The bus fault monitor can reset the processor when a catastrophic bus failure occurs. Spurious interrupts are detected and handled appropriately. A software watchdog can pull the processor out of an infinite loop. An interrupt can be sent to the CPU32 with programmable regularity for DRAM refresh, time-of-day clock, task switching, etc.

## Clock Synthesizer

The clock synthesizer generates the clock signals used by all internal operations as well as a clock output used by external devices. The clock synthesizer can operate with an inexpensive 32768-Hz watch crystal or an external oscillator for reference, using an internal phase-locked loop and voltage-controlled oscillator. At any time, software can select clock frequencies from 131 kHz to 16.78 MHz or 25.16 MHz, favoring either low power consumption or high performance. Alternately, an external clock can drive the clock signal directly at the operating frequency. With its fully static HCMOS design, it is possible to completely stop the system clock without losing the contents of the internal registers.

## Chip Select And Wait State Generation

Four programmable chip selects provide signals to enable external memory and peripheral circuits, providing all handshaking and timing signals with up to 175-ns access times with a 25-MHz system clock (265 ns @ 16.78 MHz). Each chip select signal has an associated base address and an address mask that determine the addressing characteristics of that chip select. Address space and write protection can be selected for each. The block size can be selected from 256 bytes up to 4 Gbytes in increments of  $2^n$ . Accesses can be preselected for either 8- or 16-bit transfers. Fast synchronous termination or up to three wait states can be programmed, whether or not the chip select signals are used. External handshakes can also signal the end of a bus transfer. A system can boot from reset out of 8-bit-wide memory, if desired.

## Interrupt Handling

Seven input signals are provided to trigger an external interrupt, one for each of the seven priority levels supported. Seven separate outputs can indicate the priority level of the interrupt being serviced. An input can direct the processor to a default service routine, if desired. Interrupts at each priority level can be preprogrammed to go to the default service routine. For maximum flexibility, interrupts can be vectored to the correct service routine by the interrupting device.

## Discrete I/O Pins

When not used for other functions, 16 pins can be programmed as discrete input or output lines. Additionally, in other peripheral modules, pins for otherwise unused functions can often be used for general input/output.

## IEEE 1149.1 Test

To aid in system diagnostics, the MC68340 includes dedicated user-accessible test logic that is fully compliant with the IEEE 1149.1 standard for boundary scan testability, often referred to as JTAG (Joint Test Action Group).

## DIRECT MEMORY ACCESS MODULE

The most distinguishing MC68340 characteristic is the high-speed 32-bit DMA controller, used to quickly move large blocks of data between internal peripherals, external peripherals, or memory, without processor intervention. The DMA module consists of two, independent, programmable channels. Each channel has separate request, acknowledge, and done signals. Each channel can operate in a single-address (flyby) or a dual-address mode.

In single-address mode, only one (the source or the destination) address is provided, and a peripheral device such as a serial communications controller receives or supplies the data. An external request must start a single-address transfer. In this mode, each channel supports 32 bits of address and 8, 16, or 32 bits of data.

In dual-address mode, two bus transfers occur, one from a source device and the other to a destination device. Dual-address transfers can be started by either an internal or external request. In this mode, each channel supports 32 bits of address and 8 or 16 bits of data (32 bits require external logic). The source and destination port size can be selected independently; when they are different, the data will be packed or unpacked. An 8-bit disk interface can be read twice before the concatenated 16-bit result is passed into memory.

Byte, word, and long-word counts up to 32 bits can be transferred. All addresses and transfer counters are 32 bits. Addresses increment or remain constant, as programmed. The DMA channels support two external request modes, burst transfer and cycle steal. Internal requests can be programmed to occupy 25, 50, 75, or 100 percent of the data bus bandwidth. Interrupts can be programmed to postpone DMA completion.

The DMA module can sustain a transfer rate of 12.5 Mbytes/sec in dual-address mode and nearly 50 Mbytes/sec in single-address mode @ 25.16 MHz (8.4 and 33.3 Mbytes/sec @ 16.78 MHz, respectively). The DMA controller arbitrates with the CPU32 for the bus in parallel with existing bus cycles and is fully synchronized with the CPU32, eliminating all delays normally associated with bus arbitration by allowing DMA bus cycles to butt seamlessly with CPU bus cycles.

## **SERIAL MODULE**

Most digital systems use serial I/O to communicate with host computers, operator terminals, or remote devices. The MC68340 contains a two-channel, full-duplex USART. An on-chip baud rate generator provides standard baud rates up to 76.8k baud independently to each channel's receiver and transmitter. The module is functionally equivalent to the MC68681/MC2681 DUART.

Each communication channel is completely independent. Data formats can be 5, 6, 7, or 8 bits with even, odd, or no parity and stop bits up to 2 in 1/16 increments. Four-byte receive buffers and two-byte transmit buffers minimize CPU service calls. A wide variety of error detection and maskable interrupt capability is provided on each channel. Full-duplex, autoecho loopback, local loopback, and remote loopback modes can be selected. Multidrop applications are supported.

A 3.6864-MHz crystal drives the baud rate generators. Each transmit and receive channel can be programmed for a different baud rate, or an external 1× and 16× clock input can be selected. Full modem support is provided with separate request-to-send (RTS) and clear-to-send (CTS) signals for each channel. One channel also provides service request signals. The two serial ports can sustain rates of 9.8 Mbps with a 25-MHz system clock in 1× mode, 612 kbps in 16× mode (6.5 Mbps and 410 kbps @ 16.78 MHz).

## **TIMER MODULES**

Timers and counters are used in a system to monitor elapsed time, generate waveforms, measure signals, keep time-of-day clocks, initiate DRAM refresh cycles, count events, and provide "time slices" to ensure that no task dominates the activity of the processor. A counter that counts clock pulses makes a timer, which is most useful when it causes certain actions to occur in response to reaching desired counts.

The MC68340 has two, identical, versatile, on-chip counter/timers as well as a simple timer in the SIM40. These general-purpose counter/timers can be used for precisely timed events without the errors to which software-based counters and timers are susceptible—e.g., errors caused by dynamic memory refreshing, DMA cycle steals, and interrupt servicing. The programmable timer operating modes are input capture, output compare, square-wave generation, variable duty-cycle square-wave generation, variable-width single-shot pulse generation, event counting, period measurement, and pulse-width measurement.

Each timer consists of a 16-bit countdown counter with an 8-bit countdown prescaler for a composite 24-bit resolution. The two timers can be externally cascaded for a maximum count width of 48 bits. The

counter/timer can be clocked by the internal system clock generated by the SIM40 (+2) or by an external clock input. Either the processor or external stimuli can trigger the starting and stopping of the counter. When a counter reaches a predetermined value, either an external output signal can be driven, or an interrupt can be made to the CPU32. The finest resolution of the timer is 80 ns with a 25-MHz system clock (125 ns @ 16.78 MHz).

## POWER CONSUMPTION MANAGEMENT

The MC68340 is very power efficient due to its advanced 0.8- $\mu$  HCMOS process technology and its static logic design. The resulting power consumption is typically 500 mW in full operation @ 16.78 MHz (750 mW @ 25 MHz)—far less than the comparable discrete component implementation the MC68340 can replace. For applications employing reduced voltage operation, selection of the MC68340V, which requires only a 3.3-V power supply, reduces current consumption by 40–60% in all modes of operation (as well as reducing noise emissions).

The MC68340 has many additional methods of dynamically controlling power consumption during operation. The frequency of operation can be lowered under software control to reduce current consumption when performance is less critical. Idle internal peripheral modules can be turned off to save power (5–10% each). Running a special low power stop (LPSTOP) instruction shuts down the active circuits in the CPU and peripheral modules, halting instruction execution. Power consumption in this standby mode is reduced to about 300  $\mu$ W. Processing and power consumption can be resumed by resetting the part or by generating an interrupt which can be done with the SIM40's periodic interrupt timer.

## PHYSICAL

The MC68340 is available as 0–16.78 MHz and 0–25.16 MHz, 0°C to +70°C and -40°C to +85°C, and 5.0 V  $\pm$ 5% and 3.3 V  $\pm$ 0.3 supply voltages (reduced frequencies at 3.3 V). Thirty-two power and ground leads minimize ground bounce and ensure proper isolation of different sections of the chip, including the clock oscillator. A 144 pins are used for signals and power. The MC68340 is available in a gull-wing ceramic quad flat pack (CQFP) with 0.65 mm lead spacing or a 15  $\times$  15 plastic pin grid array (PPGA) with 0.1-in pin spacing.

## COMPACT DISC-INTERACTIVE

The MC68340 was designed to meet the needs of many markets, including compact disc-interactive (CD-I). CD-I is an emerging standard for a publishing medium that will bring multimedia to a broad general audience—the consumer. CD-I players combine television and stereo systems as output devices, with interactive control using a TV remote-control-like device to provide a multimedia experience selected from software “titles” contained in compressed form on standard compact discs.

The highly integrated MC68340 is ideal as the central processor for CD-I players. It provides the M68000 microprocessor code compatibility and DMA functions required by the CD-I Green Book specification as well as many other useful on-chip functions for a very cost-effective solution. The extra demands of full-motion video CD-I systems make the best use of the MC68340 high performance. The MC68340 is CD-I compliant and has been CD-I qualified. With its low voltage operation, the MC68340V is the only practical choice for portable CD-I.

## MORE INFORMATION

The following table identifies the packages, supply voltages, temperature range, and operating frequencies available for the MC68340.


**MC68340 Package/Frequency Availability**

Package	Frequency/Volts			
	8 MHz/3.3 V	16 MHz/3.3 V	16 MHz/5 V	25 MHz/5 V
Plastic Pin Grid Array (RP)	4	4	4	4
Ceramic Quad Flat Pack (FE)	4	4	4	4
<b>Temperature</b>	0–70 °C/-40–85 °C	0–70 °C	0–70 °C/-40–85 °C	0–70 °C

The documents listed in the following table contain detailed information on the MC68340. These documents may be obtained from the Literature Distribution Centers at the addresses listed at the bottom of this page.

### Documentation

Document Number	Document Name
BR1114/D	M68300 Integrated Processor Family
MC68340UM/AD	MC68340 User's Manual
M68000PM/AD	M68000 Family Programmer's Reference Manual
AN1063/D	DRAM Controller for the MC68340
AN453	Software Implementation of SPI on the MC68340
BR573/D	M68340 Evaluation System Product Brief
BR729/D	The 68K Source
BR1407/D	3.3 Volt Logic and Interface Circuits

Motorola reserves the right to make changes without further notice to any products herein. Motorola makes no warranty, representation or guarantee regarding the suitability of its products for any particular purpose, nor does Motorola assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. "Typical" parameters can and do vary in different applications. All operating parameters, including "Typical" must be validated for each customer application by customer's technical experts. Motorola does not convey any license under its patent rights nor the rights of others. Motorola products are not designed, intended, or authorized for use as components in systems intended for surgical implant into the body, or other applications intended to support or sustain life, or for any other application in which the failure of the Motorola product could create a situation where personal injury or death may occur. Should Buyer purchase or use Motorola products for any such unintended or unauthorized application, Buyer shall indemnify and hold Motorola and its officers, employees, subsidiaries, affiliates, and distributors harmless against all claims, costs, damages, and expenses, and reasonable attorney fees arising out of, directly or indirectly, any claim of personal injury or death associated with such unintended or unauthorized use, even if such claim alleges that Motorola was negligent regarding the design or manufacture of the part. Motorola and  are registered trademarks of Motorola Inc. Motorola, Inc. is an Equal Opportunity/Affirmative Action Employer.

### Literature Distribution Centers:

USA: Motorola Literature Distribution; P.O. Box 20912; Phoenix, Arizona 85036.

EUROPE: Motorola Ltd.; European Literature Centre; 88 Tanners Drive, Blakelands, Milton Keynes, MK14 5BP, England.

JAPAN: Nipon Motorola Ltd.; 4-32-1, Nishi-Gotanda, Shinagawa-ku, Tokyo 141 Japan.

ASIA-Pacific: Motorola Semiconductors H.K. Ltd.; Silicon Harbour Center, No. 2 Dai King Street, Tai Po Industrial Estate, Tai Po, N.T., Hong Kong.



**MOTOROLA**