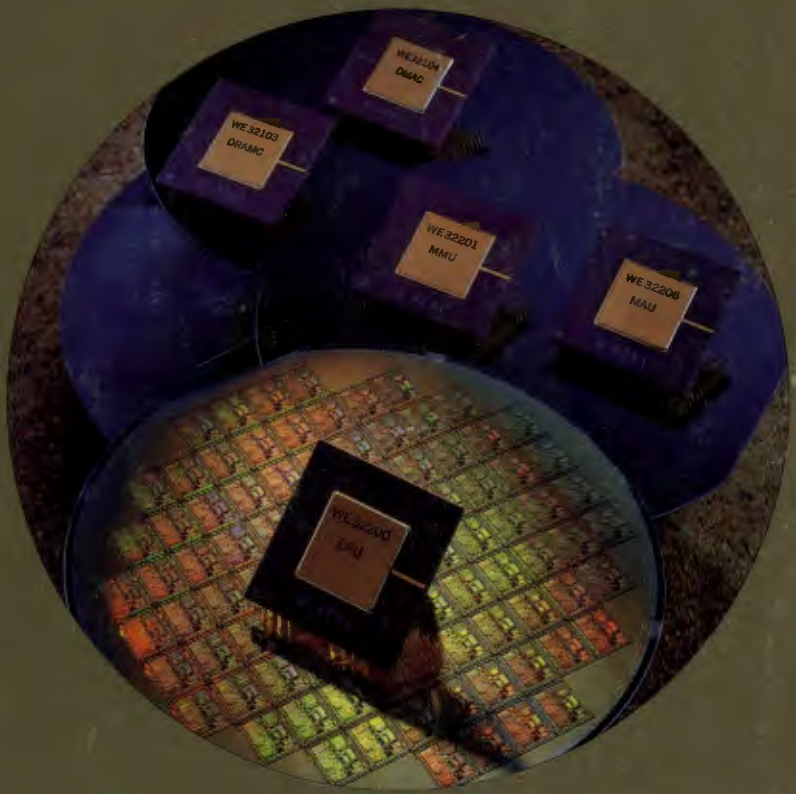
Data Book

AT&T

AT&T *WE*® 32-Bit Microprocessors and Peripherals



August 1987

August 1987

AT&T WE® 32-Bit Microprocessors and Peripherals Data Book

A Word About Trademarks . . .

The following products mentioned in this Data Book are identified with AT&T trademarks:

WE® 32100 Microprocessor
WE® 32200 Microprocessor
WE® 32101 Memory Management Unit
WE® 32201 Memory Management Unit
WE® 32102 Clock
WE® 32103 DRAM Controller
WE® 32104 DMA Controller
WE® 32204 DMA Controller
WE® 32106 Math Acceleration Unit
WE® 32206 Math Acceleration Unit
WE® 321AP Microprocessor Analysis Pod
WE® 321DM CPU & MMU Device Monitor
WE® 322DM CPU & MMU Device Monitor
WE® 321DM-SP Device Monitor Software Programs
WE® 321DS Microprocessor Development System
WE® 321EB Microprocessor Evaluation Board
WE® 321SB VMEbus Single Board Computer
WE® 321SB-VFG Firmware Generation Programs
WE® 321SB-VSD Software Debug Utility Programs
WE® 321SD Development Software Programs
WE® 321SE Evaluation Software Programs
WE® 321SG Software Generation Programs
WE® 321SG C-Software Generation Programs
WE® 321SG Cross C-Software Generation Programs
WE® 321SG-CUX Cross *UNIX* System Software Generation Programs
WE® 321SG-CRS Cross Software Generation Programs
WE® 321SG-NAT Native Software Generation Programs
WE® 32-Bit Microprocessors and Peripherals
UNIX® Operating System, System V, System V/VME, or System

VAX® 11/780 Computer is a registered trademark of Digital Equipment Corporation
MS-DOS® is a registered trademark of the Microsoft Corporation

Data Sheets Status Markings . . .

Data sheets without a status marking are final and are issued when tool made samples are approved or when the product is delivered for usage, whichever comes first. The other data sheet status markings used throughout this catalog are defined as:

Advance – Issued at the exploratory stage of development when the principle characteristics are available. Some functional characteristics are subject to change.

Preliminary – Issued after development for manufacture has been started. Some electrical and timing parameters are subject to change.

Information contained herein is subject to change.

Copyright © 1987 AT&T. All Rights Reserved.

Printed in United States of America

AT&T WE® 32-Bit Microprocessors and Peripherals Data Book

Chapter 1. General

Introduction 1-1
The System 1-1
The *WE* 32100/32200 Microprocessors 1-1
Peripheral Devices 1-1
Hardware/Software Support Tools 1-2
Board-Level Products 1-3
Development Software 1-3
Operating System Software 1-3
Quality Control and Reliability 1-4
Quality Control 1-4
Reliability 1-4
High Temperature Operating Bias/High Temperature High Bias (HTOB/HTHB) 1-4
Temperature Humidity Bias (THB) 1-5
Temperature Cycling (TC) 1-5
Handling 1-5
Integrated Circuits 1-5
Board-Level Products 1-5
Diskettes 1-6
Military Specification Devices 1-6
Assistance 1-6
Warranties 1-7

Chapter 2. Devices

WE 32100 Microprocessor 2-1
WE 32200 Microprocessor 2-65
WE 32101 Memory Management Unit 2-145
WE 32201 Memory Management Unit 2-183
WE 32102 Clock 2-223
WE 32103 DRAM Controller 2-231
WE 32104 DMA Controller 2-313
WE 32204 DMA Controller 2-363
WE 32106 Math Acceleration Unit 2-365
WE 32206 Math Acceleration Unit 2-395
WE 32-Bit Microprocessors and Peripherals Packaging 2-429

Chapter 3. Development Support Tools

WE 321DM CPU & MMU Device Monitor 3-1
WE 322DM CPU & MMU Device Monitor 3-5
WE 321DS Microprocessor Development System 3-9
WE 321EB Microprocessor Evaluation Board 3-13

Chapter 4. Board-Level Products

WE 321SB VMEbus Single Board Computer 4-1

Chapter 5. Support Software

WE 321SG Software Generation Programs Version 1.1 5-1
WE 321SG C-Software and Cross C-Software Generation Programs Version 2.0 5-5
UNIX System V Release 2.1/3.1 5-9
UNIX System V/VME 5-11

Chapter 6. Training

Training 6-1
Educational Programs 6-1

Chapter 7. Ordering Information

Ordering Information 7-1
Documentation 7-2

Device Alphanumeric Index

WE 32100 Microprocessor 2-1
WE 32101 Memory Management Unit 2-145
WE 32102 Clock 2-223
WE 32103 DRAM Controller 2-231
WE 32104 DMA Controller 2-313
WE 32106 Math Acceleration Unit 2-365
WE 32200 Microprocessor 3-65
WE 32201 Memory Management Unit 2-183
WE 32204 DMA Controller 2-363
WE 32206 Math Acceleration Unit 2-395

Chapter 1

General

Introduction

The System

This data book contains technical information about the AT&T *WE* 32-Bit Microprocessors and their support systems. The support systems consist of peripheral devices, single board products, software, hardware/software support tools, documentation, educational programs, and applications assistance.

The AT&T *WE* 32-Bit Microprocessors and Peripherals offer a complete package of devices and design tools that can speed up design time and get your product to the market in a timely and efficient manner. The AT&T *WE* 32-Bit Microprocessor chip set is the only one with 100 percent CMOS technology for low power consumption and compact designs. It offers glue-free interconnections and keeps device count, development time, and overall system costs to a minimum.

Each microprocessor and peripheral is built for the high-speed performance expected from a 32-bit system, supporting multi-user, multi-tasking applications. Designed to evolve into a new generation product with even higher performance, using the AT&T *WE* 32-Bit Microprocessor chip set ensures that the system being designed will be as current in the future as it is today.

The chip set, designed with the *UNIX* operating system in mind, offers the best hardware support possible for the world's most productive operating system. *UNIX* System V offers a highly optimized C language compiler, and can be used with other compilers for high level languages such as FORTRAN, COBOL, Pascal, and BASIC. These capabilities make *UNIX* System V the best development environment for 32-bit applications. Due to its flexibility, the AT&T *WE* 32-Bit Microprocessor chip set can be used to design a variety of products. You can offer a wide range of products from a single-user, low-priced version to high-performance, multi-user systems. Your initial design can be expanded or easily modified at any time, extending the life cycle of your product to accommodate new and changing technologies.

The WE® 32100/32200 Microprocessors

The *WE* 32100/32200 Microprocessors are high-performance, single-chip, 32-bit central processing units (CPUs) designed for efficient operation in a high-level language environment. These microprocessors represent a state-of-the-art concept in microprocessor architecture, providing one of the most powerful and extensive instruction sets available with any microprocessor.

Applications for these processors can be configured at the chip level or at the board level. At either level, the support system makes application design manageable and cost effective.

The *WE* 32200 Microprocessor is protocol and upward object code compatible with the *WE* 32100 CPU, and object code generated for the *WE* 32100 CPU will run without modification on the *WE* 32200 CPU. Enhancements designed into the *WE* 32200 Microprocessor include thirty-two 32-bit registers (the *WE* 32100 CPU has sixteen), expanded instruction set, expanded addressing modes, and byte replication on writes. Another feature, arbitrary byte alignment, allows the *WE* 32200 CPU to efficiently handle nonaligned memory accesses for both reads and writes. The *WE* 32200 CPU also utilizes dynamic bus sizing, allowing communication with both 16-bit and 32-bit memories and peripherals.

The *WE* 32100 Microprocessor is implemented in 1.0-micron CMOS technology and is packaged in a 125-pin square, ceramic pin grid array (PGA). It is available in 10-, 14-, and 18-MHz frequency versions.

The *WE* 32200 Microprocessor is implemented in 1.0-micron CMOS technology and is packaged in a 133-pin square, ceramic PGA. It is available in 24-MHz and higher frequency versions.

Peripheral Devices

The essential components of a microcomputer include a central processing unit, memory, and input/output circuitry. The peripheral devices provide the interfacing, wait-state generation, and additional circuitry needed to support the processor in its microcomputer application.

The *WE* 32101 Memory Management Unit (MMU) and the *WE* 32201 MMU provide logical-to-physical address translations, memory organization, and control and access protection for the microprocessors.

Advantages of the *WE* 32201 MMU over the *WE* 32101 MMU include a 4-Kbyte data/instruction cache; an on-chip, fully associative, content addressable memory (CAM) based, 64-entry descriptor cache; transparent multiple-context support; a flexible translation probe; and variable page sizes.

The *WE* 32102 Clock supplies the two-phase, CMOS-level frequency source required by the 32-bit microprocessors chip set. It is available at operating frequencies of 10, 14, 18, and 24 MHz and higher.

The *WE* 32103 DRAM Controller provides address multiplexing, access and cycle time management, and refresh control for dynamic random access memory (DRAM). The DRAM Controller is capable of addressing up to 16 Mbytes of DRAM, using 1 Mbit memory devices.

The *WE* 32104 DMA Controller (DMAC) is a memory-mapped peripheral device that performs direct memory-to-memory, memory-to-peripheral, and peripheral-to-memory data transfers quickly and efficiently. Specialized hardware permits these transfers at a rate much faster than a typical CPU.

The *WE* 32106 Math Acceleration Unit (MAU) and the *WE* 32206 MAU provide floating-point capability for the *WE* 32100/32200 Microprocessors. Both MAUs are fully compatible with ANSI/IEEE Standard 754-1985 for Binary Floating-Point Arithmetic, and provide single, double, and double-extended precision capabilities.

The *WE* 32206 MAU is footprint, protocol, and upward object code compatible with the *WE* 32106 MAU. In addition, the *WE* 32206 MAU has eight 80-bit user registers (the *WE* 32106 MAU has four). The transcendental functions (sine, cosine, and arctangent) and pi have also been added to the *WE* 32206 MAU's instruction set.

Hardware/Software Support Tools

The *WE* 321DS Microprocessor Development System is an integrated collection of hardware and software components used to develop, test, and debug applications based on the *WE* 32100 Microprocessor. Development System components include the *WE* 321AP Microprocessor Analysis Pod, the *WE* 321SD Development Software Programs, a *UNIX* System host, a high-level symbolic debugger, bus state analysis software, and a logic analyzer. The modular architecture of the Development System permits the user to configure the Analysis Pod, *UNIX* System host, and logic analyzer into any of four work stations that emphasize either hardware testing, software testing, or hardware/software integration and testing.

The *WE* 321DM/322DM CPU & MMU Device Monitors provide a low cost solution for debugging applications based on the *WE* 32100/32200 Microprocessors with a logic analyzer that is limited to a maximum sample rate of 10 MHz.

The device monitor consists of a logic retiming unit, a CPU cable system, an MMU cable system, and additional cables that provide a standard interface to a logic analyzer. The CPU and MMU cable systems contain footprint connectors that replace the CPU and MMU devices in the target system. The *WE* 321DM-SP Device Monitor Software Programs, supplied with the device monitor, provide for symbolic tracing and disassembly with the logic analyzer. This software includes three program trace disassemblers, configuration files, logic analyzer upload and download utilities, a symbol table utility, and a communication file for communication between the host *UNIX* System and the logic analyzer.

The *WE* 321EB Microprocessor Evaluation Board is a *WE* 32100 Microprocessor based single-board microcomputer evaluation system. It can be used as a prototyping system to evaluate the performance and the hardware/software capabilities of the *WE* 32100 Microprocessor in an application environment.

The evaluation board contains a *WE* 32100 Microprocessor operating at 18 MHz, an 18-MHz *WE* 32102 Clock, a *WE* 32101 Memory Management Unit, and a *WE* 32106 Math Acceleration Unit. Also included are a ROM-resident monitor program, 96 Kbytes of RAM,

address decoding, RS-232C serial I/O ports, programmable parallel I/O lines, programmable interval timers, and an interrupt controller.

Board-Level Products

The *WE* 321SB VMEbus Single Board Computer (SBC) is a 32-bit single-board computer that enables the rapid development of VMEbus systems for applications that require the highest level of performance, either in a *UNIX* System or other operating system environment.

The SBC is provided in a double-height Eurocard format and interfaces with VMEbus peripheral boards through the Rev. C.1 VMEbus interface. Included with the SBC are the *WE* 32100 Microprocessor, *WE* 32101 Memory Management Unit, and *WE* 32106 Math Acceleration Unit, 1 Mbyte of DRAM, EPROM/ROM configured up to 256 Kbytes, two serial ports, three timer/counters, and interrupt circuitry. The SBC can also be configured to operate with *UNIX* System V/VME Release 2.1 or 3.1 Software.

Development Software

The *WE* 321SG C-Software Generation Programs (C-SGP) are a collection of support software tools that allow the use of the computing power of a host *UNIX* System to aid in the creation and debugging of software for the *WE* 32100 Microprocessor chip set.

The C-SGP products satisfy two user programming environments:

- **Native Environment** – the *WE* 321SG C-Software Generation Programs (native C-SGP) for systems on which the host is the target system and the host/target system is running the *UNIX* Operating System
- **Cross Environment** – the *WE* 321SG Cross C-Software Generation Programs (Cross C-SGP) for systems on which the host is running the *UNIX* Operating System, and the target contains the *UNIX* Operating System, an operating system other than the *UNIX* Operating System, or is embedded in an environment with no operating system.

Each of the user programming environments can be configured in either of two floating-point modes: math acceleration unit (**MAU**) or floating-point emulation (**FPE**). The **MAU** mode provides enhanced floating-point performance through the use of in-line compilation of the MAU instruction set (MIS) for the *WE* 32106 Math Acceleration Unit. Alternatively, the **FPE** mode provides the flexibility of supporting target systems with or without a math acceleration unit. The **FPE** mode compiles floating-point operations as calls to library routines that either execute MIS or emulate the operations, depending on whether a math acceleration unit has been determined to be present at process start-up. For the Cross C-SGP, the **FPE** mode assumes the target system does not have a math acceleration unit. This support software provides utility programs that enable the user to write applications software in C language or assembly language, and then translate it into machine code.

Operating System Software

UNIX System V is a multi-user operating system whose modular design simplifies the manipulation of files and commands. The hierarchical file structure allows for easy adding, deleting, and moving of files within a structure set up by the user. Input to commands can come from a terminal or a file, and output can be directed to files or to peripheral devices. The output of a command can be directed to the input of another command, allowing users to easily create specialized functions.

The *UNIX* System shell is also a programming language that can be used to create user definable applications and functions.

The *UNIX* System oversees the execution of many user programs and commands. Although these programs and commands seem to execute simultaneously, each application is scheduled to use the processor for a short period of time to the exclusion of all other programs. Thus, in addition to providing a multi-user system, the *UNIX* System affords each user the capability of running several programs at once. This feature is called multi-programming. Many *UNIX* System based products are available from AT&T and other vendors. Descriptions of these products can be found in the **AT&T Computer Software Catalog**. Products offered through the AT&T *UNIX* System Toolchest are available to customers who have a *UNIX* System source license.

Quality Control and Reliability

AT&T standards and stringent processing controls, along with the design and construction techniques developed at AT&T Bell Laboratories, ensure the excellent quality and reliability of AT&T devices. Laboratory tests and field failure data confirm that high quality, reliable products are produced at the component, circuit pack/board, and system levels. Devices are tested by a comparison of the actual electrical, mechanical, or visual properties of the device against what is specified to the user or what is specified as a manufacturing requirement. Quality is determined at a single point in time, reliability is ascertained over a longer period of time.

Quality Control

The engineering Quality Control (QC) and Final Inspection (FI) organizations located at each integrated circuit (IC) manufacturing plant perform the actual inspection and testing of devices. After the products have been inspected, the resident Quality Assurance (QA) organizations audit the results of the tests to determine acceptability. Quality is usually stated in terms of the number of defects contained in an IC population.

Various mechanical tests are performed to ensure the integrity of internal lead bonds and the strength of exterior leads. Packages are visually inspected during assembly to check for defects such as poor alignment of bonds or damaged wires that could be a reliability risk. Assembly shop tests are then repeated by QA and FI on a sample of devices prior to shipment. Tests performed include worst-case conditions, speed, leakage, power supply current, etc. The tests are usually performed at room temperature and/or elevated temperature.

Operational life testing (OLT) is a method of testing for reliability using simulated worst-case field conditions. Mechanical and electrical stress conditions beyond normal specifications are applied to accelerate latent manufacturing defects. Effective screens for mechanical problems are thermal cycling and hot testing, since some mechanical defects are only minimally affected by accelerated stress conditions.

Board-level products are given visual inspections during and after each phase of assembly. Upon completion of assembly, the boards are subjected to computer controlled testing, checking for manufacturing defects, opens, shorts, resistance values, and in some cases, capacitance values. Next, a burn-in procedure is performed, simulating 100 days of operation under accelerated temperatures and other stressful conditions. This test is used to detect early defects. The boards are then subjected to a rigorous functional test, emulating actual operating environments. During this test, more board options are selected than would be normally used by a customer. Finally, another inspection and further testing is performed by an independent Quality Assurance team.

Reliability

In order to provide a comprehensive measure of product reliability, a two-tier program of stressing is used.

Level 1. Level 1 is the extended life testing program in which a sample of 100 devices is taken once every six weeks from each reliability testing group and stressed for periods simulating 40 years of product life. The length of stressing time depends on the acceleration factors determined for the individual tests.

Level 2. Level 2 is the normal monitoring program in which a sample of 100 devices is chosen from each testing group every week and stressed for a period approximating 10 years of product life.

To facilitate product sampling and ensure complete coverage of the reliability program, all products are grouped into families according to their basic design and function, such as custom logic, memories, microprocessors, codec/analog and digital signal processors. The families are further divided into test groups for sample selection testing. Test groups are based on factors most likely affected by the tests being performed. An explanation of each test is given below.

High Temperature Operating Bias/High Temperature High Bias (HTOB/HTHB)

The HTOB/HTHB testing accelerates failure mechanisms that are thermally excited by high temperatures, such as ionic drift, electrothermal migration, oxide breakdown, silicon material

defects, and assembly-related mechanisms. Another concern is the interaction between packaging materials and silicon processing. Therefore, reliability test groupings for HTOB/HTHB are based on silicon processing, design rules, silicon fabrication facilities and packaging lines.

HTOB/HTHB is performed at the following temperatures and times:

	Level 1	Level 2	Accel. Factor
125 °C	2350 hrs	700 hrs	128
135 °C	1260	385	238
150 °C	525	160	570

Temperature Humidity Bias (THB)

THB tests the effectiveness of chip passivation and device packaging. High humidities in the presence of electrical bias promotes electrochemical corrosion, electrothermal migration and other chemical reactions involving the presence of water. Test groupings for THB are based on packaging materials, package types and packaging lines. The test conditions for this test are 85°C and 85% relative humidity (85/85).

Two options are available for stressing parts for THB. They are:

Option 1. Devices are soaked in the 85/85 chamber with no bias for 48 hours, after which time the bias is applied and the parts are stressed for 240 hours for level 2 conditions and 800 hours for level 1 conditions.

Option 2. Devices are placed in a steam bomb at 121 °C/15 psig with no bias for 96 hours after which they are placed in the 85/85 chamber with bias applied for 48 hours for level 2 conditions and 160 hours for level 1 conditions. The steam bomb increases acceleration by a factor of 5.

Temperature Cycling (TC)

TC applies thermally-induced stress to the devices to accelerate material fatigue and precipitate failures associated with thermal expansion mismatches. TC groupings are based on packaging materials, package types, and packaging lines.

Devices are stressed from –40 °C to +130 °C with no bias applied. The level 2 stressing duration is 60 cycles; the level 1 stressing duration is 180 cycles.

Handling

AT&T products have long life expectancy with a corresponding low failure rate when handling and operating specifications are followed by the user. Operating specifications include product mounting, use, operating limits, power requirements, environmental conditions, and other items specific to the product.

Integrated Circuits

Observe the following instructions when handling and using integrated circuits.

Precaution against static discharge must be observed always. This includes grounding all personnel and equipment before contact with static-sensitive devices.

In assembly, the device solder bath (fountain) temperature should not exceed 300 °C for a maximum bath time limited to 10 seconds. For installation with a soldering iron, the iron temperature (tip) should not exceed 500 °C, and solder time per lead should be limited to 5 seconds.

Following completion of assembly and soldering operations, all printed wiring assemblies should be cleaned to remove fingerprints, dust, dirt, grease, excessive flux residue, and other foreign matter. A brush cleaning process using solvents is the acceptable cleaning method, provided only occasional isolated droplets of solvent come in contact with the devices and the component side of the board. The recommended solvents are chlorinated hydrocarbons such as trichlorethylene, 1,1,1-trichloroethane, and perchloroethylene.

Board-Level Products

Observe these precautions when unpacking and handling board level products:

- Ground yourself before handling; circuit boards contain static-sensitive devices
- Avoid touching areas of integrated circuits; static discharge can damage these circuits
- Store boards in conductive bags when not in use.

Diskettes

The following guidelines should be observed in order to protect the data stored on diskettes:

- Store diskettes vertically in their protective sleeves
- Protect diskettes from direct sunlight
- Storage and operating temperature ranges for diskettes are 50 °F—125 °F (10 °C—50 °C)
- Always handle a diskette by the sleeve; never touch exposed surfaces
- Never bend or fold diskettes
- Do not use or store diskettes near magnets or other sources of magnetic fields
- Use only felt-tipped pens when writing on diskettes
- Exercise caution when inserting diskettes into diskette drives.

Military Specification Devices

AT&T will offer military versions of its *WE* 32100 Microprocessors and Peripherals. The devices will meet military standards 883 and possibly the Joint Army-Navy (JAN) qualification. Initial production of the CPU, MMU, and MAU will be set at 10 MHz in the second half of 1988. The DRAM controller, the DMA controller, and 14-MHz CPU, MMU, and MAU are scheduled to be offered in 1989. Future plans include offerings of military versions of the *WE* 32200 Microprocessors and Peripherals.

Assistance

This data book contains technical information on the *WE* 32-Bit Microprocessors, peripherals, and hardware and software support systems.

Additional information is available in the form of manuals, application notes, and documentation from the *UNIX* Operating System.

For details concerning warranty, repairs, or ordering or to obtain additional information, contact your AT&T Account Manager or call:

□ AT&T Technologies
Dept. 51AL603140
555 Union Boulevard
Allentown, PA 18103
1-800-372-2447

□ AT&T Microelectronics
Freischützstrasse 92
8000 Munich 81
West Germany
Tel. 0 89/95 97 0
Telex 5 216 884

□ AT&T Microlelectronics Pte. Ltd.
745 Larong 5 Toa Payoh
Singapore 1231
Tel. 250 8422/253 3722
Telex RS 21473/RS 55038

□ AT&T International Japan Ltd.
Nippon Press Center Bldg.
2-1, Uchisaiwai-cho, 2-Chome,
Chiyoda-ku, Tokyo 100 Japan
Tel. (03) 593 3301
Telex J32562 ATTIJ

Warranties

Contracts of sale for the items described in this Data Book will include the following warranty language.

Device Warranty

Seller warrants to buyer that products of its manufacture will be, on the date of shipment of the product, free from defects in material and workmanship and will conform to Seller's written specifications provided to Buyer or to the specifications, if any, identified in an order and accepted by Seller, other than specifications specifying performance for a period of time. If any defect in material or workmanship or failure to meet said published specifications (a "defect") appears in the product, Seller will, at its option, either repair or replace the defective product without charge at Seller's manufacturing or repair facility or credit or refund the purchase price of the defective product provided [i] the defect appears within twelve (12) months from the date of shipment of the product, [ii] Buyer notifies Seller in writing of the claimed defect no later than thirty (30) days after the end of the twelve (12) month period, and [iii] Seller's examination of the product discloses that the claimed defect actually exists.

Buyer shall follow Seller's instructions regarding return of defective product, and no product will be accepted for repair, replacement, credit, or refund without the written authorization of and in accordance with Seller's instructions. In the case of any such return, Buyer shall bear the risk of loss or damage and shall prepay all transportation chargers to Seller. Repaired or replacement product will be shipped, freight prepaid by Seller, and Buyer shall bear the risk of loss or damage. The replaced product shall become Seller's property. In no event shall Seller be responsible for destination or reinstallation of the product or for the expenses thereof. If Seller determines that the products are not defective, Buyer shall pay Seller all costs of handling, inspection, repairs, and transportation at Seller's then prevailing rates.

Repairs and replacements covered by the above warranty are warranted to be free from defects as set forth above except that the defect must appear [i] with three (3) months from the date of repair or replacement or [ii] prior to the expiration of the above twelve (12) month period, whichever is later.

With respect to products not manufactured by Seller, Seller, to the extent permitted, extends the warranties and affords the remedies to Buyer given to Seller by its vendor of said products.

The foregoing warranties do not extend [a] to expendable items or [b] to experimental or developmental products or [c] to products which have [i] been subject to misuse, neglect, accident or abuse; [ii] been wired, repaired or altered by anyone other than Seller; [iii] been used in material violation of Seller's instructions, or [iv] have had their serial numbers or month and year of manufacture or shipment removed, defaced, or altered. The term "software" means a set of logical instructions and tables of information which guide the functioning of a processor. Such set may be contained in any medium whatsoever, including without limitation hardware containing a pattern of bits representing such set provided, however, the term "software" does not mean or include the medium.

THE FOREGOING WARRANTIES ARE IN LIEU AND EXCLUDE ALL OTHER EXPRESS AND IMPLIED WARRANTIES, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. BUYER'S SOLE AND EXCLUSIVE REMEDY SHALL BE SELLER'S OBLIGATION TO REPAIR OR REPLACE OR CREDIT OR REFUND AS SET FORTH ABOVE.

Hardware Warranty for WE® 321EB Microprocessor Evaluation Board, WE® 321DS Microprocessor Development System, WE® 321DM CPU & MMU Device Monitor, WE® 322DM CPU & MMU Device Monitor

Seller warrants to Buyer that Hardware Products will, at the time of shipment and under proper and normal use, be free from defects in material or workmanship and will conform to Seller's applicable descriptions. If any failure to conform to this WARRANTY appears in any Hardware Product, Seller will, at its option, either repair or replace the defective Product without charge at Seller's manufacturing or repair facility or at Buyer's facility, or will refund the purchase price of the defective Hardware Product, PROVIDED: that Buyer notifies Seller of the purported failure to conform to this WARRANTY within ninety (90) days after the date of shipment of the Hardware Product to Buyer and that Seller's examination of the Hardware Product discloses that the purported failure to conform to this WARRANTY is present. Buyer shall follow Seller's instructions regarding return of Hardware Products, and no Hardware Product will be accepted for repair, replacement or refund absent Buyer's following such instructions. In the case of any such return, Buyer shall bear the risk of loss or damage and shall prepay all transportation charges to Seller. Repaired or replacement Hardware Products shall be shipped to a continental U.S. destination by Seller at its expense and risk of loss or damage. Buyer shall be responsible for any customs duties, local taxes and expenses related to the importation of repair or replacement Hardware Products and any requested premium transportation. Replaced parts shall become Seller's property. Seller shall not be responsible under this WARRANTY for deinstallation or reinstallation or for related expenses arising out of the alteration of Buyer's or a third party's premises or building or removal, replacement or relocation of other items not purchased hereunder. If Seller determines that the Hardware Products do not fail to conform to this WARRANTY, Buyer shall pay Seller all costs of handling, inspection, repairs and transportation at Seller's then prevailing rates.

Repaired and replaced parts provided under the above WARRANTY are warranted as set forth above, but only for sixty (60) days from the date of repair or replacement, or, for the remainder of the WARRANTY period, whichever is greater.

The foregoing WARRANTY does not extend to Software Products; to expendable items; to experimental or developmental Hardware Products; or to Hardware Products which have been subject to misuse, neglect, accident or abuse; have been improperly wired, repaired or altered by anyone other than Seller, have been improperly installed, stored or maintained by anyone other than Seller, have been used in material violation of Seller's instructions; or have had their serial numbers or month and year of manufacture or shipment removed, defaced or altered.

THE FOREGOING WARRANTY IS IN LIEU OF AND EXCLUDES ALL OTHER EXPRESSED AND IMPLIED WARRANTIES, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. BUYER'S SOLE AND EXCLUSIVE REMEDY SHALL BE SELLER'S OBLIGATION TO REPAIR, REPLACE OR REFUND AS SET FORTH ABOVE.

Hardware Warranty for *WE* 321SB VMEbus Single Board Computer

Seller warrants to Buyer that Hardware Products will, at the time of shipment and under proper and normal use, be free from defects in material or workmanship and will conform to Seller's applicable descriptions. If any failure to conform to this WARRANTY appears in any Hardware Product, Seller will, at its option, either repair or replace the defective Product without charge at Seller's manufacturing or repair facility or at Buyer's facility, or will refund the purchase price of the defective Hardware Product, PROVIDED: that Buyer notifies Seller of the purported failure to conform to this WARRANTY within ninety (90) days after the date of shipment of the Hardware Product to Buyer and that Seller's examination of the Hardware Product discloses that the purported failure to conform to this WARRANTY is present. Buyer shall follow Seller's instructions regarding return of Hardware Products, and no Hardware Product will be accepted for repair, replacement or refund absent Buyer's following such instructions. In the case of any such return, Buyer shall bear the risk of loss or damage and shall prepay all transportation charges to Seller. Repaired or replacement Hardware Products shall be shipped to a continental U.S. destination by Seller at its expense and risk of loss or damage. Buyer shall be responsible for any customs duties, local taxes and expenses related to the importation of repair or replacement Hardware Products and any requested premium transportation. Replaced parts shall become Seller's property. Seller shall not be responsible under this WARRANTY for deinstallation or reinstallation or for related expenses arising out of the alteration of Buyer's or a third party's premises or building or removal, replacement or relocation of other items not purchased hereunder. If Seller determines that the Hardware Products do not fail to conform to this WARRANTY, Buyer shall pay Seller all costs of handling, inspection, repairs and transportation at Seller's then prevailing rates.

Repaired and replaced parts provided under the above WARRANTY are warranted as set forth above, but only for thirty (30) days from the date of repair or replacement, or, for the remainder of the WARRANTY period, whichever is greater.

The foregoing WARRANTY does not extend to Software Products; to expendable items; to experimental or developmental Hardware Products; or to Hardware Products which have been subject to misuse, neglect, accident or abuse; have been improperly wired, repaired or altered by anyone other than Seller, have been improperly installed, stored or maintained by anyone other than Seller, have been used in material violation of Seller's instructions; or have had their serial numbers or month and year of manufacture or shipment removed, defaced or altered.

THE FOREGOING WARRANTY IS IN LIEU OF AND EXCLUDES ALL OTHER EXPRESSED AND IMPLIED WARRANTIES, INCLUDING BUT NOT LIMITED TO WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. BUYER'S SOLE AND EXCLUSIVE REMEDY SHALL BE SELLER'S OBLIGATION TO REPAIR, REPLACE OR REFUND AS SET FORTH ABOVE.

Software Product Warranty

Seller warrants to Buyer that, upon shipment, Software Products will be free from defects which result in a material failure of the applicable Hardware Product to execute instructions.

If, under normal and proper licensed use, the Software Products prove to have such a defect and Buyer notifies Seller of such defect within ninety (90) days from the date of shipment thereof, Seller, at its option, will either correct or replace the same without charge at its facility or provide a refund or credit. No Software Product will be accepted for correction or replacement except upon the written authorization and in accordance with instructions of Seller. Any transportation expenses associated with returning such Software Product to Seller shall be borne by Buyer. Seller shall pay the costs of transportation of the corrected or replacement Software Product to the destination designated by Buyer. At Seller's option, correction may be incorporated into a new release of the Software Product which will be made available to Buyer or such corrections may be performed at Buyer's facility.

In the event that Seller determines after investigation that the Software Products are not defective, Buyer shall pay all costs of handling, inspection, testing and transportation, including travel and subsistence costs incurred by Seller's personnel.

Seller makes no warranty as to the following: defects other than those which result in a material failure of the applicable Hardware Product to execute instructions; defects related to Buyer's misuse, neglect, accident or abuse; defects related to Buyer's alteration of software; defects appearing in software used in violation of the license granted hereby that Software Products will meet the requirements of Buyer or that the operation of the Hardware Products using the Software Products will be uninterrupted or error-free.

THE FOREGOING WARRANTY IS EXCLUSIVE AND IS IN LIEU OF ALL OTHER EXPRESSED AND IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, WARRANTIES OR MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. BUYER'S SOLE AND EXCLUSIVE REMEDY SHALL BE SELLER'S OBLIGATION TO CORRECT, REPLACE, CREDIT OR REFUND, AS SET FORTH ABOVE.

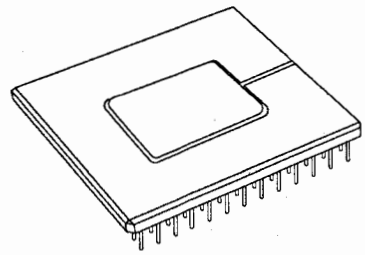
Chapter 2

Devices

WE® 32100 Microprocessor

Description

The *WE* 32100 Microprocessor (CPU) is a high-performance, single-chip, 32-bit central processing unit designed for efficient operation in a high-level language environment. It performs all the system address generation, control, memory access, and processing functions required in a 32-bit microcomputer system. It has separate 32-bit address and data buses. System memory is addressed over the 32-bit address bus by using either physical or virtual addresses. Data is read or written over the 32-bit bidirectional data bus in word (32-bit), halfword (16-bit), or byte (8-bit) widths. Extensive addressing modes result in a symmetric, versatile, and powerful instruction set. The *WE* 32100 Microprocessor is available in 10-, 14-, and 18-MHz versions; requires a single 5 V supply; and is available in a 125-pin square, ceramic pin grid array (PGA) package.



Features

- Powerful and versatile instruction set
- On-chip instruction cache
- 4 Gbytes (2^{32}) of direct memory addressing
- Physical and virtual addressing
- Coprocessor interface
- DMA and multiprocessor environment support
- 15 levels of interrupts
- Autovector and nonmaskable interrupt facilities
- Four levels of execution privilege: kernel, executive, supervisor, and user
- Synchronous or asynchronous interfacing to external devices
- Nine 32-bit general-purpose registers
- Seven 32-bit special-purpose registers
- Memory-mapped I/O
- Complete floating-point support via the *WE* 32106 Math Acceleration Unit

Description 1
Features 1
User Information 3
Architectural Summary 3
Process Status Word 4
Process Control Block 6
Interrupts 6
Bus Arbitration 10
Direct Memory Access and Multiprocessor Support 10
Coprocessor Interface 10
Read-Interlocked Operation 10
Reset 10
Flags/Conditions 11
Instruction Set and Summaries by Mnemonic and Opcode 12
Instruction Set Summary by Functional Group 21
Instruction Timing 21
Arithmetic Group 21
Data Transfer Group 23
Logical Group 24
Program Control Group 25
Stack and Miscellaneous Groups 27
Coprocessor Group Instruction 28
Operating System Instructions and Microsequences 28
Other Microsequences 28
Addressing Modes 30
Exceptions 33
Bus Exceptions 33
Exceptional Conditions 33
Normal Exceptions 33
Stack Exceptions 35
Process Exceptions 35
Reset Exceptions 35
Pin Descriptions 35
Numerical Order 36
Functional Groups 39
Characteristics 45
Timing Characteristics 45
Timing Diagrams 49
Electrical Characteristics 62
Inputs 62
Outputs 62
Operating Conditions 63

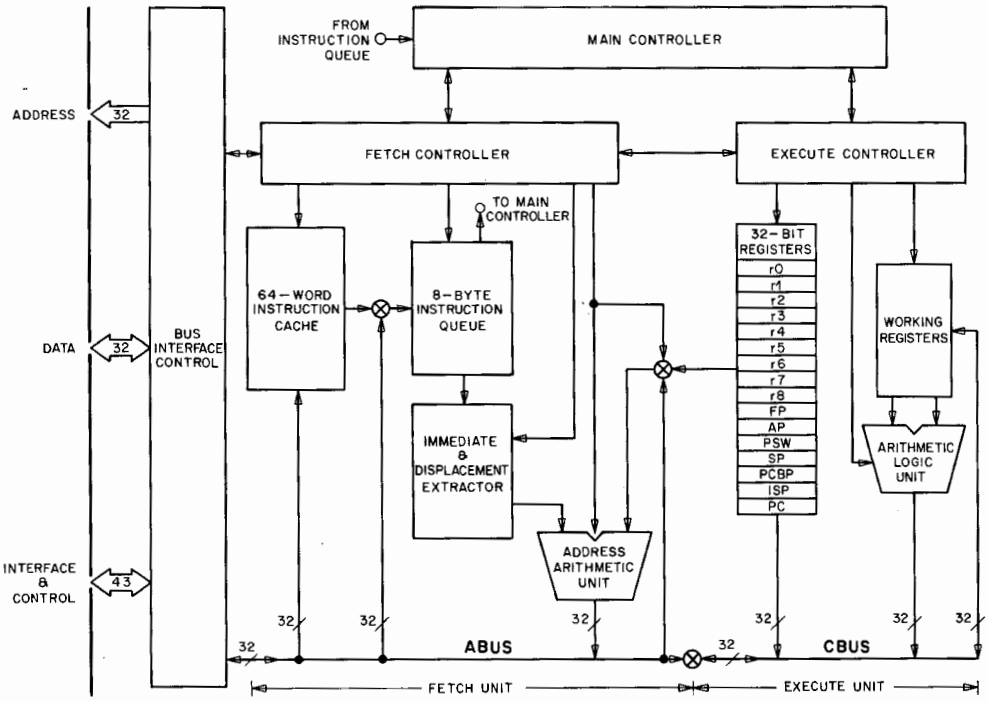


Figure 1. WE® 32100 Microprocessor Block Diagram

User Information

Architectural Summary

The *WE* 32100 Microprocessor provides separate 32-bit address and data buses to the external system, eliminating the need for external multiplexing/demultiplexing. The memory or peripherals mapped into the system memory are addressed over the 32-bit address bus by using either physical or virtual addresses. Data is read to or written from the microprocessor over the 32-bit bidirectional data bus in word (32-bit), halfword (16-bit), or byte (8-bit) widths. The microprocessor automatically extends bytes and halfwords to 32 bits for processing.

Execution speed is enhanced by an internal instruction queue and an internal instruction cache. The instruction queue is an 8-byte first-in-first-out (FIFO) queue that stores prefetched instructions. The queue is used to pipeline instructions, thereby enabling the microprocessor to overlap the execution of instructions while tracking each separately. The instruction cache is a 64-word on-chip cache used to increase the microprocessor's performance by reducing external memory reads for instruction fetches. When an instruction fetch from memory occurs, instruction data is placed in both the instruction queue and the instruction cache. If that instruction data is needed again, it is fetched from the cache rather than from external memory, resulting in increased performance.

The microprocessor uses address and data strobes, as well as other interface and control signals, to control information flow over the system address and data buses. These signals provide timing for the latching of information and facilitate interfacing with commercial memories and peripherals. The microprocessor also accommodates wait-state generation to allow hand-shaking with slow peripherals.

Figure 1 shows the four major sections of the *WE* 32100 Microprocessor: bus interface control, main controller, fetch unit, and execute unit.

Bus Interface Control. Provides all the strobes and control signals necessary to implement the interface with peripherals.

Main Controller. Responsible for acquiring and decoding instruction opcodes and for directing the action of the fetch and execute controllers as the specified instruction is executed. The main controller also responds to and directs the handling of interrupts and exceptional conditions.

Fetch Unit. Handles the instruction stream and performs memory-based operand accesses. It consists of a fetch controller, an instruction cache, an instruction queue, an immediate and displacement extractor, and an address arithmetic unit (AAU).

Execute Unit. Performs all arithmetic and logical operations and all shift and rotate operations and computes condition flags. It consists of an execute controller, sixteen 32-bit registers, working registers, and a 33-bit-wide arithmetic logic unit (ALU). The sixteen 32-bit registers are user-accessible and include nine general-purpose registers (r0—r8) and seven dedicated registers (r9—r15). All of the registers except the program counter (r15) can be referenced in all addressing modes. The processor status word (r11), process control block pointer (r13), and interrupt stack pointer (r14) are privileged registers that can be read at any time, but can be written only when the microprocessor is in the kernel (highest) execution level. The working registers are used exclusively by the microprocessor and are not user-accessible.

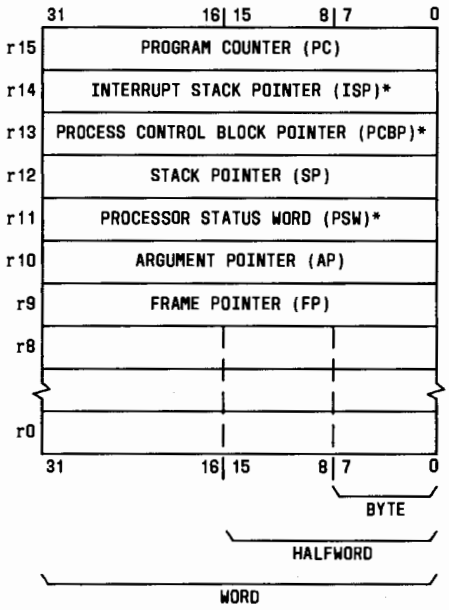
The sixteen 32-bit registers, defined below, are shown on Figure 2.

r0—r8 – General-Purpose Registers. These nine 32-bit registers can be used for accumulation, addressing, or temporary data storage. They can be used in any addressing mode by any program – privileged or nonprivileged. Registers r0, r1, and r2 are also implicitly used by certain data transfer and operating system instructions.

r9 – Frame Pointer (FP). Points to the beginning location in the stack of a function's local variables.

r10 – Argument Pointer (AP). Points to the beginning location in the stack to which a set of arguments for a function have been pushed.

r11 – Processor Status Word (PSW). Contains information that determines the current execution state. This information includes current exception type; an internal state code; the current interrupt priority level; trace enable, trace mask, cache disable, cache flush disable,



* Kernel-level privileged.

Figure 2. CPU Registers

quick-interrupt enable, and enable overflow trap flags; previous and current execution levels; and four condition codes that indicate zero (Z), overflow (V), negative (N), and carry (C). The PSW is kernel-level privileged.

r12 – Stack Pointer (SP). Contains the current 32-bit address of the top of the execution stack, i.e., the memory address of the next place in which an item can be stored (pushed) on the stack or of the last place from which an item was retrieved (popped) from the stack. The stack pointer implements a last-in-first-out (LIFO) queue, which supports efficient subroutine linkage and local variable storage.

r13 – Process Control Block Pointer (PCBP). Contains the 32-bit address of the process control block (PCB) for the current process. The PCBP is kernel-level privileged.

r14 – Interrupt Stack Pointer (ISP). Contains the 32-bit memory address of the top of the interrupt stack. This stack is used when an interrupt request is received and when the call process (CALLPS) and return to process (RETPS) instructions are encountered. The ISP is kernel-level privileged.

r15 – Program Counter (PC). Contains the 32-bit memory address of the instruction being executed or, upon instruction completion, the starting address of the next instruction to be executed.

Processor Status Word (PSW)

The PSW (r11) contains several state variables associated with the current process and provides a means of determining the microprocessor's execution state at any time. The word format for the PSW is shown on Figure 3.

The following define the PSW bit field.

Bits 0, 1 – Exception Type (ET). Bits 0 and 1 represent exceptions generated during operations. This read-only field is interpreted as:

Bit 1	Bit 0	Description
0	0	On reset exception
0	1	On process exception
1	0	On stack exception
1	1	On normal exception

Bit 2 – Trace Mask (TM). Bit 2 enables the masking of a trace trap. This read-only field masks the trace enable (TE) bit for the duration of one instruction in order to avoid a trace trap. The TM bit is set (1) at the beginning of every instruction and cleared (0) as part of each microsequence (see Microsequences) that performs a context switch or a return from gate (RETG), or when any fault or interrupt is detected and responded to.

Bits 3 through 6 – Internal State Code (ISC). Bits 3 through 6 represent the microprocessor internal state code and distinguish between exceptions of the same exception type. This field is used in conjunction with the exception type (ET) field to determine which type of exception occurred. Normal exceptional conditions are decoded on a priority scheme if more than one occurs in a particular cycle. Table 1 lists the ISC values.

Bit	31 26	25	24	23	22	21 18	17	16 13	12 11	10 9	8	7	6 3	2	1 0
Field	Unused	CFD	QIE	CD	OE	NZVC	TE	IPL	CM	PM	R	I	ISC	TM	ET

Figure 3. Word Format for the PSW

Bits 7, 8 – Register-Initial Context (RI). Bits 7 and 8 control the microprocessor context switching strategy. The I bit (bit 7) determines if a process executes from an initial (I=1) or an intermediate saved context. The R bit (bit 8, read only) determines if the registers of a process should be saved during a process switch (saved if R=1; not saved if R=0). It also controls block moves, which can be used to change MMU information.

Table 1. Exceptional Conditions

Exception Type	Exception	Internal State Code Bit 6 5 4 3
Normal exception (ET = 11)	Integer zero-divide	0 0 0 0*
	Trace trap	0 0 0 1
	Illegal opcode	0 0 1 0
	Reserved opcode	0 0 1 1
	Invalid descriptor	0 1 0 0*
	External memory fault	0 1 0 1
	Gate vector fault	0 1 1 0
	Illegal level change	0 1 1 1
	Reserved data type	1 0 0 0*
	Integer overflow	1 0 0 1
	Privileged opcode	1 0 1 0
	Breakpoint trap	1 1 1 0
	Privileged register	1 1 1 1
Stack exception (ET = 10)	Stack bound	0 0 0 0
	Stack fault	0 0 0 1
	Interrupt ID fetch	0 0 1 1
Process exception (ET = 01)	Old PCB fault	0 0 0 0
	Gate PCB fault	0 0 0 1
	New PCB fault	0 1 0 0
Reset exception (ET = 00)	Old PCB fault	0 0 0 0
	System data	0 0 0 1
	Interrupt stack fault	0 0 1 0
	External reset	0 0 1 1
	New PCB fault	0 1 0 0
	Gate vector fault	0 1 1 0

* Exceptional conditions that reset the PSW flags.

Bits 9, 10 – Previous Execution Level (PM). Bits 9 and 10 represent the previous execution level and are interpreted as:

Bit 10	Bit 9	Description
0	0	Kernel level
0	1	Executive level
1	0	Supervisor level
1	1	User level

Bits 11, 12 – Current Execution Level (CM). Bits 11 and 12 represent the current execution level. The code of bits 11 and 12 is interpreted in the same manner as that of bits 9 and 10 (respectively) in the PM code.

Changes to the CM field via instructions that have the PSW as an explicit destination may affect the XMD pins during a prefetch access. Accordingly, only microsequence instructions should be used to change the CM field state.

Bits 13 through 16 – Interrupt Priority Level (IPL). Bits 13 through 16 represent the current interrupt priority level. Fifteen levels of interrupts are available. An interrupt, unless it is a nonmaskable interrupt, must have a higher priority level than the current IPL of the PSW in order to be acknowledged. Level 0000 indicates that any of the fifteen interrupt priority levels (0001 through 1111) can interrupt the microprocessor, while level 1111 indicates that no interrupts except a nonmaskable interrupt can interrupt the microprocessor since 1111 is the highest interrupt priority level.

Bit 17 – Trace Enable (TE). Bit 17 enables the trace function. When set (1), it causes a trace trap to occur after execution of the next instruction. Debugging and analysis software use this facility for single-stepping a program. Changes to the TE field via instructions that have the PSW as an explicit destination may cause unpredictable trace-trap behavior, i.e., the instruction that changed the TE field in the PSW may or may not cause a trace trap. Accordingly, only microsequence instructions should be used to change the TE field state.

Bits 18 through 21 – Condition Codes (NZVC). Bits 18 through 21 represent condition codes. These codes reflect the status of the last instruction execution to affect them. The codes are tested by using the conditional branch instructions and indicate the following when set:

N – Negative (bit 21)
Z – Zero (bit 20)
V – Overflow (bit 19)
C – Carry (bit 18)

Bit 22 – Enable Overflow Trap (DE). Bit 22, when set, enables overflow traps. It is cleared whenever an overflow trap is detected and handled.

Bit 23 – Cache Disable (CD). Bit 23 enables and disables the instruction cache. When the CD bit is cleared, the cache is used to store and read text. When the CD bit is set, the cache is not used. The instruction cache should be disabled only when its use could cause problems, e.g., when self-modifying code is executing. Changes to the CD field via instructions that have the PSW as an explicit destination may corrupt the contents of the instruction cache. Therefore, only microsequence instructions should be used to change the CD field state.

Bit 24 – Quick-Interrupt Enable (QIE). Bit 24 enables and disables the quick-interrupt facility. If QIE is set, an interrupt is handled via the quick- interrupt sequence. If QIE is cleared, the interrupt causes a process switch (full-interrupt sequence).

Bit 25 – Cache Flush Disable (CFD). Bit 25, when set, disables instruction cache flushing (emptying of the cache's contents) when a new process is loaded via the XSWITCH_TWO microsequence (see Microsequences). When cleared, the contents of the cache are flushed when a new process is loaded via the XSWITCH_TWO microsequence.

Bits 26 through 31 – Unused. Bits 26 through 31 are not used and must always be cleared.

Process Control Block (PCB)

The PCB contains the entire switchable process context, collected into a compact form for ease of movement between system memory and privileged internal registers. The process control block pointer (PCBP), r13, contains the address of the PCB currently in execution. The PCB is shown on Figure 4 and a summary of its contents appears in Table 2.

Interrupts

The microprocessor accepts fifteen levels of interrupt requests from a 4-bit interrupt request input port (IPL0—IPL3); it also accepts a nonmaskable interrupt if $\overline{\text{NMINT}}$ is externally asserted. The pending interrupt value input on IPL0—IPL3 is internally inverted and compared to the value contained in the interrupt priority level (IPL) field of the PSW. In order for the pending interrupt to be acknowledged, its inverted value must be greater than the IPL field value. Pending interrupts whose inverted values are less than or equal to the IPL field value are ignored. The exception to this is a nonmaskable interrupt, which can interrupt the microprocessor regardless of the present interrupt priority level.

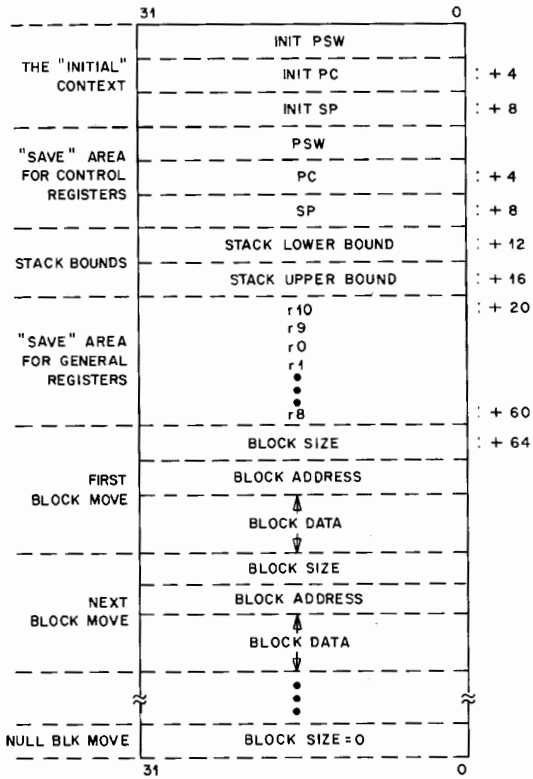


Figure 4. Process Control Block

The microprocessor acknowledges an interrupt by transmitting the inverted interrupt value on bits 2 through 5 of the address bus. In addition, the value placed on the interrupt option ($\overline{\text{INTOPT}}$) pin is inverted and transmitted on bit 6 of the address bus. The microprocessor then fetches the interrupt vector number from the interrupting device on bits 0 through 7 of the data bus and begins execution of the interrupt handling routine.

Table 3 shows the interrupt priority levels and the corresponding interrupt acknowledges output by the microprocessor.

Autovector, nonmaskable-interrupt, and quick-interrupt facilities are also available. The microprocessor provides the vector number for the interrupting device if the autovector input ($\overline{\text{AVEC}}$) is asserted at the same time as the interrupt request. The vector number is dependent on the interrupt's priority level. If the nonmaskable interrupt ($\overline{\text{NMINT}}$) input is asserted, the interrupt occurs but is treated as an autovector interrupt with the vector number 0. During the acknowledge cycle of a nonmaskable interrupt, the microprocessor address bus contains all zeros. This distinguishes a nonmaskable interrupt from all other interrupts. A nonmaskable interrupt can interrupt the microprocessor regardless of the current priority. Interrupts are handled via the quick-interrupt sequence if the quick-interrupt enable (QIE) bit in the PSW is set. Table 4 summarizes how the microprocessor handles the various interrupt requests.

Table 2. Process Control Block Section

PCB Section	Description
Initial context	This section contains a set of initial values for the PSW, PC, and SP. This allows a process to be restarted with its initial values even though, at some point, its execution is suspended.
Save area for control registers	This section contains the PSW, PC, and SP values at the time the current process is switched.
Stack bounds	This section defines the upper and lower bounds of the stack used in the current process. It contains pointers to the upper and lower memory limits of the stack.
Save area for general registers	This section contains images of what registers 0 through 10 (r0—r10) were when the process was switched. This allows the process to continue in the same state that it was in when control switched to another process.
First block move	Moves data in PCB to an arbitrary memory region.
Next & nth block move	Moves a different set of data in PCB to a different memory region.
Null block move	Terminates block move sequence.

Table 3. Interrupt Level Code Assignments

Interrupt Request Input IPL0—IPL3				Interrupt Option Input $\overline{\text{INTOPT}}$	Interrupt Acknowledge Output ADDR02—ADDR06					Priority Level in Descending Order
Bits					Bits					
3	2	1	0		06	05	04	03	02	
0	0	0	0	0	1	1	1	1	1	Highest priority
0	0	0	0	1	0	1	1	1	1	
0	0	0	1	0	1	1	1	1	0	2nd
0	0	0	1	1	0	1	1	1	0	
0	0	1	0	0	1	1	1	0	1	3rd
0	0	1	0	1	0	1	1	0	1	
0	0	1	1	0	1	1	1	0	0	4th
0	0	1	1	1	0	1	1	0	0	
0	1	0	0	0	1	1	0	1	1	5th
0	1	0	0	1	0	1	0	1	1	
0	1	0	1	0	1	1	0	1	0	6th
0	1	0	1	1	0	1	0	1	0	
0	1	1	0	0	1	1	0	0	1	7th
0	1	1	0	1	0	1	0	0	1	
0	1	1	1	0	1	1	0	0	0	8th
0	1	1	1	1	0	1	0	0	0	
1	0	0	0	0	1	0	1	1	1	9th
1	0	0	0	1	0	0	1	1	1	
1	0	0	1	0	1	0	1	1	0	10th
1	0	0	1	1	0	0	1	1	0	
1	0	1	0	0	1	0	1	0	1	11th
1	0	1	0	1	0	0	1	0	1	
1	0	1	1	0	1	0	1	0	0	12th
1	0	1	1	1	0	0	1	0	0	
1	1	0	0	0	1	0	0	1	1	13th
1	1	0	0	1	0	0	0	1	1	
1	1	0	1	0	1	0	0	1	0	14th
1	1	0	1	1	0	0	0	1	0	
1	1	1	0	0	1	0	0	0	1	Lowest priority
1	1	1	0	1	0	0	0	0	1	
1	1	1	1	0	x	x	x	x	x	No interrupt pending
1	1	1	1	1	x	x	x	x	x	

Note: x = no value placed on address bus.

Table 4. Interrupt Acknowledge Summary

Priority of Interrupt	Interrupt Acknowledge	$\overline{\text{AVEC}}$	$\overline{\text{NMINT}}$	QIE	Result
Less than PSW IPL field priority	No	x	1	x	Interrupt is not acknowledged.
Equal to PSW IPL field priority	No	x	1	x	Interrupt is not acknowledged.
Greater than PSW IPL field priority	Yes	1	1	0	Interrupt is acknowledged via full-interrupt sequence. Microprocessor fetches vector number from interrupting device.
Greater than PSW IPL field priority	Yes	0	1	0	Interrupt is acknowledged via full-interrupt sequence. Microprocessor supplies the vector number.
Any level compared to PSW IPL field priority	Yes	x	0	0	Interrupt is acknowledged via full-interrupt sequence. It is treated as an autovector at vector number 0. The address bus contains all zeros during the acknowledge.
Greater than PSW IPL field priority	Yes	1	1	1	Interrupt is acknowledged and serviced via quick-interrupt sequence. Microprocessor fetches vector number from interrupting device.
Greater than PSW IPL field priority	Yes	0	1	1	Interrupt is acknowledged and serviced via quick-interrupt sequence. Microprocessor supplies the vector number.
Any level compared to PSW IPL field priority	Yes	x	0	1	Interrupt is acknowledged and serviced via quick-interrupt sequence. It is treated as an autovector interrupt at vector number 0. The address bus contains all zeros during the acknowledge.

Note: x = don't care.

Bus Arbitration

The microprocessor arbitrates requests for its bus (relinquish and retry request, $\overline{\text{RRREQ}}$, an bus request, $\overline{\text{BUSRQ}}$) in the following manner. An $\overline{\text{RRREQ}}$ is acknowledged only during a bus transaction; however, it is ignored during the write portion of a read-interlocked transaction. An $\overline{\text{RRREQ}}$ has priority over a $\overline{\text{BUSRQ}}$. A $\overline{\text{BUSRQ}}$ is not acknowledged until the end of a bus transaction or until the end of the write portion of a read-interlocked transaction. A bus request is acknowledged immediately if the bus is idle.

Direct Memory Access and Multiprocessor Support

The microprocessor provides support for direct memory access (DMA) and shares bus control responsibilities with the system DMA controller. To initiate a DMA operation, the controller must request the microprocessor bus by asserting the $\overline{\text{BUSRQ}}$ input. Once the microprocessor recognizes the request, it drives its outputs to the states given in Table 5. The microprocessor then acknowledges the DMA request by asserting the bus request acknowledge ($\overline{\text{BRACK}}$) output. Upon completion of the DMA operation, the 3-stated signals are returned to the microprocessor's control.

The microprocessor provides for the support of multiprocessors via the $\overline{\text{BARB}}$ pin. When this signal is asserted, the microprocessor assumes the role of bus arbiter and other devices request the bus from the microprocessor. When the signal is not asserted, the microprocessor must request the bus from an external bus arbiter. The ability to reassign the bus arbiter provides the capability to have a multiprocessor system.

Coprocessor Interface

The coprocessor interface consists of ten instructions and the associated pinout and protocol. This interface is provided to insure high performance of coprocessors, thus increasing overall system throughput.

Read-Interlocked Operation

The microprocessor provides an interlocked operation that prevents another processor from accessing a memory location between the read and write halves of the operation. Interlocked operation consists of a memory fetch (read access) and one or more internal microprocessor operations, followed by a write access to the same memory location. Once the read access has been completed, the interlocked operation may not be preempted other than by a reset.

Table 5. Output States After DMA Request Acknowledge

Output Signal	Signal State
$\overline{\text{ABORT}}$	Z'
ADDR00—ADDR31	Z
$\overline{\text{AS}}$	Z'
$\overline{\text{BRACK}}$	Logic 0
$\overline{\text{CYCLEI}}$	Z'
DATA00—DATA31	Z
$\overline{\text{DRDY}}$	Z'
$\overline{\text{DS}}$	Z'
DSIZE0, DSIZE1	Z
R/$\overline{\text{W}}$	Z'
$\overline{\text{RESET}}$	Logic 1
$\overline{\text{RRRACK}}$	Logic 1
SAS0—SAS3	Z'
$\overline{\text{VAD}}$	Z
XMD0, XMD1	Z

Notes:
Z = High-impedance.
Z' = High-impedance. Normally tied high with external passive holding resistor.

Reset

Two types of reset requests are available — system-initiated and microprocessor-initiated. Both requests have highest priority and preempt any ongoing microprocessor operation. A system-initiated reset differs from a microprocessor-initiated reset only because it is externally generated. The system initiates a reset by driving the reset request ($\overline{\text{RESETR}}$) input low. Once the microprocessor recognizes the request, it sends a reset acknowledge ($\overline{\text{RESET}}$) to the system and drives its outputs to a temporary state, which prevents control signal and bus conflicts while the system is responding to the acknowledge. The system negates $\overline{\text{RESETR}}$ once it has responded to the acknowledge, but the

microprocessor continues to hold $\overline{\text{RESET}}$ active for an additional 128 clock cycles, allowing the external system to go through its own initialization sequence. At the end of this period, the microprocessor negates $\overline{\text{RESET}}$ and begins execution of the internal reset sequence. Table 6 indicates the states of the microprocessor's output pins once it has negated $\overline{\text{RESET}}$. The microprocessor must perform the following register initialization sequence before operations begin again (restart):

- Initiate physical addressing mode.
- Fetch a word at location 80 (hexadecimal) and put it into the process control block pointer (r13). This word is the beginning address of the reset process control block (PCB).
- Fetch a word at the PCB address and store it in the PSW.
- Fetch a word at the location four bytes from the initial PCB address and store it in the program counter (PC). This word is the PC value for initial execution.
- Fetch word at the location eight bytes from the initial PCB address and store it in the stack pointer (SP).

The microprocessor then begins execution at the address specified by the PC.

Flags/Conditions

The processor status word (r11) contains four flag bits – negative (N), zero (Z), overflow (V), and carry/borrow (C). For most instructions, these flags are set according to standard criteria. In the following flag descriptions, the term *logical instruction* refers to any nonarithmetic instruction. *Infinite-precision result* refers to what the result would be if the operation were done in an *infinite precision* machine. The type (signed or unsigned) of the result is determined by the destination operand. *Requested representation* refers to the actual data that is written into the desired location. The representation depends on the site requested (byte, halfword, or word) and whether the result is signed or unsigned. The flag bits are set as follows:

N – Negative. The negative flag is set if the sign of the infinite-precision result is negative for nonlogical instructions. Zero is considered positive in this case. For logical instructions, the negative flag is the most significant bit (bit 31 for words, bit 15 for halfwords, or bit 7 for bytes) of the requested representation. Note that this bit is valid even in unsigned operations.

Z – Zero. For nonlogical instructions, the zero flag is set if the infinite-precision result is equal to zero. For logical instructions, it is set if the requested representation is equal to zero.

V – Overflow. For arithmetic operations with signed destination, the V bit is set if any truncated bit of the infinite-precision result is different from the sign bit, which is the most significant bit, of the requested representation. For arithmetic operations with an unsigned destination, the V bit is set if any truncated bit is nonzero. For arithmetic left-shift operations,

Table 6. Output States on Reset

Output Signal	Signal State[1]	
	CPU[2] is Bus Arbiter[3]	CPU is not Bus Arbiter
$\overline{\text{ABORT}}$	Logic 1	Z
ADDR00—ADDR31	Z	Z
$\overline{\text{AS}}$	Logic 1	Z
$\overline{\text{BRACK}}$	Logic 1	—
$\overline{\text{BUSRQ}}$	—	Logic 1
$\overline{\text{CYCLEI}}$	Logic 1	Z
DATA00—DATA31	Z	Z
$\overline{\text{DRDY}}$	Logic 1	Z
$\overline{\text{DS}}$	Logic 1	Z
DSIZE0, DSIZE1	Logic 0	Z
IQS0, IQS1	Logic 1	Logic 1
R/$\overline{\text{W}}$	Logic 1	Z
$\overline{\text{RRRACK}}$	Z[4]	Z[4]
SAS0—SAS3	Logic 1	Z
$\overline{\text{SOI}}$	Logic 1	Logic 1
$\overline{\text{VAD}}$	Note 5	Z
XMD0, XMD1	Note 6	Z

Notes:
[1] Z = High-impedance.
[2] CPU is *WE* 32100 Microprocessor.
[3] Determined by state of $\overline{\text{BARB}}$ input pin.
[4] Open drain output not actively driven under this condition.
[5] Not guaranteed to be logic 1 (i.e., physical mode) until approximately 38 clock cycles after $\overline{\text{RESET}}$ is negated.
[6] Not guaranteed to be in kernel mode until approximately 18 clock cycles after $\overline{\text{RESET}}$ is negated.

bits are not shifted past bit 31 of the result, so the V bit is set only if there is a truncation error. For logical instructions with a signed destination, the V bit is set if any truncated bit of the 32-bit result is different from the sign bit of the requested representation. For logical instructions with an unsigned destination, the V bit is set if any truncated bit is nonzero. Instructions that do not cause these conditions reset the V bit.

C – Carry/Borrow. The carry bit is the carry into the 33rd bit position (bit 32) for word operations, the 17th bit position (bit 16) for halfword operations, or the 9th bit position (bit 8) for byte operations. For subtracts, negates, decrements, and compares, it is the borrow from the 33rd, 17th, or 9th bit position. For example, consider A–B where A and B are unsigned. If A $\geq$ B after both A and B are extended to 32 bits, then C is cleared. Otherwise, the C flag is set. For logical instructions, the carry is cleared.

Instruction Set and Summaries by Mnemonic and Opcode

The *WE* 32100 Microprocessor has an extensive instruction set, which includes special coprocessor instructions that implement interfacing with coprocessors. Instructions may appear at any byte address in memory and consist of a one- or two-byte operation code (opcode) followed by zero or more operand descriptors. The opcode specifies the operation to be performed and the operand descriptors provide the information needed to locate the operand. The microprocessor supports four classes of instructions:

Monadic Instructions. These instructions contain one operand, which serves as source, destination, or both, e.g., clear and increment.

Dyadic Instructions. These instructions contain two operands. In one type, both operands are designated as sources, as is the case for the compare instruction. The contents of the operands will not be altered. A second type of dyadic instruction designates one of the operands as the source and the other as the destination, e.g., move instructions. A third type of dyadic instruction designates one of the operands as the source and the other as both source and destination, e.g., two-address arithmetic and logical instructions. In this case, the result of the operation is stored in the destination.

Triadic Instructions. These instructions contain three operands: two designated as sources (the first and second), and one (the third) as the destination. The result of the operation is stored in the destination, e.g., three-address arithmetic and logical operations.

Program Control and Miscellaneous Instructions. These instructions alter the order of instruction execution and have operands that are not described by operand descriptors. Program control instructions are of two types: conditional, where the operation performed is based on the result of a test, e.g., conditional branch and conditional return instructions; and unconditional, where the operation is always performed, e.g., jump and call instructions. Cache flush (CFLUSH) is an example of an instruction with no operand descriptor.

Table 7 lists the instruction set mnemonics alphabetically. The hexadecimal opcode and a brief description of the instruction are presented for each mnemonic. Opcodes are either one or two bytes, depending on the instruction. Table 8 lists instructions by opcode and Tables 9—15 by functional group.

Table 7. Instruction Set Summary by Mnemonic

Mnemonic	Opcode	Instruction
ADDB2	0x9F	Add byte
ADDB3	0xDF	Add byte, 3-address
ADDH2	0x9E	Add halfword
ADDH3	0xDE	Add halfword, 3-address
ADDW2	0x9C	Add word
ADDW3	0xDC	Add word, 3-address
ALSW3	0xC0	Arithmetic left shift word
ANDB2	0xBB	AND byte
ANDB3	0xFB	AND byte, 3-address

Table 7. Instruction Set Summary by Mnemonic (Continued)

Mnemonic	Opcode	Instruction
ANDH2	0xBA	AND halfword
ANDH3	0xFA	AND halfword, 3-address
ANDW2	0xB8	AND word
ANDW3	0xF8	AND word, 3-address
ARSB3	0xC7	Arithmetic right shift byte
ARSH3	0xC6	Arithmetic right shift halfword
ARSW3	0xC4	Arithmetic right shift word
BCCB	0x53*	Branch on carry clear byte
BCCH	0x52*	Branch on carry clear halfword
BCSB	0x5B*	Branch on carry set byte
BCSH	0x5A*	Branch on carry set halfword
BEB	0x6F	Branch on equal byte (duplicate)
BEB	0x7F	Branch on equal byte
BEH	0x6E	Branch on equal halfword (duplicate)
BEH	0x7E	Branch on qual halfword
BGB	0x47	Branch on greater than byte (signed)
BGEB	0x43	Branch on greater than or equal byte (signed)
BGEH	0x42	Branch on greater than or equal halfword (signed)
BGEUB	0x53*	Branch on greater than or equal byte (unsigned)
BGEUH	0x52*	Branch on greater than or equal halfword (unsigned)
BGH	0x46	Branch on greater than halfword (signed)
BGUB	0x57	Branch on greater than byte (unsigned)
BGUH	0x56	Branch on greater than halfword (unsigned)
BITB	0x3B	Bit test byte
BITH	0x3A	Bit test halfword
BITW	0x38	Bit test word
BLB	0x4B	Branch on less than byte (signed)
BLEB	0x4F	Branch on less than or equal byte (signed)
BLEH	0x4E	Branch on less than or equal halfword (signed)
BLEUB	0x5F	Branch on less than or equal byte (unsigned)
BLEUH	0x5E	Branch on less than or equal halfword (unsigned)
BLH	0x4A	Branch on less than halfword (signed)
BLUB	0x5B*	Branch on less than byte (unsigned)
BLUH	0x5A*	Branch on less than halfword (unsigned)
BNEB	0x67	Branch on not equal byte (duplicate)
BNEB	0x77	Branch on not equal byte
BNEH	0x66	Branch on not equal halfword (duplicate)
BNEH	0x76	Branch on not equal halfword
BPT	0x2E	Breakpoint trap
BRB	0x7B	Branch with byte displacement
BRH	0x7A	Branch with halfword displacement
BSBB	0x37	Branch to subroutine, byte displacement
BSBH	0x36	Branch to subroutine, halfword displacement
BVCB	0x63	Branch on overflow clear, byte displacement
BVCH	0x62	Branch on overflow clear, halfword displacement

* Opcode matches another instruction and the operation performed is identical.

Table 7. Instruction Set Summary by Mnemonic (Continued)

Mnemonic	Opcode	Instruction
BVSB	0x6B	Branch on overflow set, byte displacement
BVSH	0x6A	Branch on overflow set, halfword displacement
CALL	0x2C	Call procedure
CALLPS**	0x30AC	Call process
CFLUSH	0x27	Cache flush
CLRB	0x83	Clear byte
CLRH	0x82	Clear halfword
CLRW	0x80	Clear word
CMPB	0x3F	Compare byte
CMPH	0x3E	Compare halfword
CMPW	0x3C	Compare word
DECB	0x97	Decrement byte
DECH	0x96	Decrement halfword
DECW	0x94	Decrement word
DISVJMP**	0x3013	Disable virtual pin and jump
DIVB2	0xAF	Divide byte
DIVB3	0xEF	Divide byte, 3-address
DIVH2	0xAE	Divide halfword
DIVH3	0xEE	Divide halfword, 3-address
DIVW2	0xAC	Divide word
DIVW3	0xEC	Divide word, 3-address
ENBVJMP**	0x300D	Enable virtual pin and jump
EXTFB	0xCF	Extract field byte
EXTFH	0xCE	Extract field halfword
EXTFW	0xCC	Extract field word
EXTOP	0x14	Extended opcode
GATE**	0x3061	Gate
INCB	0x93	Increment byte
INCH	0x92	Increment halfword
INCW	0x90	Increment word
INSFB	0xCB	Insert field byte
INSFH	0xCA	Insert field halfword
INSFW	0xC8	Insert field word
INTACK**	0x302F	Interrupt acknowledge
JMP	0x24	Jump
JSB	0x34	Jump to subroutine
LLSB3	0xD3	Logical left shift byte
LLSH3	0xD2	Logical left shift halfword
LLSW3	0xD0	Logical left shift word
LRSW3	oxD4	Logical right shift word
MCOMB	0x8B	Move complemented byte
MCOMH	0x8A	Move complemented halfword
MCOMW	0x88	Move complemented word

* Opcode matches another instruction and the operation performed is identical.
** Operating system instruction.

Table 7. Instruction Set Summary by Mnemonic (Continued)

Mnemonic	Opcode	Instruction
MNEGB	0x8F	Move negated byte
MNEGH	0x8E	Move negated halfword
MNEGW	0x8C	Move negated word
MODB2	0xA7	Modulo byte
MODB3	0xE7	Modulo byte, 3-address
MODH2	0xA6	Modulo halfword
MODH3	0xE6	Modulo halfword, 3-address
MODW2	0xA4	Modulo word
MODW3	0xE4	Modulo word, 3-address
MOVAW	0x04	Move address (word)
MOVB	0x87	Move byte
MOVBLW	0x3019	Move block of words
MOVH	0x86	Move halfword
MOVTRW**	0x0C	Move translated word
MOVW	0x84	Move word
MULB2	0xAB	Multiply byte
MULB3	0xEB	Multiply byte, 3-address
MULH2	0xAA	Multiply halfword
MULH3	0xEA	Multiply halfword, 3-address
MULW2	0xA8	Multiply word
MULW3	0xE8	Multiply word, 3-address
MVERNO	0x3009	Move version number
NOP	0x70	No operation, 1 byte
NOP2	0x73	No operation, 2 byte
NOP3	0x72	No operation, 3 byte
ORB2	0xB3	OR byte
ORB3	0xF3	OR byte, 3-address
ORH2	0xB2	OR halfword
ORH3	0xF2	OR halfword, 3-address
ORW2	0xB0	OR word
ORW3	0xF0	OR word, 3-address
POPW	0x20	Pop word
PUSHAW	0xE0	Push address word
PUSHW	0xA0	Push word
RCC	0x50*	Return on carry clear
RCS	0x58*	Return on carry set
REQLU	0x6C*	Return on equal (unsigned)
REQL	0x7C*	Return on equal (signed)
RESTORE	0x18	Restore registers
RET	0x08	Return from procedure
RETG**	0x3045	Return from gate
RETPS**	0x30C8	Return to process
RGEQ	0x40	Return on greater than or equal (signed)
RGEQU	0x50*	Return on greater than or equal (unsigned)

* Opcode matches another instruction and the operation performed is identical.
** Operating system instruction.

Table 7. Instruction Set Summary by Mnemonic (Continued)

Mnemonic	Opcode	Instruction
RLSS	0x48	Return on less than (signed)
RLSSU	0x58*	Return on less than (unsigned)
RNEQU	0x64*	Return on not equal (unsigned)
RNEQ	0x74*	Return on not equal (signed)
ROTW	0xD8	Rotate word
RSB	0x78	Return from subroutine
RVC	0x60	Return on overflow clear
RVS	0x68	Return on overflow set
SAVE	0x10	Save registers
SPOP	0x32	Coprocessor operation
SPOPD2	0x03	Coprocessor operation double, 2-address
SPOPRD	0x02	Coprocessor operation read double
SPOPRS	0x22	Coprocessor operation read single
SPOPRT	0x06	Coprocessor operation read triple
SPOPS2	0x23	Coprocessor operation single, 2-address
SPOPT2	0x07	Coprocessor operation triple, 2-address
SPOPWD	0x13	Coprocessor operation write double
SPOPWS	0x33	Coprocessor operation write single
SPOPWT	0x17	Coprocessor operation write triple
STRCPY	0x3035	String copy
STREND	0x301F	String end
SUBB2	0xBF	Subtract byte
SUBB3	0xFF	Subtract byte, 3-address
SUBH2	0xBE	Subtract halfword
SUBH3	0xFE	Subtract halfword, 3-address
SUBW2	0xBC	Subtract word
SUBW3	0xFC	Subtract word, 3-address
SWAPBI	0x1F	Swap byte interlocked
SWAPHI	0x1E	Swap halfword interlocked
SWAPWI	0x1C	Swap word interlocked
TSTB	0x2B	Test byte
TSTH	0x2A	Test halfword
TSTW	0x28	Test word
WAIT**	0x2F	Wait for interrupt
XORB2	0xB7	Exclusive OR byte
XORB3	0xF7	Exclusive OR byte, 3-address
XORH2	0xB6	Exclusive OR halfword
XORH3	0xF6	Exclusive OR halfword, 3-address
XORW2	0xB4	Exclusive OR word
XORW3	0xF4	Exclusive OR word, 3-address

* Opcode matches another instruction and the operation performed is identical.
** Operating system instruction.

Table 8. Instruction Set Summary by Opcode

Opcode	Mnemonic	Instruction
0x02	SPOPRD	Coprocessor operation read double
0x03	SPOPD2	Coprocessor operation double, 2-address
0x04	MOVAW	Move address (word)
0x06	SPOPRT	Coprocessor operation read triple
0x07	SPOPT2	Coprocessor operation triple, 2-address
0x08	RET	Return from procedure
0x0C	MOVTRW**	Move translated word
0x10	SAVE	Save registers
0x13	SPOPWD	Coprocessor operation write double
0x14	EXTOP	Extended opcode
0x17	SPOPWT	Coprocessor operation write triple
0x18	RESTORE	Restore registers
0x1C	SWAPWI	Swap word interlocked
0x1E	SWAPHI	Swap halfword interlocked
0x1F	SWAPBI	Swap byte interlocked
0x20	POPW	Pop word
0x22	SPOPRS	Coprocessor operation read single
0x23	SPOPS2	Coprocessor operation single, 2-address
0x24	JMP	Jump
0x27	CFLUSH	Cache Flush
0x28	TSTW	Test word
0x2A	TSTH	Test halfword
0x2B	TSTB	Test byte
0x2C	CALL	Call procedure
0x2E	BPT	Breakpoint trap
0x2F	WAIT**	Wait for interrupt
0x3009	MVERNO	Move version number
0x300D	ENBVJMP**	Enable virtual pin and jump
0x3013	DISVJMP**	Disable virtual pin and jump
0x3019	MOVBLW	Move block of words
0x301F	STREND	String end
0x302F	INTACK**	Interrupt acknowledge
0x3035	STRCPY	String copy
0x3045	RETG**	Return from gate
0x3061	GATE**	Gate
0x30AC	CALLPS**	Call process
0x30C8	RETPS**	Return to process
0x32	SPOP	Coprocessor operation
0x33	SPOPWS	Coprocessor operation write single
0x34	JSB	Jump to subroutine
0x36	BSBH	Branch to subroutine, halfword displacement
0x37	BSBB	Branch to subroutine, byte displacement
0x38	BITW	Bit test word
0x3A	BITH	Bit test halfword
0x3B	BITB	Bit test byte

** Operating system instruction.

Table 8. Instruction Set Summary by Opcode (Continued)

Opcode	Mnemonic	Instruction
0x3C	CMPW	Compare word
0x3E	CMPH	Compare halfword
0x3F	CMPB	Compare byte
0x40	RGEQ	Return on greater than or equal (signed)
0x42	BGEH	Branch on greater than or equal halfword (signed)
0x43	BGEB	Branch on greater than or equal byte (signed)
0x44	RGTR	Return on greater than (signed)
0x46	BGH	Branch on greater than halfword (signed)
0x47	BGB	Branch on greater than byte (signed)
0x48	RLSS	Return on less than (signed)
0x4A	BLH	Branch on less than halfword (signed)
0x4B	BLB	Branch on less than byte (signed)
0x4C	RLEQ	Return on less than or equal (signed)
0x4E	BLEH	Branch on less than or equal halfword (signed)
0x4F	BLEB	Branch on less than or equal byte (signed)
0x50*	RCC	Return on carry clear
0x50*	RGEQU	Return on greater than or equal (unsigned)
0x52*	BCCH	Branch on carry clear halfword
0x52*	BGEUH	Branch on greater than or equal halfword (unsigned)
0x53*	BCCB	Branch on carry clear byte
0x53*	BGEUB	Branch on greater than or equal byte (unsigned)
0x54	RGTRU	Return on greater than (unsigned)
0x56	BGUH	Branch on greater than halfword (unsigned)
0x57	BGUB	Branch on greater than byte (unsigned)
0x58*	RCS	Return on carry set
0x58*	RLSSU	Return on less than (unsigned)
0x5A*	BCSH	Branch on carry set halfword
0x5A*	BLUH	Branch on less than halfword (unsigned)
0x5B*	BCSB	Branch on carry set byte
0x5B*	BLUB	Branch on less than byte (unsigned)
0x5C	RLEQU	Return on less than or equal (unsigned)
0x5E	BLEUH	Branch on less than or equal halfword (unsigned)
0x5F	BLEUB	Branch on less than or equal byte (unsigned)
0x60	RVC	Return on overflow clear
0x62	BVCH	Branch on overflow clear, halfword displacement
0x63	BVCB	Branch on overflow clear, byte displacement
0x64*	RNEQU	Return on not equal (unsigned)
0x66	BNEH	Branch on not equal halfword (duplicate)
0x67	BNEB	Branch on not equal byte (duplicate)
0x68	RVS	Return on overflow set
0x6A	BVSH	Branch on overflow set, halfword displacement
0x6B	BVSB	Branch on overflow set, byte displacement
0x6C*	REQLU	Return on equal (unsigned)
0x6E	BEH	Branch on equal halfword (duplicate)
0x6F	BEB	Branch on equal byte (duplicate)
0x70	NOP	No operation, 1 byte
0x72	NOP3	No operation, 3 byte

* Opcode matches another instruction and the operation performed is identical.

Table 8. Instruction Set Summary by Opcode (Continued)

Opcode	Mnemonic	Instruction
0x73	NOP2	No operation, 2 byte
0x74*	RNEQ	Return on not equal (signed)
0x76	BNEH	Branch on not equal halfword
0x77	BNEB	Branch on not equal byte
0x78	RSB	Return from subroutine
0x7A	BRH	Branch with halfword displacement
0x7B	BRB	Branch with byte displacement
0x7C*	REQL	Return on equal (signed)
0x7E	BEH	Branch on equal halfword
0x7F	BEB	Branch on equal byte
0x80	CLRW	Clear word
0x82	CLRH	Clear halfword
0x83	CLRB	Clear byte
0x84	MOVW	Move word
0x86	MOVH	Move halfword
0x87	MOVB	Move byte
0x88	MCOMW	Move complemented word
0x8A	MCOMH	Move complemented halfword
0x8B	MCOMB	Move complemented byte
0x8C	MNEGW	Move negated word
0x8E	MNEGH	Move negated halfword
0x8F	MNEGB	Move negated byte
0x90	INCW	Increment word
0x92	INCH	Increment halfword
0x93	INCB	Increment byte
0x94	DECW	Decrement word
0x96	DECH	Decrement halfword
0x97	DECB	Decrement byte
0x9C	ADDW2	Add word
0x9E	ADDH2	Add halfword
0x9F	ADDB2	Add byte
0xA0	PUSHW	Push word
0xA4	MODW2	Modulo word
0xA6	MODH2	Modulo halfword
0xA7	MODB2	Modulo byte
0xA8	MULW2	Multiply word
0xAA	MULH2	Multiply halfword
0xAB	MULB2	Multiply byte
0xAC	DIVW2	Divide word
0xAE	DIVH2	Divide halfword
0xAF	DIVB2	Divide byte
0xB0	ORW2	OR word
0xB2	ORH2	OR halfword
0xB3	ORB2	OR byte
0xB4	XORW2	Exclusive OR word
0xB6	XORH2	Exclusive OR halfword
0xB7	XORB2	Exclusive OR byte

* Opcode matches another instruction and the operation performed is identical.

Table 8. Instruction Set Summary by Opcode (Continued)

Opcode	Mnemonic	Instruction
0xB8	ANDW2	AND word
0xBA	ANDH2	AND halfword
0xBB	ANDB2	AND byte
0xBC	SUBW2	Subtract word
0xBE	SUBH2	Subtract halfword
0xBF	SUBB2	Subtract byte
0xC0	ALSW3	Arithmetic left shift word
0xC4	ARSW3	Arithmetic right shift word
0xC6	ARSH3	Arithmetic right shift halfword
0xC7	ARSB3	Arithmetic right shift byte
0xC8	INSFW	Arithmetic left shift word
0xCA	INSFH	Insert field halfword
0xCB	INSFB	Insert field byte
0xCC	EXTFW	Extract field word
0xCE	EXTFH	Extract field halfword
0xCF	EXTFB	Extract field byte
0xD0	LLSW3	Logical left shiftword
0xD2	LLSH3	Logical left shift halfword
0xD3	LLSB3	Logical left shift byte
0xD4	LRSW3	Logical right shift word
0xD8	ROTW	Rotate word
0xDC	ADDW3	Add word, 3-address
0xDE	ADDH3	Add halfword, 3-address
0xDF	ADDB3	Add byte, 3-address
0xE0	PUSHAW	Push address word
0xE4	MODW3	Modulo word, 3-address
0xE6	MODH3	Modulo halfword, 3-address
0xE7	MODB3	Modulo byte, 3-address
0xE8	MULW3	Multiply word, 3-address
0xEA	MULH3	Multiply halfword, 3-address
0xEB	MULB3	Multiply byte, 3-address
0xEC	DIVW3	Divide word, 3-address
0xEE	DIVH3	Divide halfword, 3-address
0xEF	DIVB3	Divide byte, 3-address
0xF0	ORW3	OR word, 3-address
0xF2	ORH3	OR halfword, 3-address
0xF3	ORB3	OR byte, 3-address
0xF4	XORW3	Exclusive OR word, 3-address
0xF6	XORH3	Exclusive OR halfword, 3-address
0xF7	XORB3	Exclusive OR byte, 3-address
0xF8	ANDW3	AND word, 3-address
0xFA	ANDH3	AND halfword, 3-address
0xFB	ANDB3	AND byte, 3-address
0xFC	SUBW3	Subtract word, 3-address
0xFE	SUBH3	Subtract halfword, 3-address
0xFF	SUBB3	Subtract byte, 3-address

Instruction Set Summary by Functional Group

Tables 9 through 15 summarize the instruction set by functional group. The entries in the conditions column refer to the condition flag code assignment cases listed in Table 16. The entries in the byte column refer to the range of bytes that the associated instruction, along with operands, requires.

Instruction Timing

The following characteristics of the *WE* 32100 Microprocessor architecture make exact instruction timing calculations difficult:

- Addressing modes of operands
- On-chip instruction cache
- Instruction pipelining
- Instruction and data alignment
- Data dependencies

Tests were run to isolate the effects of each of the above characteristics. The entries in the cycle count column in Tables 9 through 15 contain the ranges, from practical best to worst case, derived from these tests. (It is recommended that actual benchmarks be run to more accurately measure performance.) The following describe the results in greater detail.

Addressing Modes of Operands. Since the instruction set is orthogonal to the addressing modes of its operands, tests were done on combinations of the five basic addressing mode classes (register, absolute address, register deferred, immediate, and absolute deferred) for each instruction. In all cases, register operands take the least time, while the absolute deferred operands take the most time to access.

On-Chip Instruction Cache. Tests were run with all instruction cache hits (best case) and all instruction misses (worst case). The instruction cache improvements ranged from 20—60% for arithmetic and logical instructions when instruction fetches were eliminated.

Instruction Pipelining. Test instructions were placed between two other instructions that had potential for overlapped execution in the CPU's pipeline (best case). They were also placed between two branch-taken instructions to eliminate pipeline overlap (worst case). These tests showed that pipelining saved between 2 to 6 cycles per instruction.

Instruction and Data Alignment. In tests that took this effect into account, performance increases of an average of 2 to 6 cycles were encountered for optimal alignment. Optimal alignment was obtained by placing as many of the test instruction's opcodes and associated operands as possible on word boundaries. Worst case alignment minimizes alignment of the opcode and operands.

Data Dependencies. This effect was found only in five instructions: MULW2, DIVW2, MOVBLW, STRCPY, and STREND. In the test involving the MULW2 and DIVW2 instructions, timing is optimized if at least one operand is zero or one. For the string instructions, the length of the string has great impact on the instruction execution time. Since string lengths are not limited, tests were run on strings of one byte (best case) and four bytes (worst case) to determine best and worst case timings.

Arithmetic Group

These instructions perform arithmetic operations on data in registers and memory.

Table 9. Arithmetic Group

Instruction	Mnemonic	Opcode	Bytes	Cycles	Conditions*
Add					
Add byte	ADDB2	0x9F	3—11	4—33	
Add halfword	ADDH2	0x9E	3—11	4—33	
Add word	ADDW2	0x9C	3—11	2—31	Case 2
Add byte, 3-address	ADDB3	0xDF	4—16	4—44	
Add halfword, 3-address	ADDH3	0xDE	4—16	4—44	
Add word, 3-address	ADDW3	0xDC	4—16	4—43	

Table 9. Arithmetic Group (Continued)

Instruction	Mnemonic	Opcode	Bytes	Cycles	Conditions*
Subtract					
Subtract byte	SUBB2	0xBF	3—11	4—33	
Subtract halfword	SUBH2	0xBE	3—11	4—33	
Subtract word	SUBW2	0xBC	3—11	2—31	
Subtract byte, 3-address	SUBB3	0xFF	4—16	4—44	
Subtract halfword, 3-address	SUBH3	0xFE	4—16	4—43	
Subtract word, 3-address	SUBW3	0xFC	4—16	4—43	Case 2
Increment					
Increment byte	INCB	0x93	2—6	2—24	
Increment halfword	INCH	0x92	2—6	2—24	
Increment word	INCW	0x90	2—6	1—22	
Decrement					
Decrement byte	DECB	0x97	2—6	2—24	
Decrement halfword	DECH	0x96	2—6	2—24	
Decrement word	DECW	0x94	2—6	1—22	
Multiply					
Multiply byte	MULB2	0xAB	3—11	20—91	
Multiply halfword	MULH2	0xAA	3—11	20—130	Case 3
Multiply word	MULW2	0xA8	3—11	18—210	
Multiply byte, 3-address	MULB3	0xEB	4—16	22—204	
Multiply halfword, 3-address	MULH3	0xEA	4—16	22—200	Case 4
Multiply word, 3-address	MULW3	0xE8	4—16	20—205	
Divide					
Divide byte	DIVB2	0xAF	3—11	21—154	
Divide halfword	DIVH2	0xAE	3—11	21—194	Case 3
Divide word	DIVW2	0xAC	3—11	19—275	
Divide byte, 3-address	DIVB3	0xEF	4—16	23—270	
Divide halfword, 3-address	DIVH3	0xEE	4—16	23—263	Case 4
Divide word, 3-address	DIVW3	0xEC	4—16	21—275	
Modulo					
Modulo byte	MODB2	0xA7	3—11	21—154	
Modulo halfword	MODH2	0xA6	3—11	21—194	Case 3
Modulo word	MODW2	0xA4	3—11	19—275	
Modulo byte, 3-address	MODB3	0xE7	4—16	23—270	
Modulo halfword, 3-address	MODH3	0xE6	4—16	23—263	Case 4
Modulo word, 3-address	MODW3	0xE4	4—16	21—275	
Arithmetic Shift					
Arithmetic left shift word	ALSW3	0xC0	4—16	5—43	Case 5
Arithmetic right shift byte	ARSB3	0xC7	4—16	5—44	
Arithmetic right shift halfword	ARSH3	0xC6	4—16	5—44	Case 3
Arithmetic right shift word	ARSW3	0xC4	4—16	5—43	

* Refer to Table 16 for condition flag code assignments.

Data Transfer Group

These instructions transfer data to and from registers and memory.

Table 10. Data Transfer Group

Instruction	Mnemonic	Opcode	Bytes	Cycles	Conditions*
Move					
Move byte	MOVB	0x87	3—11	2—31	
Move halfword	MOVH	0x86	3—11	2—31	
Move word	MOVW	0x84	3—11	1—27	
Move address (word)	MOVAW	0x04	3—11	2—22	Case 1
Move complemented byte	MCOMB	0x8B	3—11	2—31	
Move complemented halfword	MCOMH	0x8A	3—11	2—31	
Move complemented word	MCOMW	0x88	3—11	1—27	
Move negated byte	MNEGB	0x8F	3—11	2—31	
Move negated halfword	MNEGH	0x8E	3—11	2—31	Case 2
Move negated word	MNEGW	0x8C	3—11	1—27	
Move version number	MVERNO	0x3009	2	—	Unchanged
Swap (Interlocked)					
Swap byte interlocked	SWAPBI	0x1F	2—6	22—33	
Swap halfword interlocked	SWAPHI	0x1E	2—6	22—33	Case 1
Swap word interlocked	SWAPWI	0x1C	2—6	18—28	
Block Operations					
Move block of words	MOVBLW	0x3019	2	—	Unchanged
Field Operations					
Extract field byte	EXTFB	0xCF	5—21	7—55	
Extract field halfword	EXTFH	0xCE	5—21	7—55	
Extract field word	EXTFW	0xCC	5—21	4—54	
Insert field byte	INSFB	0xCB	5—21	18—72	
Insert field halfword	INSFH	0xCA	5—21	18—72	Case 1
Insert field word	INSFW	0xC8	5—21	14—71	
String Operations					
String copy	STRCPY	0x3035	2	83—182**	Unchanged
String end	STREND	0x301F	2	54—120**	

* Refer to Table 16 for condition flag code assignments.
** Dependent on string length. See Instruction Timing.

Logical Group

These instructions perform logical operations on data in registers and memory.

Table 11. Logical Group

Instruction	Mnemonic	Opcode	Bytes	Cycles	Conditions*
AND					
AND byte	ANDB2	0xBB	3—11	4—33	
AND halfword	ANDH2	0xBA	3—11	4—33	
AND word	ANDW2	0xB8	3—11	2—31	
AND byte, 3-address	ANDB3	0xFB	4—16	4—44	
AND halfword, 3-address	ANDH3	0xFA	4—16	4—44	
AND word, 3-address	ANDW3	0xF8	4—16	4—43	
Exclusive OR (XOR)					
Exclusive OR byte	XORB2	0xB7	3—11	4—33	
Exclusive OR halfword	XORH2	0xB6	3—11	4—33	Case 1
Exclusive OR word	XORW2	0xB4	3—11	2—31	
Exclusive OR byte 3-address	XORB3	0xF7	4—16	4—44	
Exclusive OR halfword 3-address	XORH3	0xF6	4—16	4—44	
Exclusive OR word, 3-address	XORW3	0xF4	4—16	4—43	
OR					
OR byte	ORB2	0xB3	3—11	4—33	
OR halfword	ORH2	0xB2	3—11	4—33	
OR word	ORW2	0xB0	3—11	2—31	
OR byte, 3-address	ORB3	0xF3	4—16	4—44	
OR halfword, 3-address	ORH2	0xF2	4—16	4—44	
OR word, 3-address	ORW3	0xF0	4—16	4—43	
Compare or Test					
Compare byte	CMPB	0x3F	3—11	4—33	
Compare halfword	CMPH	0x3E	3—11	4—33	Case 11
Compare word	CMPW	0x3C	3—11	2—31	
Test byte	TSTB	0x2B	2—6	2—24	
Test halfword	TSTH	0x2A	2—6	2—24	Case 6
Test word	TSTW	0x28	2—6	1—18	
Bit test byte	BITB	0x3B	3—11	4—31	
Bit test halfword	BITH	0x3A	3—11	4—31	Case 1
Bit test word	BITW	0x38	3—11	2—30	
Clear					
Clear byte	CLRB	0x83	2—6	2—21	
Clear halfword	CLRH	0x82	2—6	2—21	Case 2
Clear word	CLRW	0x80	2—6	1—19	
Rotate or Logical Shift					
Rotate word	ROTW	0xD8	4—16	5—43	
Logical left shift byte	LLSB3	0xD3	4—16	5—44	
Logical left shift halfword	LLSH3	0xD2	4—16	5—44	Case 1
Logical left shift word	LLSW3	0xD0	4—16	5—43	
Logical right shift word	LRSW3	0xD4	4—16	5—43	

* Refer to Table 16 for condition flag code assignments.

Program Control Group

These instructions alter normal sequential program flow. Unconditional transfers (branch, jump, or return) change the contents of the PC (r15) to the value specified. Conditional transfers first examine the status of the microprocessor flags to determine if the specified transfer should be executed. Subroutine and procedure transfer instructions save or restore registers so control can transfer to the subroutine or procedure and then return to the original program sequence.

Table 12. Program Control Group

Instruction	Mnemonic	Opcode	Bytes	Cycles	Conditions*
Unconditional Transfer					
Branch with byte displacement	BRB	0x7B	2	5—16	
Branch with halfword displacement	BRH	0x7A	3	5—14	
Jump	JMP	0x24	2—6	7—17	
Conditional Transfers					
Branch on carry clear byte	BCCB	0x53**	2	†	
Branch on carry clear halfword	BCCH	0x52**	3	††	
Branch on carry set byte	BCSB	0x5B**	2	†	
Branch on carry set halfword	BCSH	0x5A**	3	††	
Branch on overflow clear, byte displacement	BVCB	0x63	2	†	
Branch on overflow clear, halfword displacement	BVCH	0x62	3	††	
Branch on overflow set, byte displacement	BVSB	0x6B	2	†	Unchanged
Branch on overflow set, halfword displacement	BVSH	0x6A	3	††	
Branch on equal byte (duplicate)	BEB	0x6F	2	†	
Branch on equal byte	BEB	0x7F	2	†	
Branch on equal halfword (duplicate)	BEH	0x6E	3	††	
Branch on equal halfword	BEH	0x7E	3	††	
Branch on not equal byte (duplicate)	BNEB	0x67	2	†	
Branch on not equal byte	BNEB	0x77	2	†	
Branch on not equal halfword (duplicate)	BNEH	0x66	3	††	
Branch on not equal halfword	BNEH	0x76	3	††	
Branch on less than byte (signed)	BLB	0x4B	2	†	
Branch on less than halfword (signed)	BLH	0x4A	3	††	

* Refer to Table 16 for condition flag code assignments.
** Opcode matches another instruction and the operation performed is identical.
† 5—10 cycles during a branch not taken; 7—14 cycles during a branch taken.
†† 5—10 cycles during a branch not taken; 7—12 cycles during a branch taken.

Table 12. Program Control Group (Continued)

Instruction	Mnemonic	Opcode	Bytes	Cycles	Conditions[1]
Branch on less than byte (unsigned)	BLUB	0x5B[2]	2	Note 3	
Branch on less than halfword (unsigned)	BLUH	0x5A[2]	3	Note 4	
Branch on less than or equal byte (signed)	BLEB	0x4F	2	Note 3	
Branch on less than or equal halfword (signed)	BLEH	0x4E	3	Note 4	
Branch on less than or equal byte (unsigned)	BLEUB	0x5F	2	Note 3	
Branch on less than or equal halfword (unsigned)	BLEUH	0x5E	3	Note 4	
Branch on greater than byte (signed)	BGB	0x47	2	Note 3	
Branch on greater than halfword (signed)	BGH	0x46	3	Note 4	
Branch on greater than byte (unsigned)	BGUB	0x57	2	Note 3	Unchanged
Branch on greater than halfword (unsigned)	BGUH	0x56	3	Note 4	
Branch on greater than or equal byte (signed)	BGEB	0x43	2	Note 3	
Branch on greater than or equal halfword (signed)	BGEH	0x42	3	Note 4	
Branch on greater than or equal byte (unsigned)	BGEUB	0x53[2]	2	Note 3	
Branch on greater than or equal halfword (unsigned)	BGEUH	0x52[2]	3	Note 4	
Return on carry clear	RCC	0x50[2]	1	Note 5	
Return on carry set (signed)	RCS	0x58[2]	1	Note 5	
Return on overflow clear	RVC	0x60	1	Note 5	
Return on overflow set (signed)	RVS	0x68	1	Note 5	
Return on equal (unsigned)	REQLU	0x6C[2]	1	Note 5	
Return on equal (signed)	REQL	0x7C[2]	1	Note 5	
Return on not equal (unsigned)	RNEQU	0x64[2]	1	Note 5	
Return on not equal (signed)	RNEQ	0x74[2]	1	Note 5	
Return on less than (signed)	RLSS	0x48	1	Note 5	
Return on less than (unsigned)	RLSSU	0x58[2]	1	Note 5	
Return on less than or equal (signed)	RLEQ	0x4C	1	Note 5	

[1]Refer to Table 16 for condition flag code assignments.
[2]Opcode matches another instruction and the operation performed is identical.
[3]5—10 cycles during a branch not taken; 7—14 cycles during a branch taken.
[4]5—10 cycles during a branch not taken; 7—12 cycles during a branch taken.
[5]4—5 cycles during a return not taken; 13—14 cycles during a branch taken.

Table 12. Program Control Group (Continued)

Instruction	Mnemonic	Opcode	Bytes	Cycles	Conditions[1]
Return on less than or equal (unsigned)	RLEQU	0x5C	1	Note 2	
Return on greater than (signed)	RGTR	0x44	1	Note 2	
Return on greater than equal (unsigned)	RGTRU	0x54	1	Note 2	
Return on greater than or equal (signed)	RGEQ	0x40	1	Note 2	
Return on greater than or equal (unsigned)	RGEQU	0x50[3]	1	Note 2	
Subroutine Transfer					
Branch to subroutine, byte displacement	BSBB	0x37	2	Note 4	Unchanged
Branch to subroutine, halfword displacement	BSBH	0x36	3	Note 5	
Jump to subroutine	JSB	0x34	2—6	7—17	
Return from subroutine	RSB	0x78	1	13—14	
Procedure Transfer					
Save registers	SAVE	0x10	2	11—36[6]	
Restore registers	RESTORE	0x18	2	12—38[6]	
Call procedure	CALL	0x2C	7	25—36	
Return from procedure	RET	0x08	1	21—23	

[1] Refer to Table 16 for condition flag code assignments.
[2] 4—5 cycles during a return not taken; 13—14 cycles during a branch taken.
[3] Opcode matches another instruction and the operation performed is identical.
[4] 5—10 cycles during a branch not taken; 7—14 cycles during a branch taken.
[5] 5—10 cycles during a branch not taken; 7—12 cycles during a branch taken.
[6] Dependent on number of registers saved/restored.

Stack and Miscellaneous Groups

These instructions manipulate the stack and alter the machine state.

Table 13. Stack Miscellaneous Groups

Instruction	Mnemonic	Opcode	Bytes	Cycles	Conditions*
Stack Operations					
Push address word	PUSHAW	0xE0	2—6	9—20	
Push word	PUSHW	0xA0	2—6	8—23	Case 1
Pop word	POPW	0x20	2—6	9—23	
Miscellaneous					
No operation, 1 byte	NOP	0x70	1	4—11	
No operation, 2 byte	NOP2	0x73	2	4—10	
No operation, 3 byte	NOP3	0x72	3	4—10	
Breakpoint trap	BPT	0x2E	1	—	Unchanged
Cache flush	CFLUSH	0x27	1	—	
Extended opcode	EXTOP	0x14	1, 2	—	

* Refer to Table 16 for condition flag code assignments.

Coprocessor Group Instruction

These instructions implement the interface with coprocessors.

Table 14. Coprocessor Group

Instruction	Mnemonic	Opcode	Bytes	Cycles	Conditions*
Coprocessor operation	SPOP	0x32	5	—	Case 10
Coprocessor operation read single	SPOPRS	0x22	6—10	—	
Coprocessor operation read double	SPOPRD	0x02	6—10	—	
Coprocessor operation read triple	SPOPRT	0x06	6—10	—	
Coprocessor operation single 2-address	SPOPS2	0x23	7—15	—	
Coprocessor operation double 2-address	SPOPD2	0x03	7—15	—	
Coprocessor operation triple 2-address	SPOPT2	0x07	7—15	—	
Coprocessor operation write single	SPOPWS	0x33	6—10	—	
Coprocessor operation write double	SPOPWD	0x13	6—10	—	
Coprocessor operation write triple	SPOPWT	0x17	6—10	—	

* Refer to Table 16 for condition flag code assignments.

Operating System Instructions and Microsequences

These instructions manipulate, via a sequence of actions, the context (internal registers and pointers) of the microprocessor to permit changing the process that is currently in control. The instruction or microsequence is classified as privileged or nonprivileged. Privileged instructions are executed only if the kernel-execution level is in effect. Nonprivileged instructions do not depend upon the execution privilege level.

Table 15. Operating System Group

Instruction	Mnemonic	Opcode	Bytes	Cycles	Conditions*
Nonprivileged Instructions					
Move translated word	MOVTRW	0x0C	9	—	Case 7
Gate	GATE	0x3061	2	134—136	Case 8
Return from gate	RETG	0x3045	2	69—71	Case 9
Privileged Instructions					
Enable virtual pin and jump	ENBVJMP	0x300D	2	—	Unchanged
Disable virtual pin and jump	DISVJMP	0x3013	2	—	
Call process	CALLPS	0x30AC	2	**	Case 8
Return to process	RETPS	0x30C8	2	†	Case 9
Wait for interrupt	WAIT	0x2F	1	—	Unchanged
Interrupt acknowledge	INTACK	0x302F	5—6	—	

* Refer to Table 16 for condition flag code assignments.
** 224—226 cycles with R-bit and no block moves; 302— 304 cycles with R-bit set and no block moves.
† 189—191 cycles with R-bit and no block moves; 277—279 cycles with R-bit set and no block moves.

Other Microsequences

The microprocessor automatically executes a sequence of actions in response to interrupts, exceptions (normal, stack, process, or reset), or context switches. The following are the corresponding microsequences performed for each of these conditions:

On-Interrupt. Used on an interrupt to perform an implicit process switch from the interrupted process to the interrupt handler or to perform the quick-interrupt action.

On-Normal Exception. Performs a function similar to that of a GATE to transfer control to the appropriate exception-handler function (determined from the internal state code (ISC) field in the PSW).

On-Stack Exception. Performed when a stack bounds violation is detected during a GATE instruction or a normal exception.

On-Process Exception. Performed when the microprocessor receives a memory fault signal on a PCB access.

On-Reset Exception. Used to restart the system when an error condition in accessing critical system data is detected or when an external reset request occurs.

Context Switches. Three context switch microsequences are available: XSWITCH_ONE, XSWITCH_TWO, and XSWITCH_THREE. Various combinations of the three are used to save the context (control registers) of the current process, load the context of the new process, and execute block moves for the new process.

Table 16. Condition Flag Code Assignments

Case	Condition Flags* N (Negative)	Z (Zero)	C (Carry)	V (Overflow)	Special Conditions**
1	MSB of dst	1 if dst = 0	0	0	V flag is set when expanded-operand mode is used, and the result is truncated when represented in dst.
2	1 if result < 0	1 if result = 0	1 on carry or borrow	1 on integer overflow	—
3	1 if dst < 0	1 if dst = 0	0	1 on integer overflow	—
4	1 if dst < 0	1 if dst = 0	0	1 on integer overflow	V flag may not be set when dst is a signed word and bit 31 of absolute value of the result is 1 and bits 32—63 are 0s.
5	1 if dst < 0	1 if dst =0	0	0	V flag is set if expanded-operand mode changes the type of dst and integer overflow occurs.
6	1 if src < 0	1 if src = 0	0	0	N flag is affected if src is signed integer.
7	MSB of word returned	1 if word returned = 0	0	0	—
8	—	—	—	—	All flags determined by new PSW.
9	—	—	—	—	All flags determined by restored PSW.
10	—	—	—	—	When coprocessor status word is accepted, bits 18—21 of the word read are put into bits 18—21 of the PSW, respectively.
11	1 if result < 0	1 if result = 0	1 on carry or borrow	0	—

* MSB is most significant bit, dst is destination, and src is source.
** Cases 1 through 6: when PSW is the source, condition flags are unaffected; when PSW is destination, condition flags assume the value of bits 18—21 of the operation performed.

Addressing Modes

The addressing mode of an operand in an instruction refers to the way in which the location of the operand is determined. A memory-based operand requires an effective address, determined in reference to the operand descriptor. The operand descriptor for register operands specifies the register. Unless specified by the instruction, all operands are addressed by a descriptor. An operand descriptor may be one or more bytes. The format of the first byte of a descriptor is

mmmmrrrr

where rrrr (bits 0—3) represent a general-purpose register (r0—r15) and mmmm (bits 4—7) represent the addressing mode. The addressing modes are summarized in Table 17 and are interpreted as:

Mode Field **Description**

0—3 **Positive Literal Mode.** The four bits rrrr (r0—r15) concatenated with the low-order two bits of mmmm constitute a positive 6-bit literal. If a literal is specified as a destination or its address is requested by the opcode, an illegal operand exception is generated.

4 **Register Mode.** Bits rrrr specify one of 15 registers (r0—r14). An illegal operand exception is generated if the instruction attempts to take this operand's address.

If rrrr specifies PC, the mode becomes "word immediate." The four bytes following the mmmmrrrr byte provide a 32-bit immediate value, with first byte following the operand descriptor byte containing the low-order bits (0—7).

5 **Register Deferred Mode.** The register specified (r0—r10 and r12—r14) contains the operand address; r11 cannot be specified.

If rrrr specifies PC, the mode becomes "halfword immediate." The two bytes following the mmmmrrrr byte provide a 16-bit immediate value (range –32768 to 32767). The first byte contains the low-order bits of the immediate value; the second contains the high-order bits.

6 **FP Short Offset Mode.** This mode implicitly refers to the FP (r9). Bits rrrr are used as a literal (range 0 to 14) and added to the FP to produce the operand address. This mode, an optimization of the "byte displacement" mode for short FP offsets, is useful for referring to local variables of high-level language functions.

If rrrr specifies PC, the mode becomes byte immediate. The byte following the mmmmrrrr byte provides 8-bit immediate value (range –128 to 127).

7 **AP Short Offset Mode.** This mode implicitly refers to the AP (r10). Bits rrrr are used as a literal (range 0 to 14) and added to the AP to produce the operand address. This mode, an optimization of the "byte displacement" mode for short AP offsets, is useful for referring to high-level language function arguments.

If rrrr specifies PC, the mode becomes "absolute." The four bytes following the mmmmrrrr byte provide the operand address for instruction. The first byte contains bits 0—7 (low-order bits); the second, bits 8—15; the third, bits 16—23; and the fourth, bits 24—31.

8 **Word Displacement Mode.** The four bytes following the operand descriptor byte are fetched from the instruction stream and added to the register specified by rrrr. Bits rrrr can range from 0 to 10 and 12 to 15; r11 cannot be specified. The sum is the address of the instruction operand. The first byte contains the low-order bits of the displacement value.

9 **Word Displacement Deferred Mode.** The four bytes following the operand descriptor byte are fetched from the instruction stream and added to the register specified by rrrr. Bits rrrr can range from 0 to 10 and 12 to 15; r11 cannot be specified. The first byte contains the low-order bits of the displacement value. The sum is the address of a 32-bit pointer, which is fetched and used as the instruction's operand address.

10 **Halfword Displacement Mode.** The two bytes following the operand descriptor byte are fetched from the instruction stream, sign-extended to 32 bits, and added to the rrrr register. Bits rrrr can range from 0 to 10 and 12 to 15; r11 cannot be specified. The first byte contains the low-order bits of the displacement value. The sum is the instruction's operand address.

11 **Halfword Displacement Deferred Mode.** The two bytes following the operand descriptor byte are fetched from the instruction stream, sign-extended to 32 bits, and added to the rrrr register. Bits rrrr can range from 0 to 10 and 12 to 15; r11 cannot be specified. The first byte contains the low-order bits of the displacement value. The sum is the address of a 32-bit pointer, which is fetched and used as the instruction's operand address.

12 **Byte Displacement Mode.** The byte following the operand descriptor byte is fetched from the instruction stream, sign-extended to 32 bits, and added to the register specified by rrrr. Bits rrrr can range from 0 to 10 and 12 to 15; r11 cannot be specified. The sum is the instruction's operand address.

13 **Byte Displacement Deferred Mode.** The byte following the operand descriptor byte is fetched from the instruction stream, sign-extended to 32 bits, and added to the register specified by rrrr. Bits rrrr can range from 0 to 10 and 12 to 15; r11 cannot be specified. The sum is the address of a 32-bit pointer, which is fetched and used as the instruction's operand address.

14 **Expanded Operand Type Mode.** Operand type and its address mode are determined by a combination of the mmmmrrrr byte (r0—r14) and the next byte in the instruction stream. This next byte is formatted as

nnnngggg

where nnnn (bits 4—7) defines the actual address mode and gggg (bits 0—3) defines the actual register of the operand. A value of 14 for nnnn generates an invalid descriptor exception, i.e., only one level of expansion is possible. The mode determination register is provided by the gggg field of the second byte. Operand type is determined by the rrrr of the first byte. Field type overrides instruction opcode type and may be overridden only by another expanded operand type descriptor. It does not override the width description of immediate operands. Byte immediates will still be 8 bits, halfword 16, and word 32. Valid types for rrrr are:

7 Signed byte
3 Unsigned byte
6 Signed halfword
2 Unsigned halfword
4 Signed word
0 Unsigned word

Types 1, 5, and 8—14 are not defined on the *WE* 32100 Microprocessor and generate a reserved data-type exception. If r15 (PC) is specified by rrrr, the mode becomes "absolute deferred" instead of expanded operand. The four bytes following the mmmmrrrr byte provide a 32-bit absolute address of the instruction operand. The first byte contains bits 0—7; the second, bits 8—15; the third, bits 16—23; and the fourth, bits 24—31.

15 **Negative Literal Mode.** Bits rrrr **(r0—r15).** Concatenated with mmmm constitute a negative 8-bit literal (range −1 to −16). If a literal is specified as a destination, or if the opcode requests the literal operand's address, an illegal operand exception is generated.

Table 17. Addressing Modes

Mode	Syntax	Mode Field	Register Field	Total Bytes	Notes
Absolute					
Absolute	*$expr*	7	15	5	—
Absolute deferred	**$expr*	14	15	5	—
Displacement (From a Register)					
Byte displacement	*expr*(%r*n*)	12	0—10, 12—15	2	1
Byte displacement deferred	**expr*(%r*n*)	13	0—10, 12—15	2	1
Halfword displacement	*expr*(%r*n*)	10	0—10, 12—15	3	1
Halfword displacement deferred	**expr*(%r*n*)	11	0—10, 12—15	3	1
Word displacement	*expr*(%r*n*)	8	0—10, 12—15	5	1
Word displacement deferred	**expr*(%r*n*)	9	0—10, 12—15	5	1
AP short offset	*so*(%ap)	7	0—14	1	2
FP short offset	*so*(%fp)	6	0—14	1	2
Immediate					
Byte immediate	*&imm8*	6	15	2	
Halfword immediate	*&imm16*	5	15	3	
Word immediate	*&imm32*	4	15	5	3,4
Positive literal	*&lit*	0—3	0—15	1	
Negative literal	*&lit*	15	0—15	1	
Register					
Register	%r*n*	4	0—14	1	2,4
Register deferred	(%r*n*)	5	0—10, 12—14	1	1,2
Special Mode					
Expanded-operand type	*{type}opnd*	14	0—14	2—6	5

[1] r11 cannot be specified with this mode.
[2] Mode field has special meaning if register field is 15; see absolute or immediate modes in this table.
[3] Mode may not be used for a destination operand.
[4] Mode may not be used if the instruction takes effective address of the operand.
[5] *type* overrides instruction type; *opnd* determines actual address mode. For total bytes, add 1 to byte count for address mode determined by *opnd*.

Exceptions

Bus Exceptions

Bus exceptions cause the termination of the current memory access and result when retry is required or when an exception occurs during an access. The bus exceptions are of three types: relinquish and retry, retry, and exception. A relinquish and retry causes the microprocessor to give up its bus and to subsequently retry the preempted access once the bus has been returned to its control. An external device may request a relinquish and retry by asserting the relinquish and retry request ($\overline{\text{RRREQ}}$) input, driving the input to its active state. A retry causes the microprocessor to retry the access. An external device may request a retry by asserting the $\overline{\text{RETRY}}$ input. An exception is the result of an error condition during a bus cycle, which an external device can report to the microprocessor by asserting the $\overline{\text{FAULT}}$ input. This causes the microprocessor to terminate the access and, perhaps, to execute an exception handling routine, depending on the type of access.

Table 18 describes how the microprocessor handles the simultaneous assertion of two or more bus exceptions.

Exceptional Conditions

There are four groups of exceptional conditions: normal, stack, process, and reset. Each group consists of several exceptional conditions. The exception type (ET) field in the processor status word (PSW) indicates which of the four groups caused an exception, and the internal state code (ISC) field distinguishes between exceptional conditions in each group.

Normal Exceptions. Normal exceptions consist of three types of events generated by the microprocessor: traps, internal faults, and external faults. When a trap is generated, the instruction that caused the trap is completed, and the PC points to the next executable instruction. Note that integer overflows may or may not behave in this way due to pipelining. When an internal or external fault is generated, the PC points to the opcode of the instruction that caused the exception. This instruction may have been executed partially or not at all. Each type of trap or exception uses a different trap vector to branch to the corresponding exception handling software.

Table 18. Simultaneously Asserted Exception Conditions

Simultaneously Asserted Signals	Behavior
Condition 1: $\overline{\text{RRREQ}}$, $\overline{\text{RETRY}}$, $\overline{\text{FAULT}}$	In this instance, the relinquish and retry request ($\overline{\text{RRREQ}}$) is honored first. The microprocessor acknowledges this request by relinquishing the bus and then asserting the relinquish and retry request acknowledge ($\overline{\text{RRRACK}}$) output. The access is retried once $\overline{\text{RRREQ}}$ and $\overline{\text{RETRY}}$ are negated by the requesting devices. If the fault ($\overline{\text{FAULT}}$) input is still asserted during the retried access, the exception is honored (recognized). The fault input will be recognized only during the retried access.
Condition 2: $\overline{\text{RRREQ}}$, $\overline{\text{RETRY}}$	In this instance, the relinquish and retry request($\overline{\text{RRREQ}}$) is honored first. The microprocessor 3-states the appropriate signals and then asserts the relinquish and retry acknowledge ($\overline{\text{RRRACK}}$) output. The access is retried once $\overline{\text{RRREQ}}$ and $\overline{\text{RETRY}}$ are negated.
Condition 3: $\overline{\text{RRREQ}}$, $\overline{\text{FAULT}}$	Same as condition 1.
Condition 4: $\overline{\text{RETRY}}$, $\overline{\text{FAULT}}$	In this instance, the $\overline{\text{RETRY}}$ request is honored first. The fault is recognized on the retried access if it is still asserted.

Note: This table applies only when the microprocessor is the bus master ($\overline{\text{BARB}}$ asserted).

There are three kinds of traps:

- **Breakpoint Trap.** This trap is invoked whenever the breakpoint trap (BPT) instruction is executed.
- **Integer Overflow.** This trap is enabled when the overflow enable (OE) bit in the PSW is set. Overflow trapping has the following behavior:
  - Whenever an overflow trap occurs, the OE bit is cleared before the PSW is saved.
  - When an instruction causes an overflow trap to occur, the next instruction may or may not be executed before the microsequence is entered. Therefore, the saved PC may point to the instruction following the trapped instruction, or to the next instruction after that one. Even if the instruction following the trapped instruction is executed, flags may or may not be set.
  - If two consecutive instructions cause overflow traps, only one overflow trap will occur.
  - An overflow trap occurs if the OE bit is set and the execution of an instruction causes the V (overflow) bit in the PSW to be set after the instruction is completed. Specifically, this can be caused by the return from gate (RETG), return to process (RETPS) instructions, or an explicit move to the PSW.
  - When an overflow trap occurs, the overflow flag may or may not be set in the PSW that is saved.
- **Trace Trap.** Trace trapping is enabled when the trace enable (TE) bit in the PSW is set. This causes a trace trap to occur after each instruction is executed, except for the RETPS, CALLPS, and RETG instructions.

The ten types of internal and external faults generated by the microprocessor are:

- **External Memory.** This exception occurs when alignment requirements are violated, if an external device asserts the $\overline{\text{FAULT}}$ input on an access, if an exception occurs during a coprocessor status fetch, or if no coprocessor responds to a coprocessor broadcast. The following list describes the alignment exception behavior:
  - No alignment exception ever occurs on a byte access.
  - No alignment exception ever occurs on an instruction fetch access.
  - An alignment exception occurs if the access is a data access of word length and address bit 1 (ADDR01) and/or address bit 0 (ADDR00) is 1.
  - An alignment exception occurs if the access is a data access of halfword length and address bit 0 (ADDR00) is 1.
- **Gate Vector.** This exception is caused by a memory fault when reading gate tables during a GATE instruction.
- **Illegal Level Change.** This exception is caused by an attempt to increase the current execution privilege on a return from gate (RETG) instruction.
- **Illegal Opcode.** The opcode is not defined for the microprocessor.
- **Integer Zero-Divide.** This exception is caused by an attempt to divide by zero and is always enabled. This exception resets the PSW flags.
- **Invalid Descriptor.** Address mode generated cannot be used in the way specified below. This exception resets the PSW flags and may result from the following causes:
  - Literal or immediate used as destination
  - Effective address requested of literal or immediate
  - Effective address requested of a register.
- **Privileged Opcode.** The opcode is defined for kernel-execution level only. An attempt to execute it in another execution level causes this exception.
- **Privileged Register.** An attempt to write the three privileged registers in an execution level other than kernel causes this exception. This exception resets the PSW flags.

- **Reserved Data Type.** The operand type described by the expanded operand descriptor type is not implemented in the microprocessor. This exception resets the PSW flags.
- **Reserved Opcode.** The opcode is not implemented on the microprocessor, but is reserved for future use.

Stack Exceptions. This exception may occur during a process switch or a gate sequence. The three types of stack exceptions are:

- **Interrupt ID Fetch.** A memory exception during the interrupt acknowledge.
- **Stack Bound.** A stack pointer (SP) value outside the upper or lower stack bound on a system call.
- **Stack.** A memory exception when storing the PC or PSW on the execution stack during a system call.

Process Exceptions. These exceptions may occur during a process switch. The three types of process exceptions are:

- **Gate Process Control Block.** A memory exception when accessing the process control block (PCB) for a stack bounds check during a GATE.
- **New Process Control Block.** A memory exception when accessing the PCB for the new process during a process switch.
- **Old Process Control Block.** A memory exception when accessing the PCB for the exiting process on a process switch.

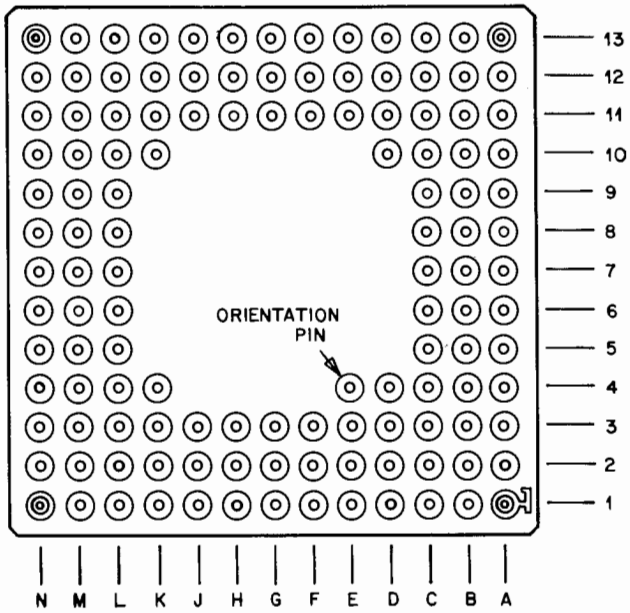
Reset Exceptions. These exceptions are triggered by an error condition in accessing critical system data and require restarting the system. Reset exceptions may occur during reset or during process and normal exceptions. The six types of reset exceptions are:

- **External Reset.** Normal response to an external (system) reset signal.
- **Gate Vector.** A memory exception when accessing a gate table while processing a normal exception.
- **Interrupt Stack.** A memory exception when accessing the interrupt stack.
- **New Process Control Block.** A memory exception when accessing the PCB of an exception-handler process.
- **Old Process Control Block.** A memory exception when accessing the PCB of a process exception-handler.
- **System Data.** A memory exception when accessing the process control block pointers for stack, process, or reset exceptions.

Pin Descriptions

The *WE* 32100 Microprocessor is available in a 125-pin square, ceramic PGA package. Figure 5 and Tables 19—27 describe the pin assignments. The term *asserted* in the pin function descriptions means that the signal is driven to its active state either by the microprocessor (outputs) or an external device (inputs). The term *negated* means that the signal is driven to its inactive state. A bar over a signal name (e.g., $\overline{AS}$) indicates that the signal is active low, logic 0. The 0 bit is the least significant bit for signals that occupy two or more pins (e.g., SAS0). In the type column, I indicates an input, O an output, and I/O a bidirectional signal.

Table 19 lists signals in the numerical order for the 125-pin square PGA package; Tables 20—27 describe the signals by functional group.



Bottom View

Figure 5. WE® 32100 Microprocessor 125-Pin Square, Ceramic PGA Package

Numerical Order

Table 19. Pin Descriptions – 125-Pin Square PGA Package

Pin	Name	Type	Description
A1	DATA23	I/O	Microprocessor data 23
A2	DATA24	I/O	Microprocessor data 24
A3	DATA26	I/O	Microprocessor data 26
A4	ADDR28	O	Microprocessor address 28
A5	IQSO	O	Instruction queue status 0
A6	ADDR30	O	Microprocessor address 30
A7	XMD1	O	Execution mode 1
A8	IQS1	O	Instruction queue status 1
A9	SAS3	O	Access status code 3
A10	XDM0	O	Execution mode 0
A11	SAS2	O	Access status code 2
A12	DSIZE1	O	Data size 1
A13	SAS1	O	Access status code 1

Table 19. Pin Descriptions – 125-Pin Square PGA Package (Continued)

Pin	Name	Type	Description
B1	DATA22	I/O	Microprocessor data 22
B2	DATA28	I/O	Microprocessor data 28
B3	DATA29	I/O	Microprocessor data 29
B4	DATA31	I/O	Microprocessor data 31
B5	ADDR29	O	Microprocessor address 29
B6	ADDR31	O	Microprocessor address 31
B7	$\overline{\text{SOI}}$	O	Start of instruction
B8	$\overline{\text{BRACK}}$	I/O	Bus request acknowledge
B9	$\overline{\text{BUSRQ}}$	I/O	Bus request
B10	SAS0	O	Access status code 0
B11	R/$\overline{\text{W}}$	O	Read/write
B12	DSIZE0	O	Data size 0
B13	$\overline{\text{RRREQ}}$	I	Relinquish and retry request
C1	ADDR19	O	Microprocessor address 19
C2	ADDR26	O	Microprocessor address 26
C3	DATA25	I/O	Microprocessor data 25
C4	ADDR27	O	Microprocessor address 27
C5	DATA30	I/O	Microprocessor data 30
C6	+5V	—	Power
C7	GND	—	Ground
C8	+5V	—	Power
C9	GND	—	Ground
C10	$\overline{\text{DONE}}$	I	Coprocessor done
C11	$\overline{\text{HIGHZ}}$	I	High impedance
C12	$\overline{\text{RRRACK}}$	O	Relinquish and retry request acknowledge
C13	$\overline{\text{FAULT}}$	I	Fault
D1	ADDR24	O	Microprocessor address 24
D2	ADDR20	O	Microprocessor address 20
D3	ADDR25	O	Microprocessor address 25
D4	DATA27	I/O	Microprocessor data 27
D10	$\overline{\text{SRDY}}$	I	Synchronous ready
D11	$\overline{\text{RETRY}}$	I	Retry
D12	$\overline{\text{BARB}}$	I	Bus arbiter
D13	$\overline{\text{RESET}}$	O	Reset
E1	ADDR18	O	Microprocessor address 18
E2	ADDR23	O	Microprocessor address 23
E3	+5V	—	Power
E4	—	—	Device socket orientation pin
E11	+5V	—	Power
E12	$\overline{\text{DTACK}}$	I	Data transfer acknowledge
E13	$\overline{\text{BLKFTCH}}$	I	Block (double-word) fetch
F1	ADDR21	O	Microprocessor address 21
F2	ADDR22	O	Microprocessor address 22
F3	GND	—	Ground
F11	CLK23	I	Input clock 23
F12	$\overline{\text{DSHAD}}$	I	Data bus shadow
F13	$\overline{\text{RESETR}}$	I	Reset request

Table 19. Pin Descriptions – 125-Pin Square PGA Package (Continued)

Pin	Name	Type	Description
G1	DATA21	I/O	Microprocessor data 21
G2	ADDR17	O	Microprocessor address 17
G3	DATA20	I/O	Microprocessor data 20
G11	CLK34	I	Input clock 34
G12	$\overline{\text{CYCLEI}}$	O	Cycle initiate
G13	$\overline{\text{STOP}}$	I	Microprocessor stop
H1	ADDR15	O	Microprocessor address 15
H2	DATA18	I/O	Microprocessor data 18
H3	GND	—	Ground
H11	GND	—	Ground
H12	$\overline{\text{AS}}$	O	Address strobe
H13	$\overline{\text{DS}}$	O	Data strobe
J1	DATA19	I/O	Microprocessor data 19
J2	DATA17	I/O	Microprocessor data 17
J3	+5V	—	Power
J11	+5V	—	Power
J12	$\overline{\text{ABORT}}$	O	Access abort
J13	$\overline{\text{DRDY}}$	O	Data ready
K1	ADDR16	O	Microprocessor address 16
K2	ADDR13	O	Microprocessor address 13
K3	DATA15	I/O	Microprocessor data 15
K4	DATA12	I/O	Microprocessor data 12
K10	DATA00	I/O	Microprocessor data 00
K11	GND	—	Ground
K12	—	—	**WARNING:** This pin is for manufacture use only and must be tied high (5 V).
K13	$\overline{\text{NMINT}}$	I	Nonmaskable interrupt
L1	ADDR14	O	Microprocessor address 14
L2	DATA16	I/O	Microprocessor data 16
L3	DATA11	I/O	Microprocessor data 11
L4	ADDR11	O	Microprocessor address 11
L5	GND	—	Ground
L6	ADDR08	O	Microprocessor address 08
L7	+5V	—	Power
L8	GND	—	Ground
L9	+5V	—	Power
L10	ADDR00	O	Microprocessor address 00
L11	$\overline{\text{INTOPT}}$	I	Interrupt option
L12	IPL3	I	Interrupt priority level 3
L13	$\overline{\text{AVEC}}$	I	Autovector
M1	ADDR12	O	Microprocessor address 12
M2	DATA14	I/O	Microprocessor data 14
M3	DATA13	I/O	Microprocessor data 13
M4	DATA08	I/O	Microprocessor data 08
M5	ADDR07	O	Microprocessor address 07
M6	DATA05	I/O	Microprocessor data 05
M7	ADDR03	O	Microprocessor address 03

Table 19. Pin Descriptions – 125-Pin Square PGA Package (Continued)

Pin	Name	Type	Description
M8	ADDR05	O	Microprocessor address 05
M9	DATA03	I/O	Microprocessor data 03
M10	DATA02	I/O	Microprocessor data 02
M11	DATA01	I/O	Microprocessor data 01
M12	IPL2	I	Interrupt priority level 2
M13	$\overline{\text{VAD}}$	O	Virtual address
N1	DATA10	I/O	Microprocessor data 10
N2	DATA09	I/O	Microprocessor data 09
N3	ADDR10	O	Microprocessor address 10
N4	ADDR09	O	Microprocessor address 09
N5	ADDR06	O	Microprocessor address 06
N6	DATA07	I/O	Microprocessor data 07
N7	ADDR02	O	Microprocessor address 02
N8	DATA06	I/O	Microprocessor data 06
N9	ADDR01	O	Microprocessor address 01
N10	ADDR04	O	Microprocessor address 04
N11	DATA04	I/O	Microprocessor data 04
N12	IPL0	I	Interrupt priority level 0
N13	IPL1	I	Interrupt priority level 1

Functional Groups

Table 20. Address and Data

Name	Pin	Type	Description
ADDR00—ADDR31	L10, N9, N7, M7, N10, M8, N5, M5, L6, N4, N3, L4, M1, K2, L1, H1, K1, G2, E1, C1, D2, F1, F2, E2, D1, D3, C2, C4, A4, B5, A6, B6	O	**Address.** These pins are used by the microprocessor to issue 32-bit addresses for off-chip accesses.
DATA00—DATA31	K10, M11, M10, M9, N11, M6, N8, N6, M4, N2, N1, L3, K4, M3, M2, K3, L2, J2, H2, J1, G3, G1, B1, A1, A2, C3, A3, D4, B2, B3, C5, B4	I/O	**Data.** These bidirectional pins are used to convey data to and from the microprocessor.

Table 21. Interface and Control Signals

Name	Pin	Type	Description
$\overline{\text{AS}}$	H12	O	**Address Strobe.** When low (0), this signal indicates the presence of a valid physical address on the address pins. If the address is virtual, the falling edge of $\overline{\text{AS}}$ indicates a valid address and the address pins are 3-stated subsequent to the falling edge of $\overline{\text{AS}}$.
$\overline{\text{CYCLEI}}$	G12	O	**Cycle Initiate.** This signal is asserted at the beginning of a bus transaction and negated two clock cycles later. $\overline{\text{CYCLEI}}$ is asserted in both the read and write halves of an interlocked-read transaction.
$\overline{\text{DONE}}$	C10	I	**Coprocessor Done.** This input is recognized during a coprocessor instruction. It informs the microprocessor that a coprocessor has completed its operation.

Table 21. Interface and Control Signals (Continued)

Name	Pin	Type	Description
$\overline{\text{DRDY}}$	J13	O	**Data Ready.** When asserted, this signal indicates the end of a bus transaction that has had no bus exceptions ($\overline{\text{FAULT}}$, $\overline{\text{RETRY}}$, $\overline{\text{RRREQ}}$ signals).
$\overline{\text{DS}}$	H13	O	**Data Strobe.** During a read operation this signal, when low, indicates that a slave device can place data on the data bus. During a write operation this signal, when low, indicates that the microprocessor has placed valid data on the data bus.
$\overline{\text{DTACK}}$	E12	I	**Data Transfer Acknowledge.** This signal is used to handshake between the microprocessor and a slave device. During a read operation, the microprocessor latches data present on the data bus and terminates the bus transaction one cycle after $\overline{\text{DTACK}}$ is driven low by a slave device. During a write operation, the transaction is terminated one cycle after a slave device drives $\overline{\text{DTACK}}$ low. If $\overline{\text{DTACK}}$ is high, wait states are inserted in the current cycle. $\overline{\text{DTACK}}$ is ignored if the data bus shadow signal ($\overline{\text{DSHAD}}$) is asserted. The $\overline{\text{DTACK}}$ is an asynchronous input and is double-latched to avoid metastability.
$\overline{\text{SRDY}}$	D10	I	**Synchronous Ready.** This signal is a synchronous input that begins the termination of a read or write operation when asserted. It is sampled only once on the leading edge of the fifth clock state during read and write operations. If $\overline{\text{SRDY}}$ is not asserted at this time and $\overline{\text{DTACK}}$ was not asserted during the previous clock state, wait state cycles are inserted until either signal is asserted. $\overline{\text{SRDY}}$ is ignored if the data bus shadow input ($\overline{\text{DSHAD}}$) was previously asserted.

Table 22. Access Status Signals

Name	Pin	Type	Description
$\overline{\text{BLKFTCH}}$	E13	I	**Block (Double-Word) Fetch.** This input indicates to the microprocessor that the memory system can perform a double-word (8-byte) program block fetch. On all instruction fetches, the data size (DSIZE0, DSIZE1) pins show a double-word access. If the memory system can handle a double-word access, it activates this input. Otherwise, the input is left inactive, and the microprocessor fetches a block of instruction by two consecutive reads.
DSIZE0, DSIZE1	B12, A12	O	**Data Size.** This two-bit output is used to indicate whether the microprocessor is transferring byte, halfword, word, or double-word data in the current bus transaction. On all instruction fetches, the DSIZE0 and DSIZE1 pins have the value for double word.

DSIZE1	DSIZE0	Description
0	0	Word transaction
0	1	Double-word transaction
1	0	Halfword transaction
1	1	Byte transaction

Table 22. Access Status Signals (Continued)

Name	Pin	Type	Description
R/$\overline{W}$	B11	O	**Read/Write.** This signal indicates whether the bus transaction is a read or a write. When low (0), the operation is a write; when high (1), the operation is a read. This pin is valid during the time the address strobe ($\overline{AS}$) is active.
SAS0—SAS3	B10, A13, A11, A9	O	**Access Status Codes.** These pins describe the type of bus transaction being executed. SAS3 SAS2 SAS1 SAS0 Description 0 0 0 0 Move translated word 0 0 0 1 Coprocessor data write acknowledge 0 0 1 0 Autovector interrupt 0 0 1 1 Coprocessor data fetch 0 1 0 0 Stop acknowledge 0 1 0 1 Coprocessor broadcast 0 1 1 0 Coprocessor status fetch 0 1 1 1 Read interlocked 1 0 0 0 Address fetch 1 0 0 1 Operand fetch 1 0 1 0 Write 1 0 1 1 Interrupt acknowledge 1 1 0 0 Instruction fetch after PC discontinuity 1 1 0 1 Instruction prefetch 1 1 1 0 Instruction fetch 1 1 1 1 No operation
$\overline{VAD}$	M13	O	**Virtual Address.** When low, this signal indicates that the address is virtual; when high, that the address is physical. $\overline{VAD}$ is a level signal, asserted by execution of the enable virtual pin and jump (ENBVJMP) instruction and negated by execution of the disable virtual pin and jump (DISVJMP) instruction.
XMD0, XMD1	A10, A7	O	**Execution Mode.** These two outputs indicate the present execution mode of the microprocessor: XMD1 XMD0 Description 0 0 Kernel mode 0 1 Executive mode 1 0 Supervisor mode 1 1 User mode If an MMU is present in the system, it may latch and use a spurious execution mode value if XMD0 or XMD1 changes during an access. Since XMD0 and XMD1 reflect the state of the current execution level (CM) bits in the PSW, changes to the CM field via nonmicrosequence instructions must be avoided.

Table 23. Interrupt Signals

Name	Pin	Type	Description
$\overline{\text{AVEC}}$	L13	I	**Autovector.** When this input is asserted with the interrupt priority level input, the microprocessor supplies its own vector. The vector number value is the inverted interrupt option ($\overline{\text{INTOPT}}$) input concatenated with the interrupt priority level value. When autovector is not asserted, the interrupting device supplies the vector (see Interrupts).
$\overline{\text{INTOPT}}$	L11	I	**Interrupt Option.** This asynchronous input is latched, along with the interrupt priority level inputs IPL0—IPL3, inverted, and output on ADDR06 during an interrupt acknowledge transaction.
IPL0—IPL3	N12, N13, M12, L12	I	**Interrupt Priority Level.** These asynchronous inputs indicate the level of the pending interrupt. The code is based on a decreasing priority scheme, with 0000 having the highest priority and 1110 the lowest. Level 1111 indicates that no interrupts are pending. IPL0 is the least significant bit of the interrupt priority level code. To be acknowledged, the requesting level on the pins inverted must be greater than the present interrupt priority level (IPL field) in the PSW. The exception to this is a nonmaskable interrupt, which can interrupt the microprocessor regardless of the present IPL field priority level.
$\overline{\text{NMINT}}$	K13	I	**Nonmaskable Interrupt.** When asserted, this asynchronous input indicates that a nonmaskable interrupt is being requested. The microprocessor acknowledges this interrupt with an autovector interrupt acknowledge cycle (see Interrupts). During this acknowledge cycle, the microprocessor address bus contains all zeros.

Table 24. Arbitration Signals

Name	Pin	Type	Description
$\overline{\text{BARB}}$	D12	I	**Bus Arbiter.** When this input is strapped low, the microprocessor is the arbiter of the bus. As arbiter, the microprocessor need not request the bus to obtain access to the bus. When the pin is strapped high, the microprocessor is not the arbiter and must request the bus to use it. The following outputs are 3-stated when the microprocessor is not the bus arbiter until the CPU does a bus transaction:

$\overline{\text{ABORT}}$	$\overline{\text{DS}}$
ADDR00—ADDR31	DSIZE0, DSIZE1
$\overline{\text{AS}}$	R/$\overline{\text{W}}$
$\overline{\text{CYCLEI}}$	SAS0—SAS3
DATA00—DATA31	$\overline{\text{VAD}}$
$\overline{\text{DRDY}}$	XMD0, XMD1

Table 24. Arbitration Signals (Continued)

Name	Pin	Type	Description
$\overline{\text{BRACK}}$	B8	I/O	**Bus Request Acknowledge.** This signal is an output if the microprocessor is the arbiter of the bus and an input if it is not. As an output, this pin indicates that the bus request ($\overline{\text{BUSRQ}}$) has been recognized and that the microprocessor has 3-stated the bus for the requesting bus master. The bus signals that 3-stated when the $\overline{\text{BRACK}}$ is issued are: $\overline{\text{ABORT}}$ — $\overline{\text{DS}}$ ADDR00—ADDR31 — DSIZE0, DSIZE1 DATA00—DATA31 — R/$\overline{\text{W}}$ $\overline{\text{AS}}$ — SAS0—SAS3 $\overline{\text{CYCLEI}}$ — $\overline{\text{VAD}}$ $\overline{\text{DRDY}}$ — XMD0, XMD1 As an input, this pin indicates that the microprocessor's bus request has been recognized and that the microprocessor may take possession of the bus.
$\overline{\text{BUSRQ}}$	B9	I/O	**Bus Request.** This asynchronous signal is an input if the microprocessor is the arbiter of the bus and an output if it is not. As an input, this signal indicates that an external device is requesting the bus; as an output, the signal indicates that the microprocessor is requesting the bus.

Table 25. Bus Exception Signals

Name	Pin	Type	Description
$\overline{\text{ABORT}}$	J12	O	**Access Abort.** This pin is asserted on an access that is to be ignored by the memory system. This occurs when the microprocessor has an instruction cache hit on program counter discontinuity or an alignment fault.
$\overline{\text{DSHAD}}$	F12	I	**Data Bus Shadow.** The DATA00—DATA31, $\overline{\text{DRDY}}$, DSIZE0, DSIZE1, and R/$\overline{\text{W}}$ pins are 3-stated when this input is asserted. This input is used by the memory management unit (MMU) to remove the microprocessor from the data bus.
$\overline{\text{FAULT}}$	C13	I	**Fault.** This input notifies the microprocessor that a fault condition has occurred. It is an asynchronous input and is double-latched prior to the assertion of $\overline{\text{DTACK}}$, and synchronous after the assertion of $\overline{\text{DTACK}}$ (latched once). The $\overline{\text{FAULT}}$ signal is ignored if $\overline{\text{DSHAD}}$ is asserted.
$\overline{\text{RESET}}$	D13	O	**Reset Acknowledge.** This signal indicates that the microprocessor has recognized an external reset request or that it has generated an internal reset (i.e., reset exception). The microprocessor executes its reset routine once it negates $\overline{\text{RESET}}$.
$\overline{\text{RESETR}}$	F13	I	**Reset Request.** This asynchronous signal is used to reset the microprocessor. $\overline{\text{RESETR}}$ must be active for 3 clock cycles in order to be acknowledged. The microprocessor acknowledges the request by immediately asserting $\overline{\text{RESET}}$. (See Reset.)

Table 25. Bus Exception Signals (Continued)

Name	Pin	Type	Description
$\overline{\text{RETRY}}$	D11	I	**Retry.** When this signal is asserted, the microprocessor terminates the current bus transaction and retries it when $\overline{\text{RETRY}}$ is negated.
$\overline{\text{RRRACK}}$	C12	O	**Relinquish and Retry Request Acknowledge.** This signal is asserted in response to a relinquish and retry bus exception when the microprocessor has relinquished the bus (3-stated the bus). This open-drain output is passively pulled high by an external pull-up resistor when the bus transaction terminated by the relinquish and retry bus exception is retried.
$\overline{\text{RRREQ}}$	B13	I	**Relinquish and Retry Request.** This signal is used to preempt a bus transaction so that the microprocessor bus may be used. The signal causes the microprocessor to terminate the current bus transaction and to 3-state the following pins: $\overline{\text{ABORT}}$ DATA00—DATA31 SAS0—SAS2 ADDR00—ADDR31 $\overline{\text{DRDY}}$ $\overline{\text{VAD}}$ $\overline{\text{AS}}$ $\overline{\text{DS}}$ R/$\overline{\text{W}}$ $\overline{\text{CYCLE}}$ SAS0—SAS3 XMD0, XMD1 The $\overline{\text{RRRACK}}$ signal is asserted after all the above-mentioned pins are 3-stated. During this 3-state phase, the bus master requesting the relinquish and retry may take possession of the bus. No external bus arbitration signals are acknowledged during the assertion of $\overline{\text{RRREQ}}$. When $\overline{\text{RRREQ}}$ is negated, the preempted bus transaction is retried.
$\overline{\text{STOP}}$	G13	I	**Stop.** When asserted, this asynchronous signal halts the execution of any further instructions beyond those already started. At most, there may be one more instruction beyond the instruction during which $\overline{\text{STOP}}$ was asserted before the microprocessor comes to halt.

Table 26. Development System Support Signals

Name	Pin	Type	Description
$\overline{\text{HIGHZ}}$	C11	I	**High Impedance.** This signal puts all output pins on the microprocessor into the high-impedance state when asserted. This pin is intended for testing purposes.
IQS0, IQS1	A5, A8	O	**Instruction Queue Status.** This two-bit code indicates the activity on the microprocessor instruction queue: **IQS1** **IQS0** **Description** 0 0 Discard 4 bytes 0 1 Discard 1 byte 1 0 Discard 2 bytes 1 1 No discard this cycle
SOI	B7	O	**Start of Instruction.** When asserted, this signal indicates that the microprocessor's internal control has fetched the opcode for the next instruction from the internal instruction queue. Since the instructions are pipelined, it does not always mean the end of the previous instruction execution.

Table 27. Clock Signals

Name	Pin	Type	Description
CLK34	G11	I	**Input Clock.** The falling edge of this clock signifies the beginning of a machine cycle. This clock input has the same frequency as CLK23 and lags it by 90°.
CLK23	F11	I	**Input Clock.** This clock input has the same frequency as CLK34 and leads it by 90°.

Characteristics

V_{CC} = 5.0 V ± 5%, V_{SS} = 0 V, C_L = 130 pF, T_A = 0 to 70 °C

Timing Characteristics

All TTL timing specifications are referenced to and from 0.8 V for a low voltage and 2.0 V for a high voltage. CMOS clock references are to and from $V_{CC}/2$. All timing information is subject to change. All min and max values are in ns.

Table 28. Timing Characteristics

				10 MHz		14 MHz		18 MHz	
Num	Symbol	Description*	Fig(s)	Min	Max	Min	Max	Min	Max
1	tDATVC34H	Data set-up time	7	13	—	7	—	0	—
2A	tDSBHDATX	Data hold time	7	0	—	0	—	0	—
2B	tC34HDATX	Data hold time	7	23	—	16	—	14	—
3	tSRYLC34L	Synchronous ready set-up time	7	16	—	7	—	8	—
4	tC34LSRYH	Synchronous ready hold time	7	23	—	16	—	11	—
5	tDTALC34H	Data transfer ack set-up time**	7	15	—	10	—	8	—
6	tDSBHDTAH	Data transfer ack hold time**	7	0	—	0	—	0	—
7	tDSHLG01Z	Outputs 3-stated time	—	—	72	—	50	—	39
8	tDSHHG01V	Outputs valid time	—	—	70	—	50	—	39
9	tIPLVC23L	Interrupt priority level set-up time**	7	15	—	10	—	8	—
10	tIPLVSASV	Interrupt priority level valid time	—	0	—	0	—	0	—
11	tIOPLC23L	Interrupt option set-up time**	7	15	—	10	—	8	—
12	tIOPTLSASV	Interrupt option assertion time	—	0	—	0	—	0	—
13	tNMILC23L	Nonmaskable interrupt set-up time**	7	15	—	10	—	8	—
14	tNMILSASV	Nonmaskable interrupt assertion time	—	0	—	0	—	0	—
15	tBFTLC34L	Block (double-word) fetch set-up time**	10	15	—	10	—	8	—
16A	tDSBHBFTH	Block (double-word) fetch asserted time†	10	0	—	0	—	0	—

* Ack = Acknowledge.
** These are the synchronous signals; set-up times are specified for testing and informational purposes. The set-up times guarantee recognition at the next clock edge.
† Required only in systems using the *WE* 32101 Memory Management Unit.

Table 28. Timing Characteristics (Continued)

Num	Symbol	Description*	Fig(s)	10 MHz		14 MHz		18 MHz	
				Min	Max	Min	Max	Min	Max
16B	tC34LBFTH	Block (double-word) fetch hold time	10	20	—	14	—	11	—
17	tRTYLC34L	Synchronous retry set-up time	9	15	—	10	—	8	—
18A	tDSBHRTYH	Delayed retry hold time**	17	0	—	0	—	0	—
18B	tC34LRTYH	Synchronous retry hold time	9	20	—	14	—	11	—
19	tRRRLC34L	Synchronous relinquish and retry request set-up time	9	15	—	10	—	8	—
20A	tRRALRRRH	Delayed relinquish and retry request hold time**	18	0	—	0	—	0	—
20B	tC34LRRRH	Synchronous relinquish and retry request hold time	9	20	—	14	—	11	—
21	tFATLC34L	Synchronous fault set-up time	9	15	—	10	—	8	—
22A	tDSBHFATH	Delayed fault hold time**	19	0	—	0	—	0	—
22B	tC34LFATH	Synchronous fault hold time	9	20	—	14	—	11	—
23	tBRQLC23H	Bus request set-up time†	13	15	—	10	—	8	—
24	tBRALBRQH	Bus request hold time	13	0	—	0	—	0	—
25	tRSRLC23L	Reset request set-up time†	15	15	—	10	—	8	—
26	tC23LRSRH	Reset request valid time to guarantee reset	15	3Tc	—	3Tc	—	3Tc	—
27	tRTYLC34H	Asynchronous retry set-up time†	20	15	—	10	—	8	—
28	tRRRLC34H	Asynchronous relinquish and retry request set-up time†	20	15	—	10	—	8	—
29	tFATLC34H	Asynchronous fault set-up time†	20	19	—	11	—	9	—
30	tSTPLC23L	CPU stop set-up time†	12	15	—	10	—	8	—
31	tSASVSTPH	CPU stop valid time	12	0	—	0	—	0	—
32	tAVCLC23L	Autovector set-up time†	7	15	—	10	—	8	—
33	tAVCVSASV	Autovector valid time	—	0	—	0	—	0	—
34	tBRALC23H	Bus request ack set-up time†,††	—	15	—	10	—	8	—
35	tBRALBRQH	Bus request ack hold time†	—	0	—	0	—	0	—
36	tDONLC23H	Slave processor done set-up time	16	15	—	10	—	8	—
37	tSASVDONH	Slave processor done hold time	16	0	—	0	—	0	—
38	tDSBHRTYH	Asynchronous retry hold time	20	0	—	0	—	0	—
39	tDSBHRRRH	Asynchronous relinquish and retry request hold time	20	0	—	0	—	0	—
40	tDSBHFATH	Asynchronous fault hold time	20	0	—	0	—	0	—
41	tC34LADDV	Address assertion time	7	—	50	—	35	—	26

* Ack = Acknowledge.
** Required only in systems using the *WE* 32101 Memory Management Unit.
† These are the synchronous signals; set-up times are specified for testing and informational purposes. The set-up times guarantee recognition at the next clock edge.
†† These specifications are valid when the CPU is the slave ($\overline{BARB}$ = 1).

Table 28. Timing Characteristics (Continued)

Num	Symbol	Description*	Fig(s)	10 MHz		14 MHz		18 MHz	
				Min	Max	Min	Max	Min	Max
42	tC23HADDZ	Virtual address 3-state time	7	—	68	—	35	—	24
43	tC34LADDZ	Address 3-state time	13	—	72	—	55	—	43
44	tC34LDATV	Data assertion time	8	—	61	—	40	—	31
45	tC34LDATZ	Data 3-state time	13	—	72	—	51	—	40
46	tC34LSASV	Access status code assertion time	7	—	62	—	48	—	37
46A	tBRQHSASV	Access status code driven time	13	0.75Tc	—	0.75Tc	—	0.75Tc	—
47	tC34LSASZ	Access status code 3-state	13	—	75	—	47	—	40
48	tC34LDSZV	Data size assertion time	7	—	70	—	46	—	36
49	tC34LDSZZ	Data size 3-state time	13	—	62	—	50	—	39
50	tC34HDRYL	Data ready assertion time	7	—	55	—	34	—	26
51	tC34HDRYH	Data ready negation time	7	—	55	—	40	—	31
52A	tC34HASBL	Address strobe assertion time	7	—	50	—	35	—	24
52B	tADDVASBL	Address set-up time	7	20	—	20	—	16	—
52C	tASBLADDV	Virtual address hold time	7	45	—	32	—	27	—
53A	tC34HASBH	Address strobe negation time	7	—	51	—	33	—	26
53B	tASBHADDX	Address hold time	7	34	—	19	—	15	—
54A	tC34HDSBL	Data strobe assertion time	7,8	—	53	—	35	—	27
54B	tDATVDSBL	Data set-up time	7	20	—	15	—	12	—
54C	tDSBLDSBH	Data strobe assertion (width) time	8	90	—	0.9Tc	—	0.85Tc	—
55	tC34HDSBH	Data strobe negation time	7	—	51	—	33	—	26
55B	tDSBHDATZ	Data hold time	8	135	—	96	—	77	—
55D	tDSBHDSBL	Data strobe negated between first and second fetches time	10	90	—	0.9Tc	—	0.9Tc	—
56	tC34LCYCL	Cycle initiate assertion time	7	—	57	—	37	—	30
57	tC34LCYCH	Cycle initiate negation time	7	—	56	—	36	—	28
60	tC34LR/WL	Read/write assertion time	8	—	69	—	46	—	36
61	tR/WLASBL	Read/write set-up time	8	25	—	18	—	14	—
61A	tR/WLDSBL	Read/write set-up time	8	125	—	89	—	69	—
62	tC34LBRAL	Bus request ack assertion time	13	—	57	—	41	—	30
63	tC34LBRAH	Bus request ack negation	13	—	56	—	38	—	30
64	tC34LASBZ	Address strobe 3-state time	13	—	58	—	36	—	28
65	tC34LDSBZ	Data strobe 3-state time	13	—	58	—	46	—	36
66	tC34LR/WZ	Read/write 3-state time	13	—	67	—	42	—	33
67	tC34LDRYZ	Data ready 3-state time	13	—	66	—	44	—	34
68	tC34LCYCZ	Cycle initiate 3-state time	13	—	57	—	46	—	36

* Ack = Acknowledge.

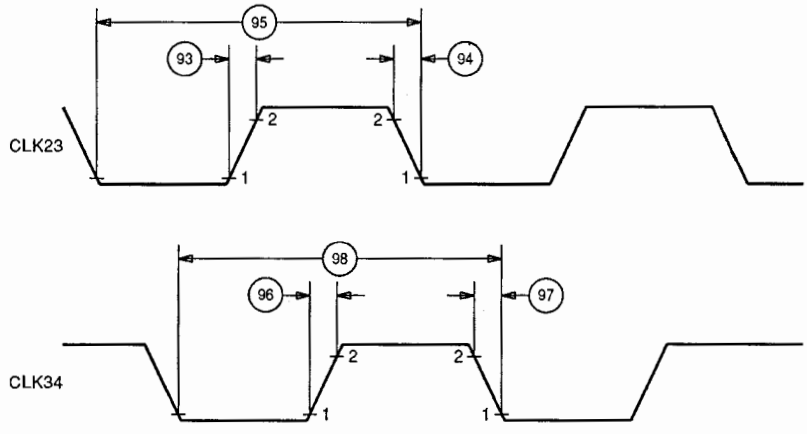
Table 28. Timing Characteristics (Continued)

Num	Symbol	Description*	Fig(s)	10 MHz		14 MHz		18 MHz	
				Min	Max	Min	Max	Min	Max
69	tC34LVADL	Virtual address assertion time	—	—	56	—	40	—	31
70	tC34HVADH	Virtual address negation time	—	—	56	—	40	—	31
71	tC34HABTL	Access abort assertion time	14	—	53	—	35	—	27
72	tC34HABTH	Access abort negation time	14	—	52	—	35	—	27
73	tC34LSOIL	Start of instruction assertion time	11	—	62	—	44	—	34
74	tC34LSOIH	Start of instruction negation time	11	—	62	—	44	—	34
75	tC34LIQSV	Instruction queue status assertion time	11	—	62	—	44	—	34
76	tC34LRSTL	Reset ack assertion time	15	—	62	—	44	—	34
77	tC34LRSTH	Reset ack negation	15	—	62	—	44	—	34
78	tC34LXMDV	Execution mode assertion	13	—	59	—	44	—	34
79	tC34LXMDZ	Execution mode 3-state time	13	—	75	—	60	—	47
80	tC34LRRAL	Relinquish and retry request ack assertion time	18	—	55	—	39	—	30
81	tC34LRRAH	Relinquish and retry request ack high-impedance time	18	—	62	—	44	—	34
82	tC34HBRQL	Bus request assertion time**	—	—	70	—	44	—	35
83	tC34HBRQH	Bus request negation time**	—	—	62	—	44	—	34
84	tHIGZOUTZ	All outputs to 3-state time	—	—	100	—	71	—	55
85	tDTALBFTL	Delayed block fetch time	—	—	Tc/2	—	Tc/2	—	Tc/2
90	tDTALRTYL	Delayed retry time	17	—	Tc/2	—	Tc/2	—	Tc/2
91	tDTALRRRL	Delayed relinquish and retry request time	18	—	Tc/2	—	Tc/2	—	Tc/2
92	tDTALFATL	Delayed fault time	19	—	Tc/2	—	Tc/2	—	Tc/2
93	tC23L1C23H2	Clock 23 rise time	6	—	4	—	4	—	4
94	tC23H2C23L1	Clock 23 fall time	6	—	4	—	4	—	4
95	tC23L1C23L1	Clock 23 period (T)	6	100	—	71.4	—	56	—
96	tC34L1C34H2	Clock 34 rise time	6	—	4	—	4	—	4
97	tC34H2C34L1	Clock 34 fall time	6	—	4	—	4	—	4
98	tC34L1C34L1	Clock 34 period (T)	6	100	—	71.4	—	56	—
99	tC23J	Clock 23 jitter	—	—	0.5	—	0.5	—	0.5
100	tC23E	Clock 23 duty cycle error	—	—	3	—	2	—	2
101	tC34J	Clock 34 jitter	—	—	0.5	—	0.5	—	0.5
102	tC34E	Clock 34 duty cycle error	—	—	3	—	2	—	2
103	tSKEW	Clock skew	—	—	3	—	2	—	2

* Ack = Acknowledge.
** These specifications are valid when the CPU is the slave ($\overline{BARB}$ = 1).

Timing Diagrams

These timing diagrams represent a subset of all possible transactions and are intended only for the purposes of displaying and clarifying timing relationships.

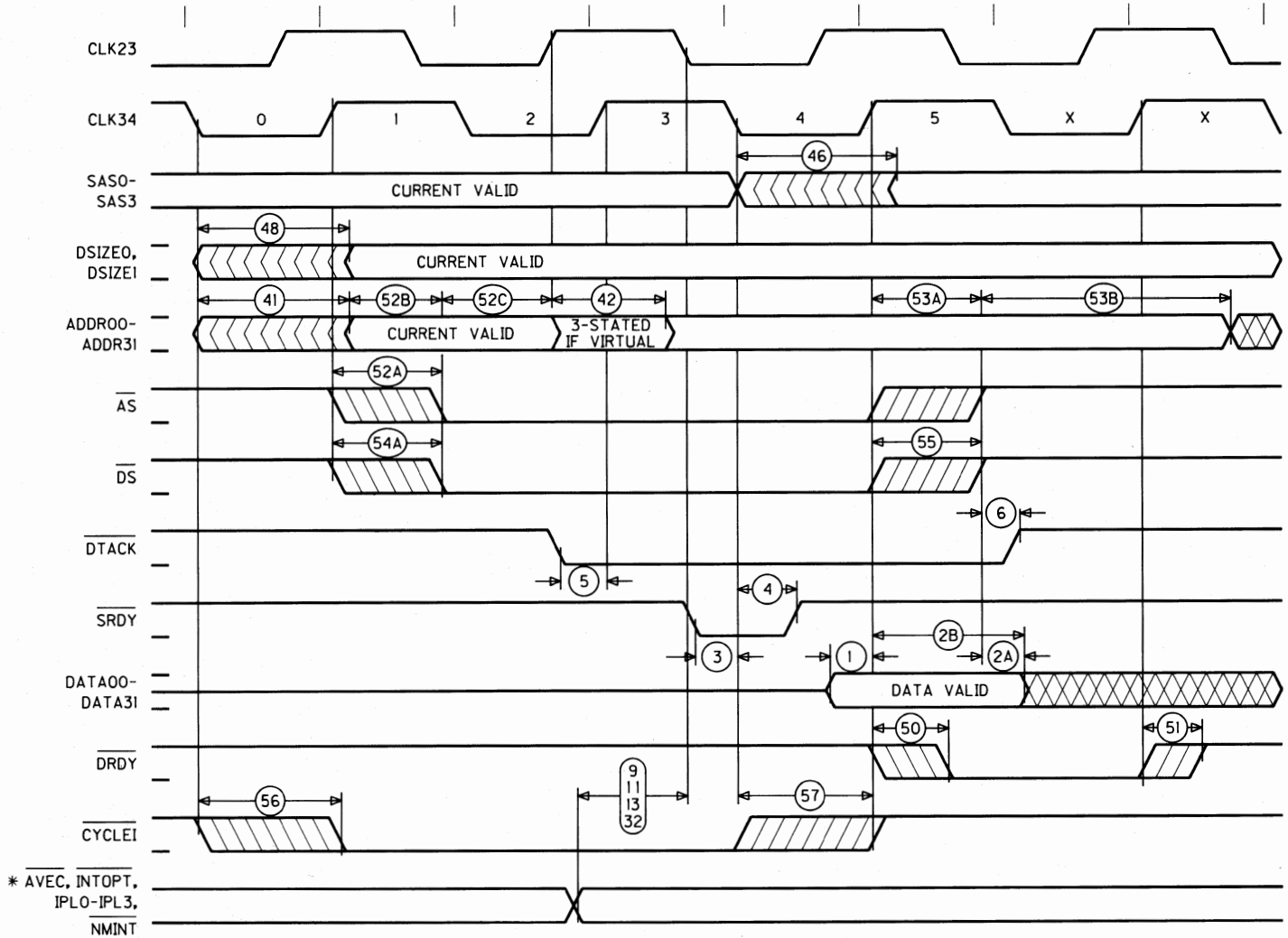


Notes:
Duty Cycle Error – The duty cycle of each clock input may deviate from 50% but should not exceed timing specification numbers 100 and 102 for CLK23 and CLK34, respectively.

Skew – CLK23 nominally leads CLK34 by 90° (1/4 clock period). This phase lead should never exceed timing specification number 103.

Jitter – The period of each clock input may deviate from its nominal value but should not exceed timing specification numbers 99 or 101 for CLK23 and CLK34, respectively.

Figure 6. Clock Inputs



* These signals must be held valid until the processor acknowledges their reception via the SAS codes assigned for this purpose.

Figure 7. Read Timing

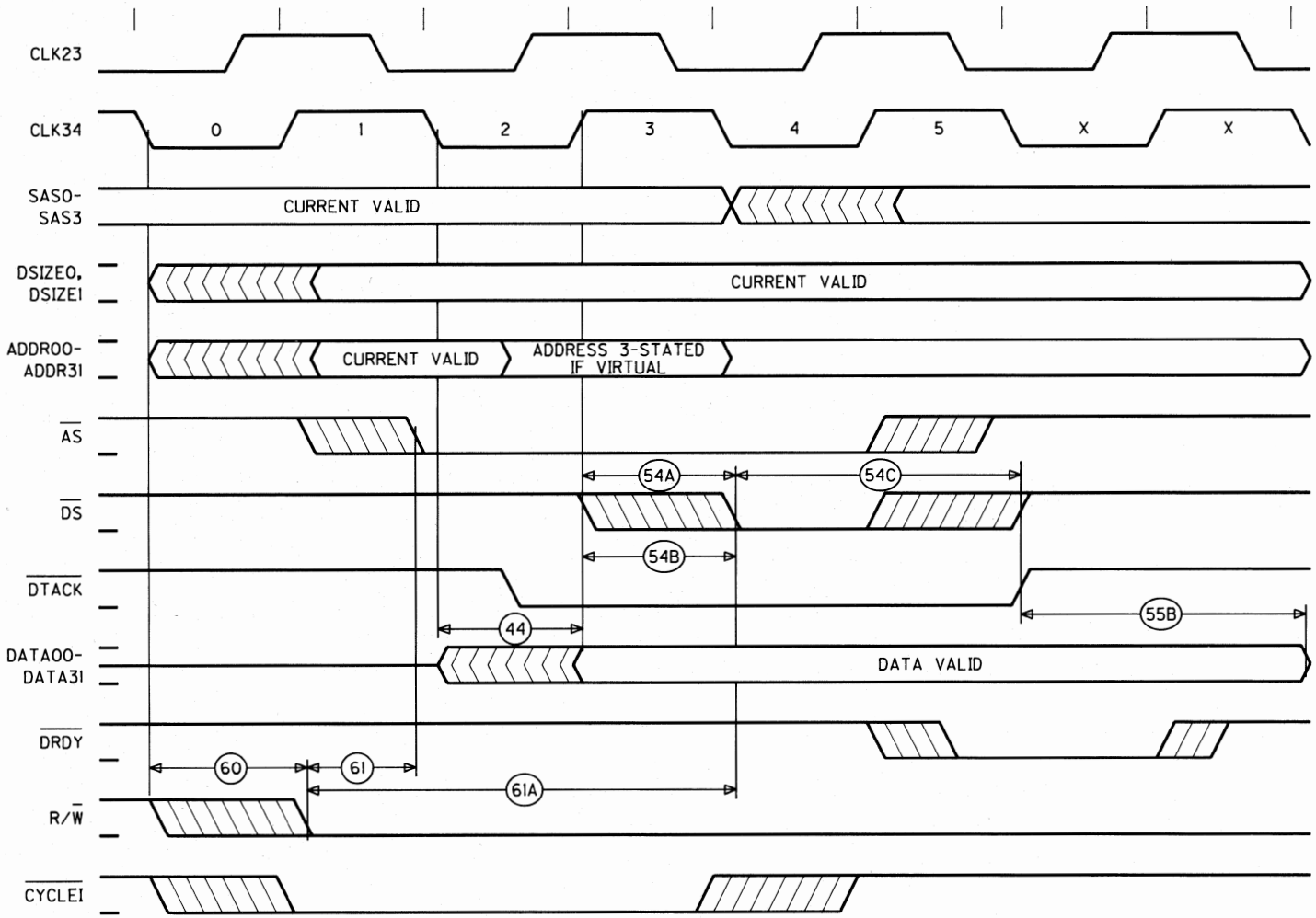


Figure 8. Write Timing

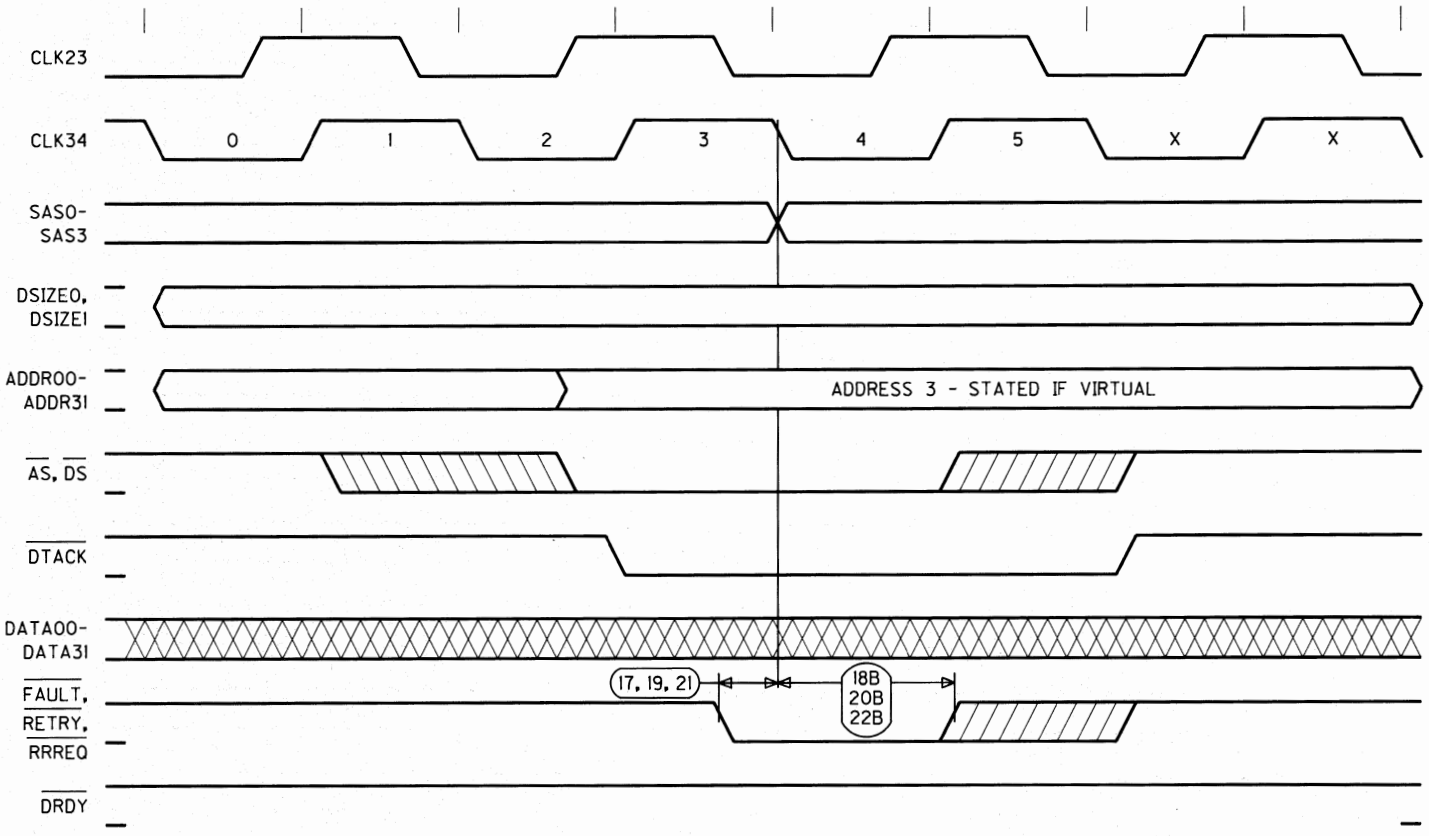


Figure 9. Bus Exceptions (FAULT, RETRY, RREQ) Timing

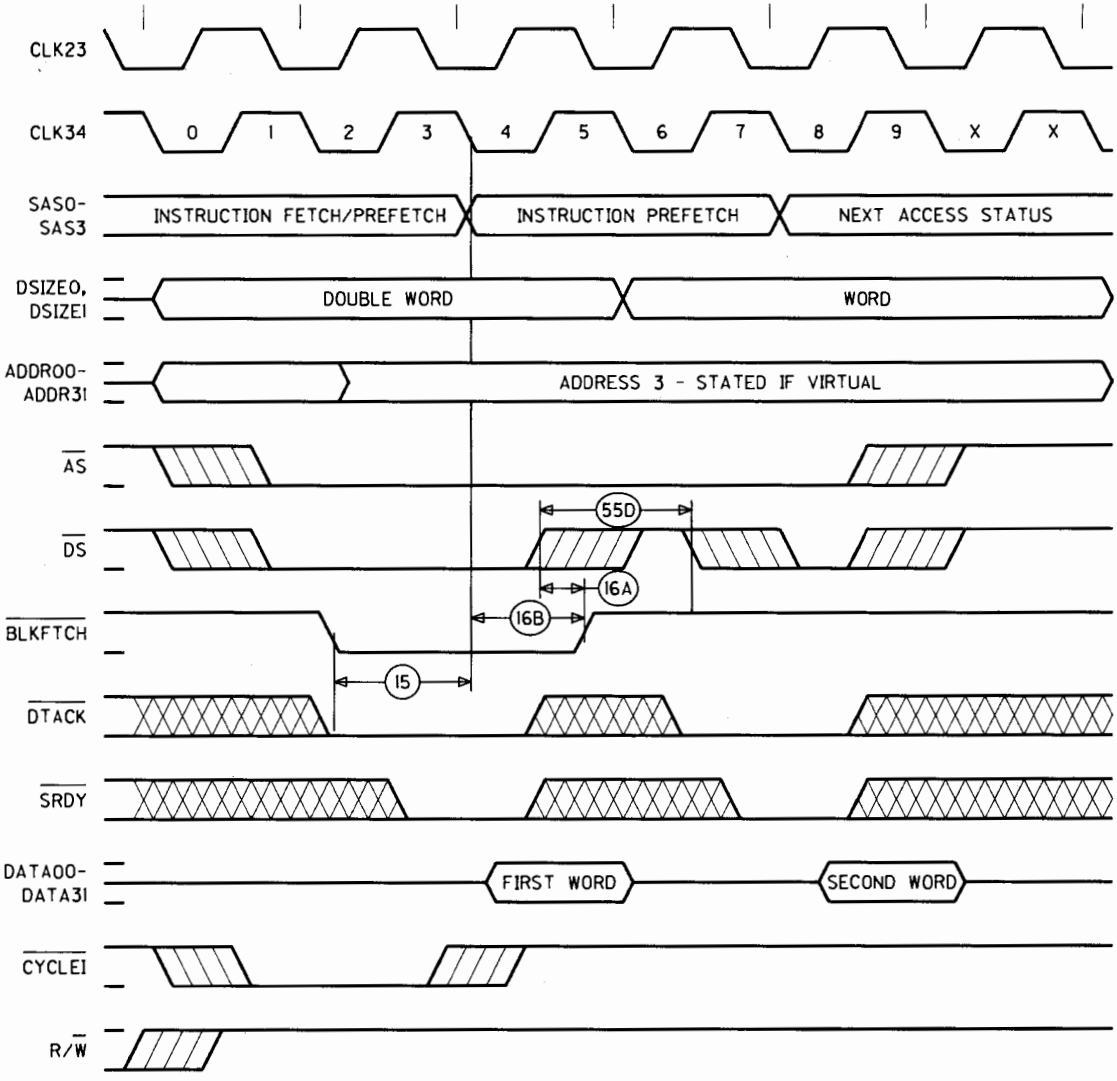


Figure 10. Block (Double-Word) Fetch Timing

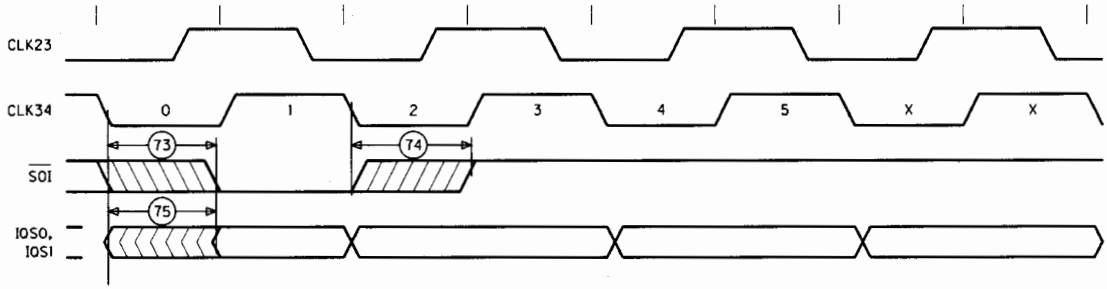
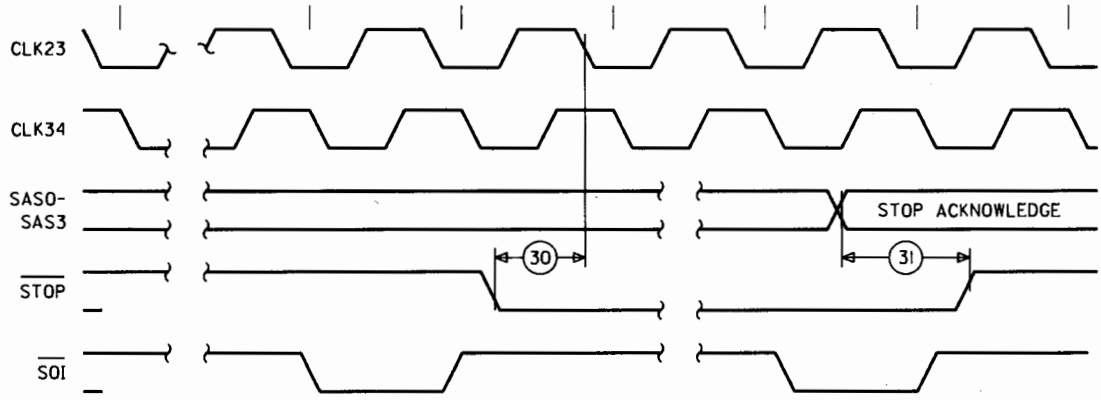


Figure 11. Start of Instruction and Instruction Queue Status Timing



Notes:
There is at most one assertion of $\overline{SOI}$ before $\overline{STOP}$ is acknowledged on SAS0—SAS3 pins.
$\overline{STOP}$ must be asserted with the $\overline{STOP}$ acknowledge status valid.
$\overline{STOP}$ must be inactive for at least one cycle for the CPU to proceed to the next access.

Figure 12. Assertion of $\overline{STOP}$ after $\overline{SOI}$ Timing

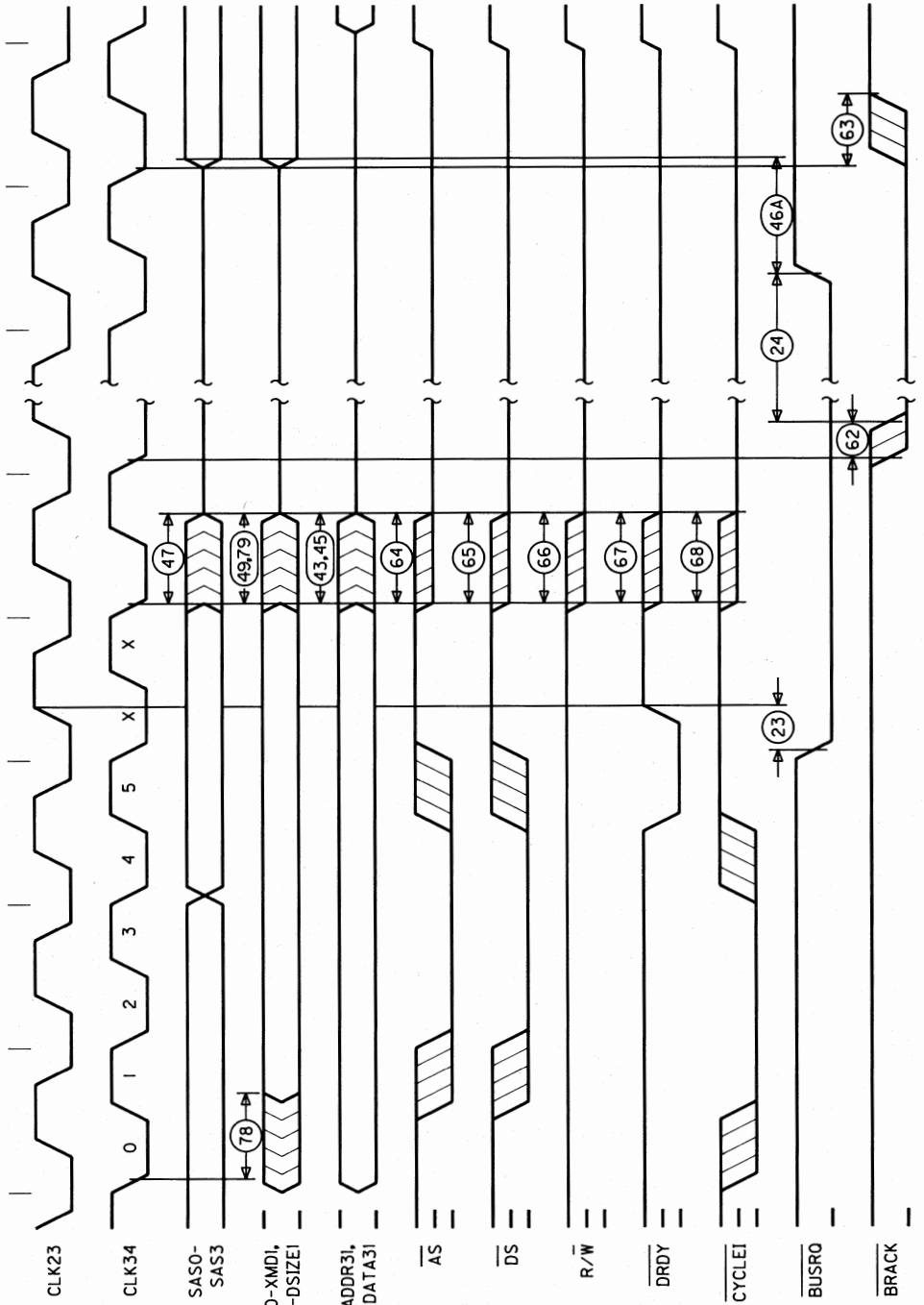


Figure 13. Bus Arbitration Timing

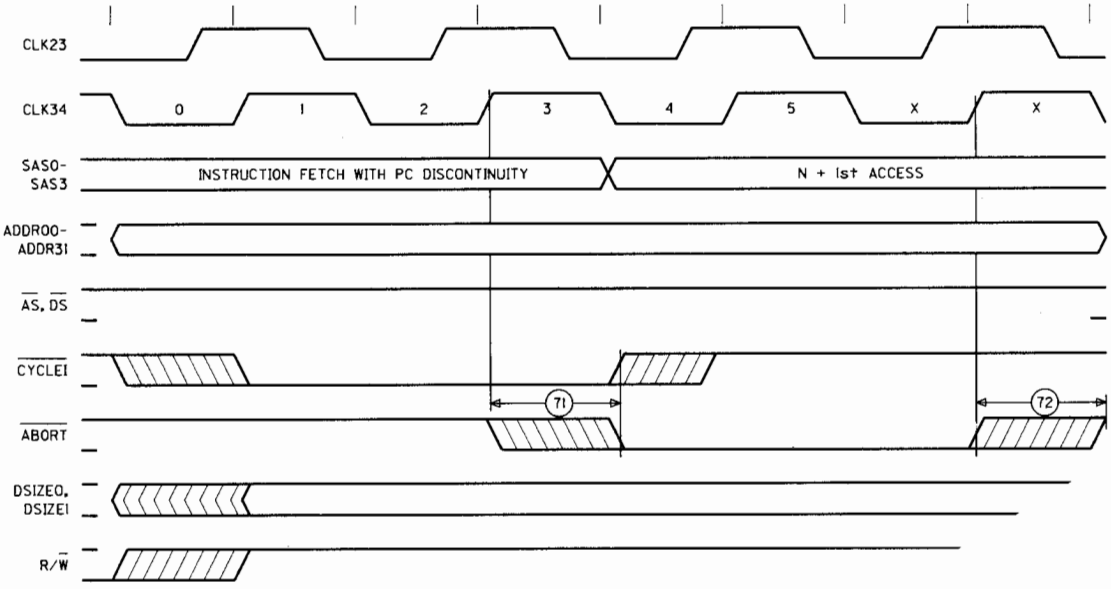
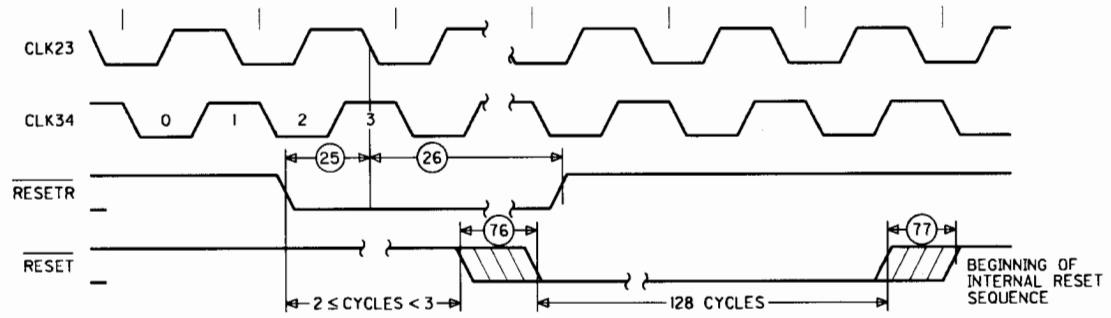


Figure 14. Aborted Access Timing



Notes:
RESETR must be asserted for at least 3 cycles to be recognized.
RESET is negated 128 cycles after negation of RESETR.

Figure 15. Reset Timing

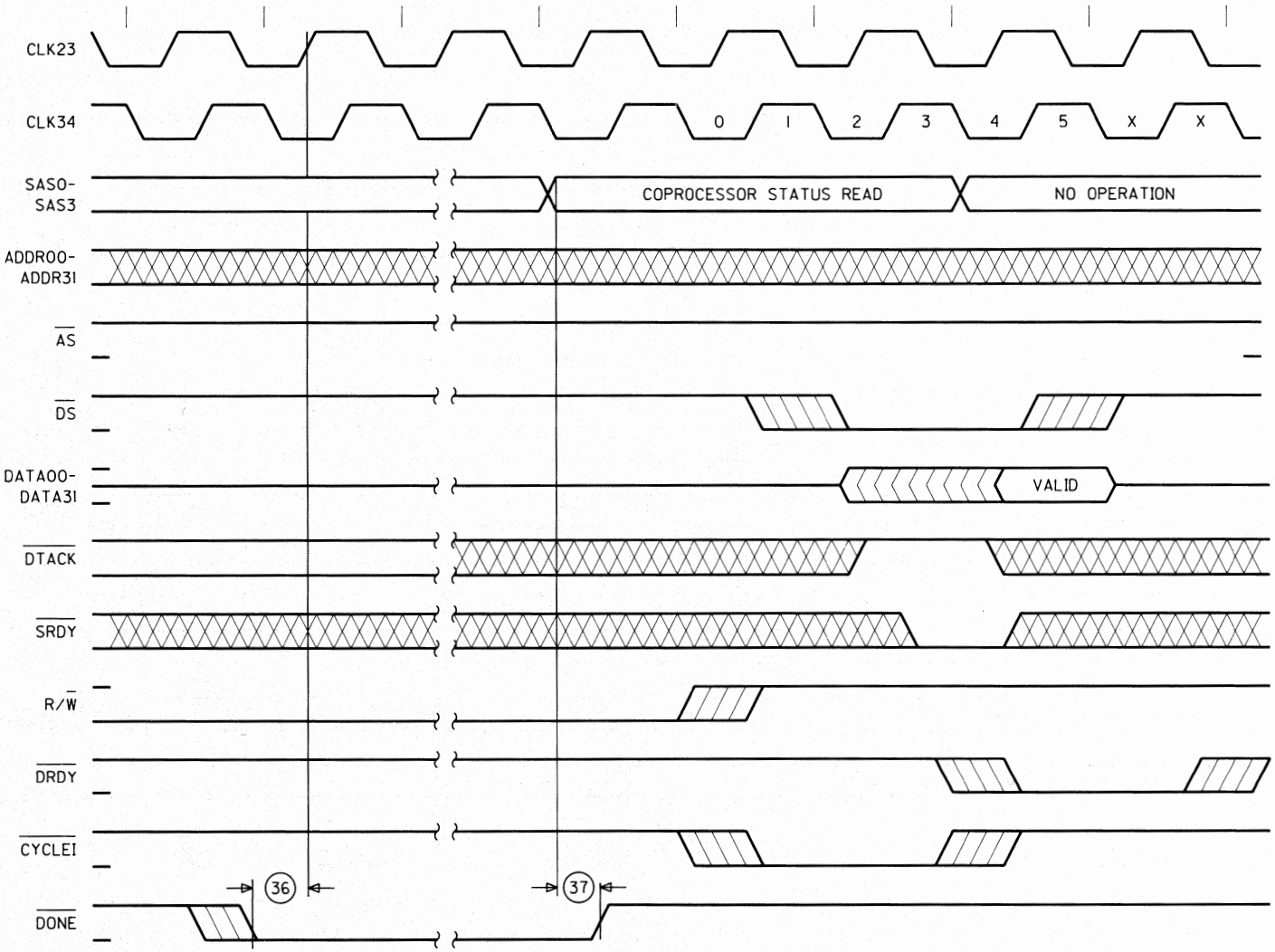
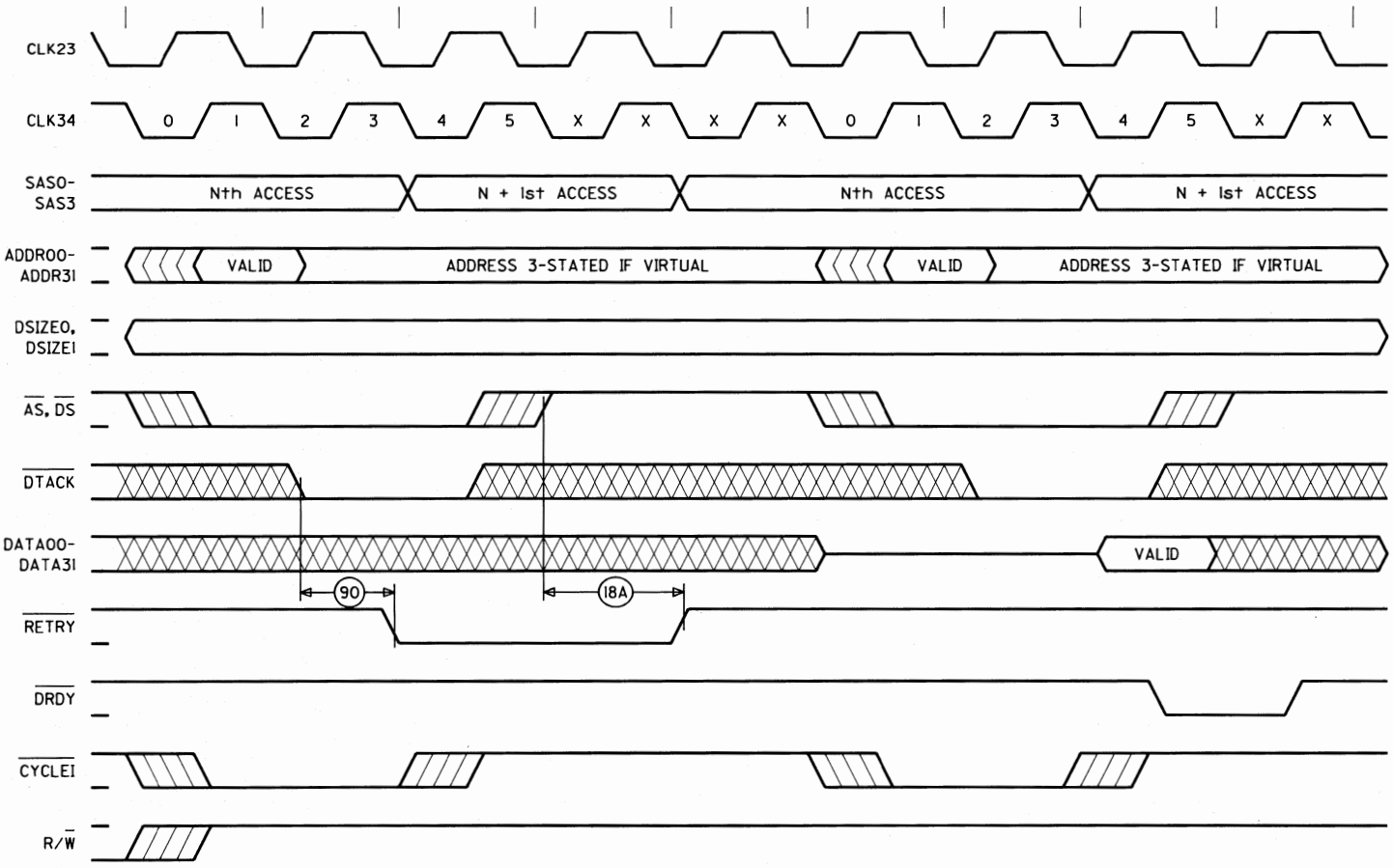
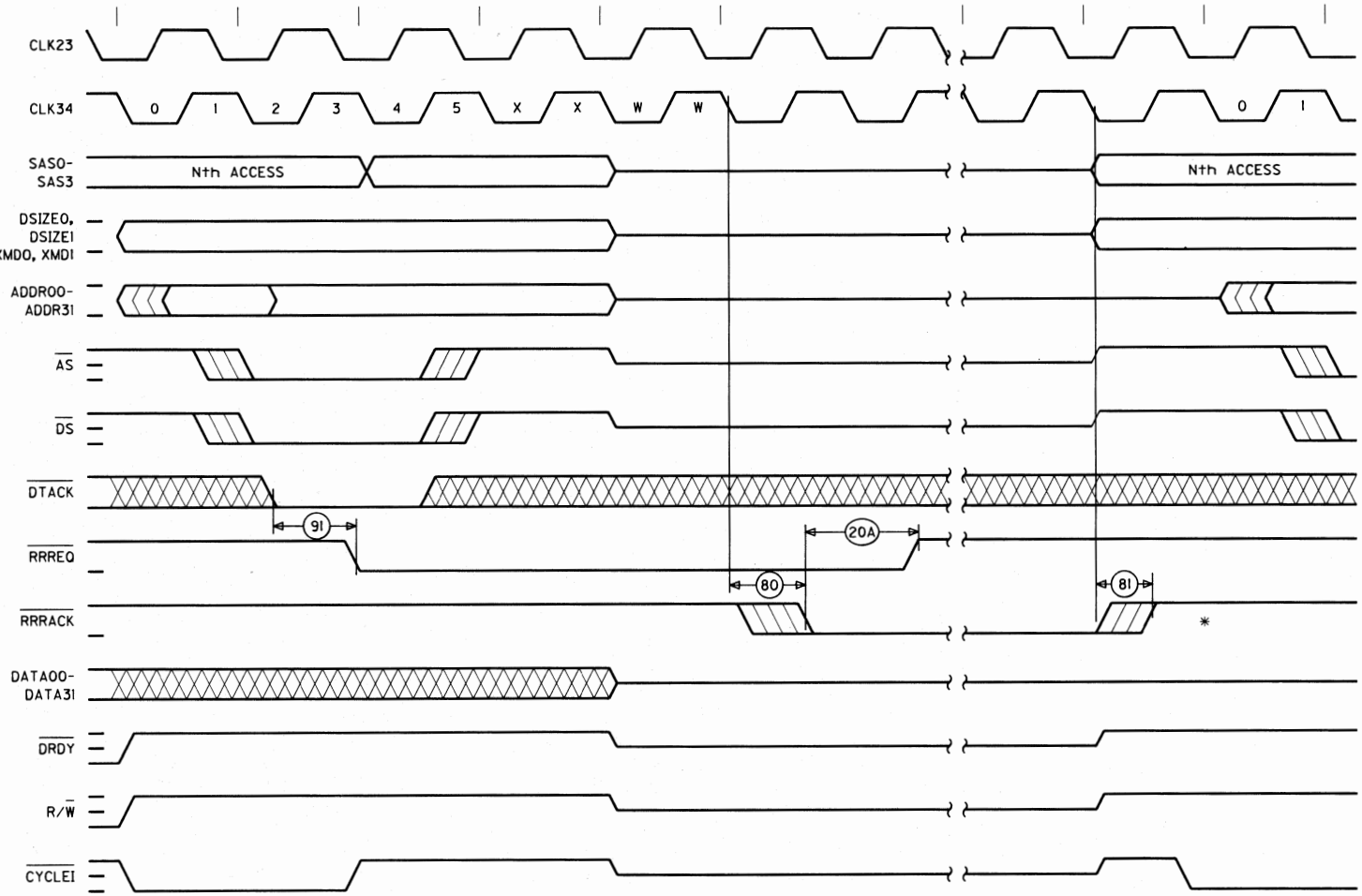


Figure 16. Coprocessor Status Read After $\overline{\text{DONE}}$ Timing



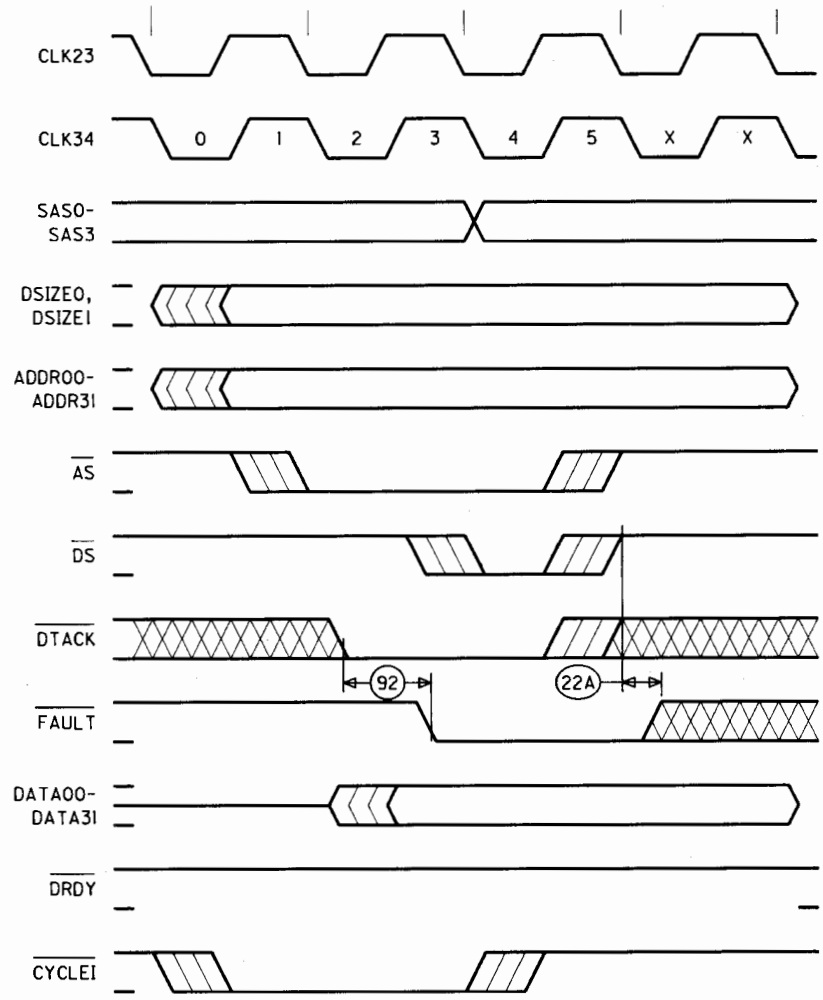
Note: Between the time $\overline{\text{RETRY}}$ is negated and state zero of the retried access, 2.5 to 3.5 cycles elapse.

Figure 17. Retry After $\overline{\text{DTACK}}$ Timing (Read Transaction is Shown)



* Shown pulled high by external resistor.

Figure 18. Relinquish and Retry after $\overline{\text{DTACK}}$ Timing



Notes:
FAULT must meet set-up time with respect to the falling edge of clock states 3 and 4 (of CLK34) after DTACK.
SRDY is don't care at zero wait states.

Figure 19. Fault After DTACK Timing (Write Transaction is Shown)

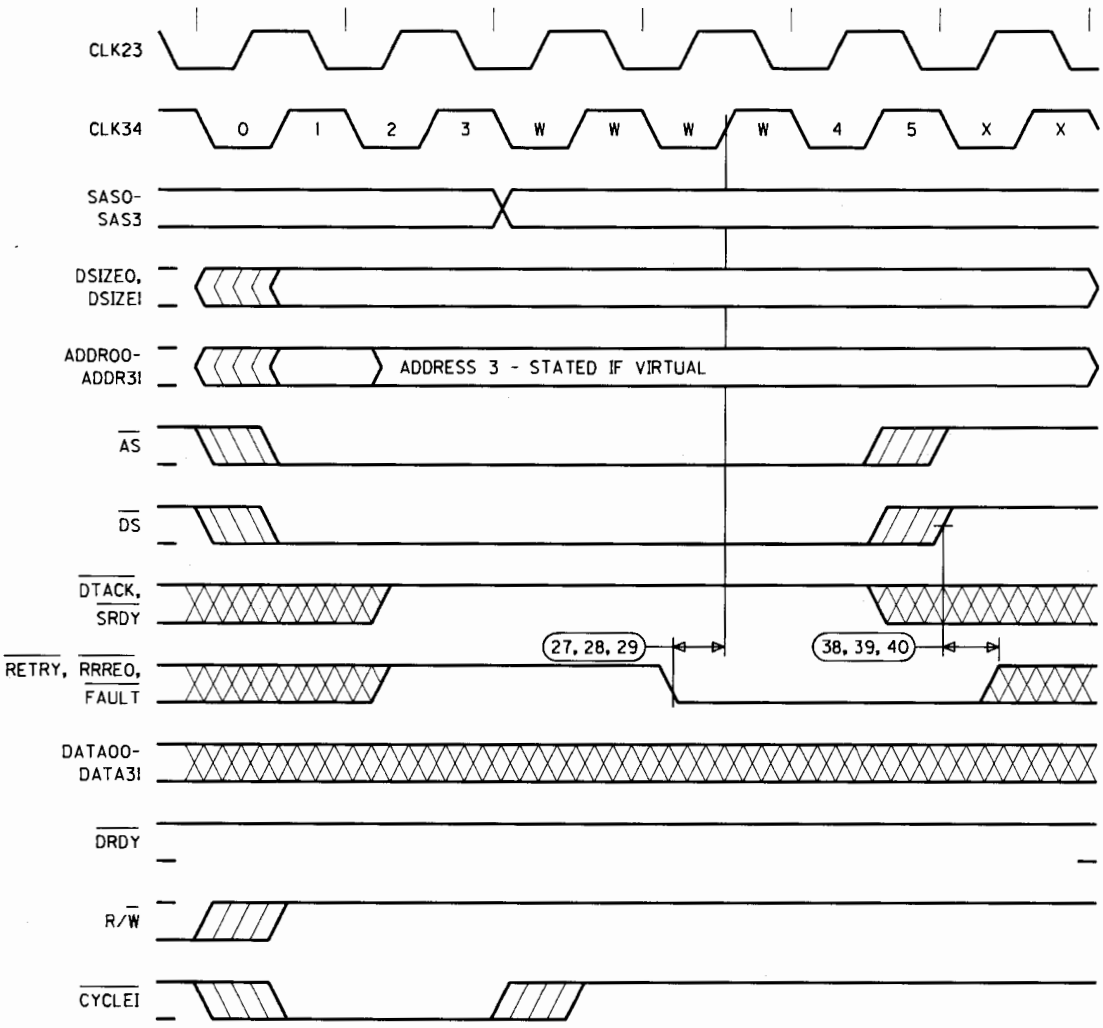


Figure 20. Bus Exception ($\overline{\text{RETRY}}$, $\overline{\text{FAULT}}$, or $\overline{\text{RRREQ}}$) with no $\overline{\text{DTACK}}$ or $\overline{\text{SRDY}}$ Timing (Read Transaction is Shown)

Electrical Characteristics

Inputs

All inputs, except the two CMOS input clocks, are TTL-compatible. The TTL inputs and clock inputs are specified separately below.

Table 29. DC Input Parameters

Inputs		Min	Max	Unit
TTL input voltage	high-level	2.0	V_{CC} +0.5	V
	low-level	–0.5	0.8	V
CMOS clocks input voltage	high-level	V_{CC} –1.3	V_{CC} +0.5	V
	low-level	0	0.8	V
TTL input loading current ($2.0\ V \leq V_{IH} \leq V_{CC}$)	high-level	0	0.01	mA
TTL input loading current ($0\ V \leq V_{IL} \leq 0.8\ V$)	low-level	–0.01	0	mA
CMOS clocks input loading current ($V_{CC} - 1.5\ V \leq V_{IH} \leq V_{CC}$)	high-level	0	0.01	mA
CMOS clocks input loading current ($0 \leq V_{IL} \leq 1.0$)	low-level	–0.01	0	mA

Outputs

The four classes of outputs that can support TTL input voltage levels are:

Class 1: This class is capable of driving one TTL load or 8 PNP Schottky TTL loads and has current allowance for an external holding resistor employed in 3-state buffers. The minimum holding resistor value is 2.7 KΩ.

Class 2: This class is the same as Class 1 except that it does not have current allowance for a holding resistor.

Class 3: The signal in this class is an open drain output used for wired-logic operations, allowing more than one device to drive a node without conflict. An external resistor is required to pull this signal high. The minimum pull-up resistor value is 680 Ω.

Class 4: This class is the same as Class 1; however, its minimum holding resistor value is 1.8 KΩ.

The following lists the outputs assigned to each class:

Class 1			Class 2	Class 3	Class 4
$\overline{ABORT}$	$\overline{DRDY}$	R/$\overline{W}$	ADDR00—ADDR31	$\overline{RRRACK}$	SAS0—SAS3
$\overline{AS}$	$\overline{DS}$	$\overline{RESET}$	DATA00—DATA31		
$\overline{BRACK}$	DSIZE0,	$\overline{SOI}$			
$\overline{BUSRQ}$	DSIZE1	$\overline{VAD}$			
$\overline{CYCLEI}$	IQS0, IQS1	XMD0, XMD1			

Table 30. DC Output Parameters

Outputs		Min	Max	Unit
Output sink current (I_{OL}) ($V_{OL} \leq 0.4$ V)	Class 1	—	5.5	mA
	Class 2	—	3.5	mA
	Class 3	—	10.0	mA
	Class 4	—	6.5	mA
Output source current (I_{OH}) ($V_{OH} \geq 2.4$ V)	Class 1	—	–5.5	mA
	Class 2	—	–3.5	mA
	Class 3*	—	–10.0	μA
	Class 4	—	–5.5	mA
Output logic levels	high-level	2.4	—	V
	low-level	—	0.4	V

* See explanation of Class 3.

Operating Conditions

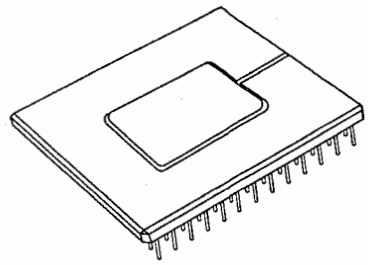
Table 31. DC Operating Conditions

Parameter		Symbol	Min	Nom	Max	Unit
Supply voltage		V_{CC}	4.75	5.00	5.25	V
Input load capacitance	TTL inputs	C_{IN}	—	—	12	pF
	CMOS clocks		—	—	7	pF
Total output load capacitance	Class 1	C_L	—	—	130	pF
	Class 2		—	—	130	pF
	Class 3		—	—	130	pF
	Class 4		—	—	130	pF
Ambient temperature at the microprocessor pins		T_A	0	—	70	°C
Humidity range		—	5%	—	95%	—
Power dissipation	at 10 MHz	PD	—	—	0.55	W
	at 14 MHz		—	—	0.77	W
	at 18 MHz		—	—	0.99	W
Operating frequency		F	—	—	18	MHz
Capacitive derating factor ($25pF \leq CL \leq 225pF$)	at 10 MHz	dt/dC	—	0.1	—	ns/pF
	at 14 MHz		—	0.1	—	ns/pF
	at 18 MHz		—	.08	—	ns/pF

WE® 32200 Microprocessor

Description

The *WE* 32200 Microprocessor (CPU) is a high-performance, single-chip, 32-bit central processing unit designed for efficient operation in a high-level language environment. It is protocol and upward object code compatible with the *WE* 32100 Microprocessor. The *WE* 32100 CPU runs object code without modification on the *WE* 32200 CPU. The *WE* 32200 CPU performs all the system address generation, control, memory access, and processing functions required in a 32-bit microcomputer system. It has separate 32-bit address and data buses. System memory is addressed over the 32-bit address bus by using either physical or virtual addresses. Data is read or written over the 32-bit bidirectional data bus in word (32-bit), halfword (16-bit), or byte (8-bit) widths, using arbitrary byte alignment for data and instructions. Dynamic bus sizing allows the *WE* 32200 CPU to communicate with both 16-bit and 32-bit memories in the same system. Extensive addressing modes result in a symmetric, versatile, and powerful instruction set. The *WE* 32200 Microprocessor is available in 24-MHz and higher frequency versions; requires only a single 5 V supply; and is in a 133-pin square, hermetic, ceramic pin grid array (PGA) package.



Features

- 32-bit virtual memory microprocessor with 4 Gbytes (2^{32}) virtual memory space and up to 4 Gbytes physical addressing space
- Efficient execution of high-level language programs
- Extensive and orthogonal instruction set with 25 addressing modes
- Arbitrary byte alignment for data and instructions
- Direct support for process-oriented operating systems, such as *UNIX* System V
- *WE* 32100 Microprocessor object code and protocol compatible
- Four levels of execution privilege: kernel, executive, supervisor, and user
- Fifteen levels of interrupt
- High-performance, on-chip, 64 x 32 bit instruction cache
- Byte replication on writes
- Dynamic bus sizing for 16-bit and 32-bit data and instructions
- Seventeen 32-bit general-purpose registers
- Eight 32-bit general-purpose privileged registers
- Seven 32-bit special-purpose registers
- Memory-mapped I/O
- Complete ANSI/IEEE Standard 754 floating-point support via the *WE* 32206 MAU Coprocessor
- Development system support signals
- General-purpose coprocessor interface
- Synchronous or asynchronous interfacing to external devices
- High priority two-wire bus arbitration through relinquish and retry

Description 65
Features 65
User Information 66
System Design 66
Interrupts 69
Bus Arbitration 71
Arbitrary Byte Alignment 72
Dynamic Bus Sizing 72
Byte Replication 75
Direct Memory Access and Multiprocessor Support 75
Coprocessor Interface 76
Semaphores/Read-Write Interlocked Support 76
Reset 77
Bus Exceptions 77
Programming 77
Processor Status Word (PSW) 77
Process Control Block (PCB) 81
Flags and Conditions 83
Addressing Modes 83
Format 1 Addressing Modes 83
Format 2 Addressing Modes 86
Exceptional Conditions 88
Normal Exceptions 88
Stack Exceptions 90
Process Exceptions 90
Reset Exceptions 90
Instruction Set by Functional Grouping 90
Arithmetic Group 91
Data Transfer Group 92
Logical Group 93
Binary Coded Decimal (BCD) Group 94
Program Control Group 95
Stack and Miscellaneous Groups 98
Coprocessor Group Instruction 98
Operating System Instructions and Microsequences 98
Other Microsequences 100
Instruction Set and Summaries by Mnemonic and Opcode 101
Pin Descriptions 110
Numerical Order 111
Functional Groups 114
Characteristics 121
Timing Characteristics 121
Timing Diagrams 126
Electrical Characteristics 142
Inputs 142
Outputs 142
Operating Conditions 143

User Information

System Design

The *WE* 32200 Microprocessor provides separate 32-bit address and data buses to the external system, eliminating the need for external multiplexing/demultiplexing. The memory or peripherals mapped into the system memory are addressed over the 32-bit address bus by using either peripheral or virtual addresses. Data is read to or written from the microprocessor over the 32-bit bidirectional data bus in word (32-bit), halfword (16-bit), or byte (8-bit) widths that can be aligned on any byte location. The microprocessor automatically extends bytes and halfwords to 32 bits for processing. For ease of memory interface, the CPU can communicate with both a 32-bit and a 16-bit port in the same system, using its dynamic bus sizing capability.

Execution speed is enhanced by an internal instruction queue and an internal instruction cache. The instruction queue is an 8-byte first-in-first-out (FIFO) queue that stores prefetched instructions. The queue is used to pipeline instructions, enabling the microprocessor to overlap the execution of instructions while tracking each separately. The instruction cache is a 64-word on-chip cache used to increase the microprocessor's performance by reducing external memory reads for instruction fetches. When an instruction fetch from memory occurs, instruction data is placed in both the instruction queue and the instruction cache. If that instruction data is needed again, it is fetched from the cache rather than from external memory, resulting in increased performance.

The microprocessor uses address and data strobes, as well as other interface and control signals, to control information flow over the system address and data buses. These signals provide timing for the latching of information and facilitate interfacing to commercial memories and peripherals. The microprocessor also accommodates wait-state generation to allow hand-shaking with slow peripherals.

The *WE* 32200 Microprocessor consists of four major sections, as shown on Figure 1: bus interface control, main controller, fetch unit, and execute unit.

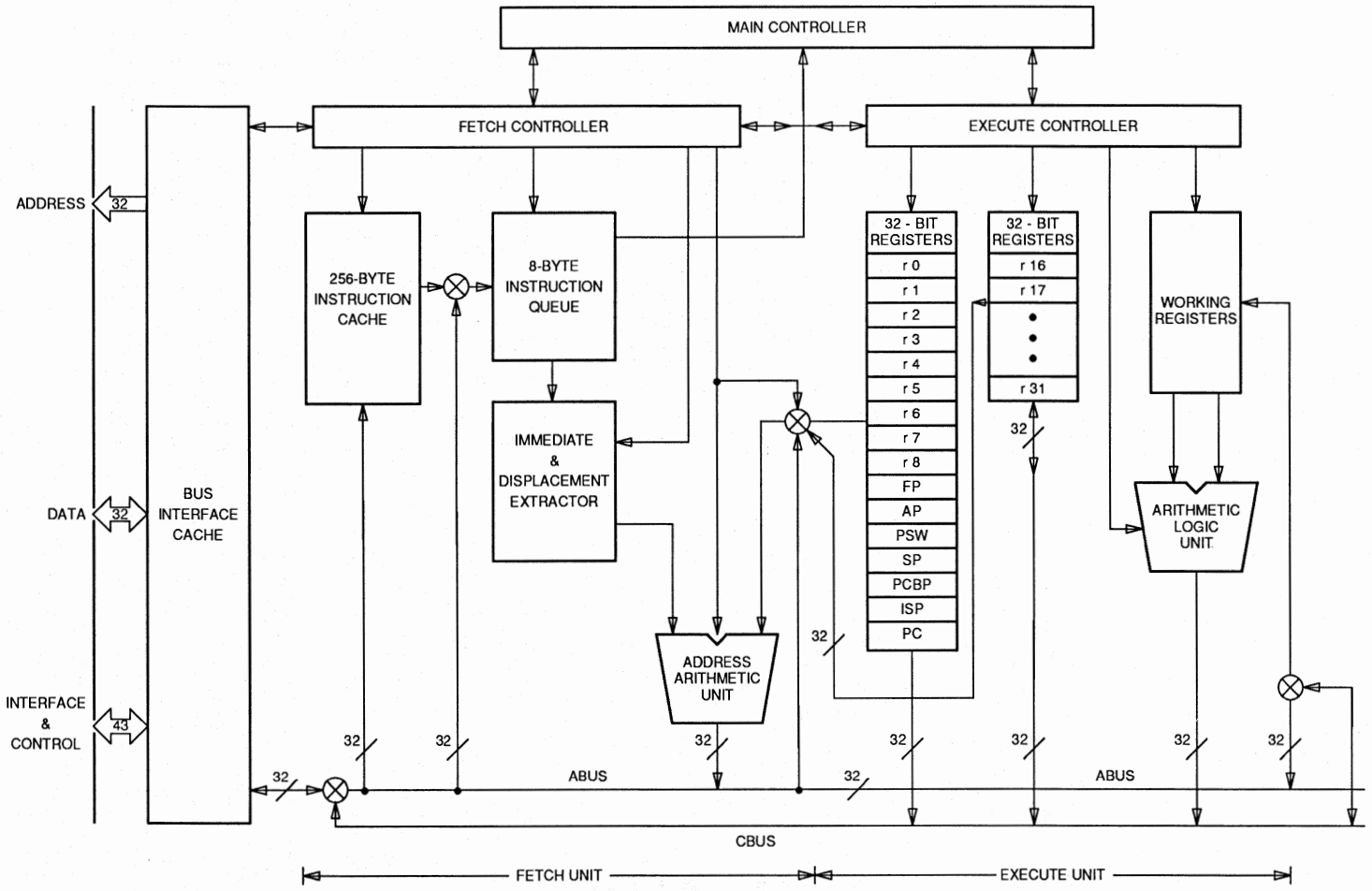


Figure 1. WE® 32200 Microprocessor Block Diagram

Bus Interface Control. Provides all the strobes and control signals necessary to implement the interface with peripherals.

Main Controller. Responsible for acquiring and decoding instruction opcodes and for directing the action of the fetch and execute controllers as the specified instruction is executed. The main controller also responds to and directs the handling of interrupts and exceptional conditions.

Fetch Unit. Handles the instruction stream and performs memory-based operand accesses. It consists of a fetch controller, an instruction cache, an instruction queue, an immediate and displacement extractor, and an address arithmetic unit (AAU).

Execute Unit. Performs all arithmetic, binary coded decimal (BCD), logical operations, and shift and rotate operations, and computes condition flags. It consists of an execute controller, thirty-two 32-bit registers, working registers, and a 33-bit-wide arithmetic logic unit (ALU). The thirty-two 32-bit registers are user-accessible and include seventeen general-purpose registers (r0—r8, r16—r23), eight general-purpose kernel registers (r24—r31), and seven dedicated registers (r9—r15). All of the registers except the program counter (r15) and the process status word (r11) can be referenced in all addressing modes. The general-purpose kernel registers (r24—31), processor status word (r11), process control block pointer (r13), and interrupt stack pointer (r14) are privileged registers that can be read at any time, but that can be written only when the microprocessor is in the kernel (highest) execution level. The working registers are used exclusively by the microprocessor and are not user-accessible.

The thirty-two 32-bit registers, defined below, are shown on Figure 2:

r0—r8, r16—r23 – General-Purpose Registers. These seventeen 32-bit registers can be used for accumulation, addressing, or temporary data storage. They can be used in any addressing mode by any program – privileged or nonprivileged. Registers r0, r1, and r2 are also implicitly used by certain data transfer and operating system instructions.

r9 – Frame Pointer (FP). Points to the beginning location in the stack of a function's local variables.

r10 – Argument Pointer (AP). Points to the beginning location in the stack where a set of arguments for a function have been pushed.

r11 – Processor Status Word (PSW). Contains information that determines the current execution state. This information includes current exception type; an internal state code; the current interrupt priority level; trace enable, trace mask; cache disable, cache flush disable; quick-interrupt enable and enable overflow trap flags; previous and current execution levels; five condition codes, indicating zero (Z), overflow (V), negative (N), carry (C), and extend carry (for BCD operations); save additional registers; arbitrary byte alignment enable; and user call process/normal exception option. The PSW is kernel-level privileged.

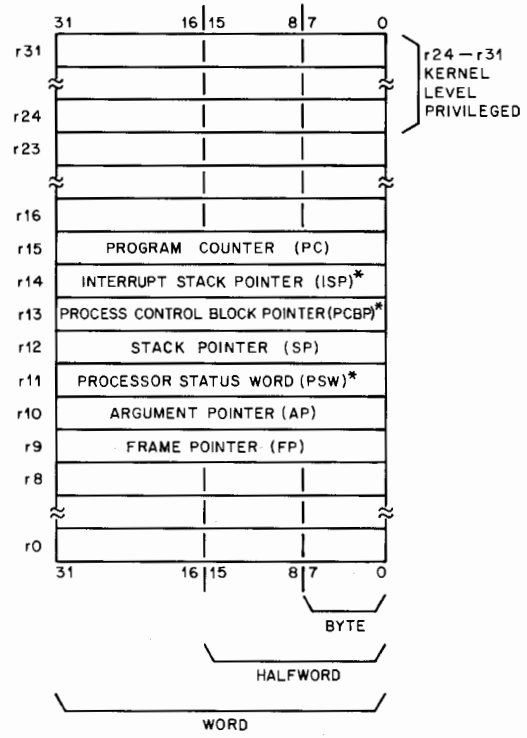
r12 – Stack Pointer (SP). Contains the current 32-bit address of the top of the execution stack, i.e., the memory address of the next place in which an item can be stored (pushed) on the stack or of the last place from which an item was retrieved (popped) from the stack. The stack pointer implements a last-in-first-out (LIFO) queue, which supports efficient subroutine linkage and local variable storage.

r13 – Process Control Block Pointer (PCBP). Contains the 32-bit address of the process control block (PCB) for the current process. The PCBP is kernel-level privileged.

r14 – Interrupt Stack Pointer (ISP). Contains the 32-bit memory address of the top of the interrupt stack. This stack is used when an interrupt request is received and when the call process (CALLPS) and return to process (RETPS) instructions are encountered. The ISP is kernel-level privileged.

r15 – Program Counter (PC). Contains the 32-bit memory address of the instruction being executed or, upon instruction completion, the starting address of the next instruction to be executed.

r24—r31 – General-Purpose Kernel Registers. These eight registers can be used for accumulation, for addressing, or for temporary storage. They are kernel-level privileged and can be used in any addressing mode by any privileged program.



* Kernel-level privileged.

Figure 2. WE® 32200 Microprocessor CPU Registers

Interrupts

The microprocessor accepts fifteen levels of interrupt requests from a 4-bit interrupt request input port (IPL0—IPL3); it also accepts a nonmaskable interrupt if $\overline{\text{NMINT}}$ is externally asserted. The pending interrupt value input on IPL0—IPL3 is internally inverted and compared to the value contained in the interrupt priority level (IPL) field of the PSW. In order for the pending interrupt to be acknowledged, its inverted value must be greater than the IPL field value. Pending interrupts whose inverted values are less than or equal to the IPL field value are ignored. The exception to this is a nonmaskable interrupt, which can interrupt the microprocessor regardless of the present interrupt priority level.

The microprocessor acknowledges an interrupt by transmitting the inverted interrupt value on bits 2 through 5 of the address bus. In addition, the value placed on the interrupt option ($\overline{\text{INTOPT}}$) pin is inverted and transmitted on bit 6 of the address bus. The microprocessor then fetches the interrupt vector number from the interrupting device on bits 0 through 7 of the data bus and begins execution of the interrupt handling routine.

Table 1 shows the interrupt priority levels and their corresponding interrupt acknowledge output by the microprocessor.

Autovector, nonmaskable-interrupt, and quick-interrupt facilities are also available. The microprocessor provides the vector number for the interrupting device if the autovector ($\overline{\text{AVEC}}$) input is asserted at the same time as the interrupt request. The vector number is dependent on the interrupt's priority level. If the

nonmaskable interrupt ($\overline{\text{NMINT}}$) input is asserted, the interrupt occurs but is treated as an autovector interrupt with vector number 0. During the acknowledge cycle of a nonmaskable interrupt, the microprocessor address bus contains all zeros. This distinguishes a nonmaskable interrupt acknowledge from all other interrupt acknowledges. A nonmaskable interrupt can interrupt the microprocessor regardless of the current priority. Interrupts are handled via the quick-interrupt sequence if the quick-interrupt enable (QIE) bit in the PSW is set. Table 2 summarizes how the microprocessor handles the various interrupt requests.

Table 1. Interrupt Level Code Assignments

Interrupt Request Input IPL0—IPL3 Bits 3	2	1	0	Interrupt Option Input $\overline{\text{INTOPT}}$	Interrupt Acknowledge Output ADDR02—ADDR06 Bits 06	05	04	03	02	Priority Level Descending Order
0	0	0	0	0	1	1	1	1	1	Highest priority
0	0	0	0	1	0	1	1	1	1	
0	0	0	1	0	1	1	1	1	0	2nd
0	0	0	1	1	0	1	1	1	0	
0	0	1	0	0	1	1	1	0	1	3rd
0	0	1	0	1	0	1	1	0	1	
0	0	1	1	0	1	1	1	0	0	4th
0	0	1	1	1	0	1	1	0	0	
0	1	0	0	0	1	1	0	1	1	5th
0	1	0	0	1	0	1	0	1	1	
0	1	0	1	0	1	1	0	1	0	6th
0	1	0	1	1	0	1	0	1	0	
0	1	1	0	0	1	1	0	0	1	7th
0	1	1	0	1	0	1	0	0	1	
0	1	1	1	0	1	1	0	0	0	8th
0	1	1	1	1	0	1	0	0	0	
1	0	0	0	0	1	0	1	1	1	9th
1	0	0	0	1	0	0	1	1	1	
1	0	0	1	0	1	0	1	1	0	10th
1	0	0	1	1	0	0	1	1	0	
1	0	1	0	0	1	0	1	0	1	11th
1	0	1	0	1	0	0	1	0	1	
1	0	1	1	0	1	0	1	0	0	12th
1	0	1	1	1	0	0	1	0	0	
1	1	0	0	0	1	0	0	1	1	13th
1	1	0	0	1	0	0	0	1	1	
1	1	0	1	0	1	0	0	1	0	14th
1	1	0	1	1	0	0	0	1	0	
1	1	1	0	0	1	0	0	0	1	Lowest priority
1	1	1	0	1	0	0	0	0	1	
1	1	1	1	0	x	x	x	x	x	No interrupt pending
1	1	1	1	1	x	x	x	x	x	

Note: x = no value placed on address bus.

Table 2. Interrupt Acknowledge Summary

Priority of Interrupt	Interrupt Acknowledge	$\overline{\text{AVEC}}$	$\overline{\text{NMINT}}$	QIE	Result
Less than PSW IPL field priority	No	x	1	x	Interrupt is not acknowledged.
Equal to PSW IPL field priority	No	x	1	x	Interrupt is not acknowledged.
Greater than PSW IPL field priority	Yes	1	1	0	Interrupt is acknowledged via full-interrupt sequence. Microprocessor fetches vector number from interrupting device.
Greater than PSW IPL field priority	Yes	0	1	0	Interrupt is acknowledged via full-interrupt sequence. Microprocessor supplies the vector number.
Any level compared to PSW IPL field priority	Yes	x	0	0	Interrupt is acknowledged via full-interrupt sequence. It is treated as an autovector at vector number 0. The address bus contains all zeros during the acknowledge.
Greater than PSW IPL field priority	Yes	1	1	1	Interrupt is acknowledged and serviced via quick-interrupt sequence. Microprocessor fetches vector number from interrupting device.
Greater than PSW IPL field priority	Yes	0	1	1	Interrupt is acknowledged and serviced via quick-interrupt sequence. Microprocessor supplies the vector number.
Any level compared to PSW IPL field priority	Yes	x	0	1	Interrupt is acknowledged and serviced via quick-interrupt sequence. It is treated as an autovector interrupt at vector number 0. The address bus contains all zeros during the acknowledge.

Note: x = don't care.

Bus Arbitration

The microprocessor arbitrates requests for its bus (relinquish and retry request, $\overline{\text{RRREQ}}$, and bus request, $\overline{\text{BUSRQ}}$) in the following manner. The relinquish and retry request is acknowledged only during a bus transaction; however, it is ignored during the write portion of a read-interlocked transaction. A relinquish and retry request has priority over a bus request. A bus request is not acknowledged until the end of a bus transaction or until the end of the write portion of a read-interlocked transaction. A bus request is acknowledged immediately if the bus is idle.

Arbitrary Byte Alignment

The arbitrary byte alignment feature enables the microprocessor to efficiently handle nonaligned memory accesses for both reads and writes. The microprocessor automatically generates multiple accesses when nonaligned data types cross word boundaries. To illustrate how this is accomplished, the following notation is used:

Let n = most significant 30 bits of the address,
x = second least significant bit of the address,
and y = least significant bit of the address.

Now n01 stands for a 32-bit address with the two least significant bits being 01 (x = 0, y = 1), and (n+1)xy stands for address nxy + 100 (binary arithmetic).

For example (see Table 3), reading a word (four 8-bit bytes) from memory location n01 requires two read transaction cycles: the first a 3-byte read from location n01 and the second a byte read from memory location (n+1)00. Since one byte of data cannot cross a word boundary, a byte read transaction can always be completed in one read cycle with byte data size.

Similar cases exist for data writes (see Table 4), except that in virtual memory mode, probes occur on nonaligned data writes to guarantee restartability during memory faults. In physical memory mode, no probes exist and data writes are the same as read transactions.

Dynamic Bus Sizing

The dynamic bus sizing feature, available in the *WE* 32200 CPU, allows communication with both 16-bit and 32-bit memories and peripherals. For example, if the CPU is trying to read a word of data from memory and encounters a 16-bit port acknowledge (i.e., the CPU received only half of the word), it automatically generates a second access to read the other half of the word. Dynamic bus sizing also works with data writes and instruction fetches. The 16-bit port is defined as the upper half of the data bus. The CPU always initially assumes it is communicating with a 32-bit port. When a 16-bit port is accessed, the CPU generates additional memory accesses if the data size is word, three bytes, or halfword located in the middle of a word boundary. A useful application of dynamic bus sizing is to reduce board space in boot ROMs: a 32-bit boot ROM can be replaced by a 16-bit boot ROM. Tables 5 and 6 show the read and write accesses for 16-bit aligned and nonaligned data. Table 7 shows the 16-bit instruction fetches.

Table 3. Read Accesses for Aligned and Nonaligned Data (32-Bit)

Operand Address	Operand Size	Transaction Number	Transaction Address	Transaction Data Size
n00	word	1	n00	word
n01	word	1	n01	3 bytes
		2	(n+1)00	byte
n10	word	1	n10	halfword
		2	(n+1)00	halfword
n11	word	1	n11	byte
		2	(n+1)00	3 bytes
n00	halfword	1	n00	halfword
n01	halfword	1	n01	halfword
n10	halfword	1	n10	halfword
n11	halfword	1	n11	byte
		2	(n+1)00	byte
n00	byte	1	n00	byte
n01	byte	1	n01	byte
n10	byte	1	n10	byte
n11	byte	1	n11	byte

Table 4. Write Accesses for Aligned and Nonaligned Data (32-Bit)

Operand Address	Operand Size	Transaction Number	Transaction Address	Transaction Data Size
n00	word	1	n00	word
n01	word	1	n01	0 byte (probe)
		2	(n+1)00	byte
		3	n01	3 bytes
n10	word	1	n10	0 byte (probe)
		2	(n+1)00	halfword
		3	n10	halfword
n11	word	1	n11	0 byte (probe)
		2	(n+1)00	3 bytes
		3	n11	byte
n00	halfword	1	n00	halfword
n01	halfword	1	n01	halfword
n10	halfword	1	n10	halfword
n11	halfword	1	n11	0 byte (probe)
		2	(n+1)00	byte
		3	n11	byte
n00	byte	1	n00	byte
n01	byte	1	n01	byte
n10	byte	1	n10	byte
n11	byte	1	n11	byte

Table 5. Read Accesses for Aligned and Nonaligned Data (16-Bit)

Operand Address	Operand Size	Transaction Number	Transaction Address	Transaction Data Size
n00	word	1	n00	word
		2	n10	halfword
n01	word	1	n01	3 bytes
		2	n10	halfword
		3	(n+1)00	byte
n10	word	1	n10	halfword
		2	(n+1)00	halfword
n11	word	1	n11	byte
		2	(n+1)00	3 bytes
		3	(n+1)10	byte
n00	halfword	1	n00	halfword
n01	halfword	1	n01	halfword
		2	n10	byte
n10	halfword	1	n10	halfword

Table 5. Read Accesses for Aligned and Nonaligned Data (16-Bit) (Continued)

Operand Address	Operand Size	Transaction Number	Transaction Address	Transaction Data Size
n11	halfword	1	n11	byte
		2	(n+1)00	byte
n00	byte	1	n00	byte
n01	byte	1	n01	byte
n10	byte	1	n10	byte
n11	byte	1	n11	byte

Table 6. Write Accesses for Aligned and Nonaligned Data (16-Bit)

Operand Address	Operand Size	Transaction Number	Transaction Address	Transaction Data Size
		2	n10	halfword
n01	word	1	n01	0 byte (probe)
		2	(n+1)00	byte
		3	n01	3 bytes
		4	n10	halfword
n10	word	1	n10	0 byte (probe)
		2	(n+1)00	halfword
		3	n10	halfword
n11	word	1	n11	0 byte (probe)
		2	(n+1)00	3 bytes
		3	(n+1)10	byte
		4	n11	byte
n00	halfword	1	n00	halfword
n01	halfword	1	n01	halfword
		2	n10	byte
n10	halfword	1	n10	halfword
n11	halfword	1	n11	0 byte (probe)
		2	(n+1)00	byte
		3	n11	byte
n00	byte	1	n00	byte
n01	byte	1	n01	byte
n10	byte	1	n10	byte
n11	byte	1	n11	byte

Table 7. 16-Bit Instruction Fetches

Instruction Address	Instruction Size	Transaction Number	Transaction Address	Transaction Data Size	Bytes Transferred
n00	double word	1	n00	double word	2
		2	n10	halfword	2
n01	double word	1	n01	double word	2
		2	n11	halfword	2
n10	double word	1	n10	double word	2
		2	n00	halfword	2
n11	double word	1	n11	double word	2
		2	n01	halfword	2
n00	word	1	n00	word	2
		2	n10	halfword	2
n01	word	1	n01	word	2
		2	n11	halfword	2
n10	word	1	n10	word	2
		2	n00	halfword	2
n11	word	1	n11	word	2
		2	n01	halfword	2

Byte Replication

For ease of memory interface on halfword and byte data writes, the *WE* 32200 CPU performs byte replication. To illustrate byte replication, the following notation is used in Table 8:

Let Ax = active bytes, where x differentiates between active bytes,
Rx = replicated bytes, where x indicates the active byte, Ax, being replicated,
XXX = byte which may be driven on writes but value is undefined,
and XX = (in operand address column) don't care.

The notation in the operand address column is the same as that defined under Arbitrary Byte Alignment.

Table 8 shows all of the operand/transaction address/data size combinations possible. For example, if the CPU is writing a halfword, the data appears on the upper and lower half of the data bus.

Byte replication also allows the CPU to perform dynamic bus sizing. Using the same example, if the CPU were accessing a 16-bit port, the data would still be written to memory since the halfword was on both halves of the data bus. If byte replication were not performed, the CPU may have needed to generate another write access.

Direct Memory Access and Multiprocessor Support

The microprocessor provides the support for direct memory access (DMA) and shares bus control responsibilities with the system DMA controller. To initiate a DMA operation the controller, must request the microprocessor bus by asserting the bus request input ($\overline{\text{BUSRQ}}$). Once the microprocessor recognizes the request, it drives its outputs to the states given in Table 9. The microprocessor then acknowledges the DMA request by asserting the bus request acknowledge output ($\overline{\text{BRACK}}$). Upon completion of the DMA operation, the 3-stated signals are returned to the microprocessor's control.

The microprocessor provides for the support of multiprocessors via the $\overline{\text{BARB}}$ pin. When this signal is asserted (strapped to 0), the microprocessor assumes the role of bus arbiter and other devices request the bus from the microprocessor. When this signal is not asserted (strapped to 1), the microprocessor must request the bus from an external bus arbiter thus providing a multiprocessor system capability.

Table 8. Replication Patterns for Write Accesses

Operand Address	Data Size	Replication Pattern			
		D31—D24	D23—D16	D15—D8	D7—D0
XX	0 byte	XXX	XXX	XXX	XXX
n00	byte	A1	R1	R1	R1
n01	byte	R1	A1	R1	R1
n10	byte	R1	R1	A1	R1
n11	byte	R1	R1	R1	A1
n00	halfword	A1	A2	R1	R2
n01	halfword	XXX	A1	A2	XXX
n10	halfword	R1	R2	A1	A2
n00	3 bytes	A1	A2	A3	XXX
n01	3 bytes	XXX	A1	A2	A3
n00	word	A1	A2	A3	A4

Table 9. Output States After DMA Request Acknowledge

Output Signal	Signal State
$\overline{\text{ABORT}}$	Z'
ADDR00—ADDR31	Z
$\overline{\text{AS}}$	Z'
$\overline{\text{BRACK}}$	Logic 0
$\overline{\text{CYCLEI}}$	Z'
DATA00—DATA31	Z
$\overline{\text{DRDY}}$	Z'
$\overline{\text{DS}}$	Z'
DSIZE0—DSIZE2	Z
R/$\overline{\text{W}}$	Z'
$\overline{\text{RESET}}$	Logic 1
$\overline{\text{RRRACK}}$	Logic 1
SAS0—SAS3	Z'
$\overline{\text{VAD}}$	Z
XMD0, XMD1	Z

Notes:
Z = High-impedance.
Z' = High-impedance. Normally tied high with external passive holding resistor.

Coprocessor Interface

The coprocessor interface consists of ten instructions and the associated pinout and protocol. This interface is provided to insure high performance of coprocessors, thus increasing overall system throughput.

Semaphores/Read-Write Interlocked Support

The microprocessor provides two types of interlocked instructions. The unconditional read/write instruction swaps two operands, while the conditional read/write instruction (i.e., semaphore instruction) performs the swap only if the two operands are equal. In general, an interlock operation prevents another process from accessing a memory location between the read and write halves of the operation. Interlocked operations consist of a memory fetch (read access), one or more internal microprocessor operations, and then a write access (if any) to the same memory location. Once the read access has been completed, the interlocked operation may not be preempted (other than by a reset) until the write is completed or until the microprocessor decides that no write operation is necessary.

Reset

Two types of reset requests are available – system- and microprocessor-initiated. Both requests have highest priority and preempt any ongoing microprocessor operation. A system-initiated reset only differs from a microprocessor-initiated reset because it is externally generated. The system initiates a reset by driving the reset request input ($\overline{\text{RESETR}}$) low. Once the microprocessor recognizes the request, it sends a reset acknowledge ($\overline{\text{RESET}}$) to the system and also drives its outputs to a temporary state that prevents control signal and bus conflicts while the system responds to the acknowledge. The system negates $\overline{\text{RESETR}}$ once it has responded to the acknowledge. From this point the microprocessor continues to hold $\overline{\text{RESET}}$ active for an additional 128 clock cycles allowing the external system to go through its own initialization sequence. At the end of this period it will negate $\overline{\text{RESET}}$ and begin execution of the internal reset sequence. Table 10 indicates the states of the microprocessor's output pins once it has negated $\overline{\text{RESET}}$. This sequence performs the register initialization required to begin restart of operations as follows:

- The microprocessor initiates physical addressing mode.
- The microprocessor fetches a word at location 80 (hexadecimal) and puts it into the process control block pointer (r13). This word is the beginning address of the reset process control block (PCB).
- The microprocessor fetches a word at the PCB address and stores it in the PSW.
- It fetches a word at the location four bytes from the initial PCB address and stores it in the program counter (PC). This word is the PC value for initial execution.
- The microprocessor fetches a word at the location eight bytes from the initial PCB address and stores it in the stack pointer (SP).
- It begins execution at the address specified by the PC.

Bus Exceptions

Bus exceptions cause the termination of the current memory access and result when retry is required or when an exception occurs during an access. The bus exceptions are of three types: relinquish and retry, retry, and exception. A relinquish and retry causes the microprocessor to give up its bus and to subsequently retry the preempted access once the bus has been returned to its control. An external device may request a relinquish and retry by driving the $\overline{\text{RRREQ}}$ input to its active state. A retry causes the microprocessor to retry the access. An external request device may request a retry by asserting the $\overline{\text{RETRY}}$ input. An exception is the result of an error condition during a bus cycle, which an external device can report to the microprocessor by asserting the $\overline{\text{FAULT}}$ input. This causes the microprocessor to terminate the access and, perhaps, to execute an exception handling routine, depending on the type of access.

Table 11 describes how the microprocessor handles the simultaneous assertion of two or more bus exceptions.

Programming

Processor Status Word (PSW)

The PSW (r11) contains several state variables associated with the current process and provides a means of determining the microprocessor's execution state at any time. The word format for the PSW is shown on Figure 3.

Bit	31 30	29	28	27	26	25	24	23	22	21 18	17	16 13	12 11	10 9	8	7	6 3	2	1 0
Field	Unused	EA	EX/UC	AR	X	CFD	QIE	CD	OE	NZVC	TE	IPL	CM	PM	R	I	ISC	TM	ET

Figure 3. Word Format for the PSW

Table 10. Output States on Reset

Output Signal	Signal State[1]	
	CPU[2] is Bus Arbiter[3]	CPU is not Bus Arbiter
$\overline{\text{ABORT}}$	Logic 1	Z
ADDR00—ADDR31	Z	Z
$\overline{\text{AS}}$	Logic 1	Z
$\overline{\text{BRACK}}$	Logic 1	—
$\overline{\text{BUSRQ}}$	—	Logic 1
$\overline{\text{CYCLEI}}$	Logic 1	Z
DATA00—DATA31	Z	Z
$\overline{\text{DRDY}}$	Logic 1	Z
$\overline{\text{DS}}$	Logic 1	Z
DSIZE0—DSIZE2	Logic 0	Z
IQS0, IQS1	Logic 1	Logic 1
R/$\overline{\text{W}}$	Logic 1	Z
$\overline{\text{RRRACK}}$	Z[4]	Z[4]
SAS0—SAS3	Logic 1	Z
$\overline{\text{SOI}}$	Logic 1	Logic 1
$\overline{\text{VAD}}$	Note 5	Z
XMD0, XMD1	Note 6	Z

[1] Z = High-impedance.
[2] CPU is *WE* 32200 Microprocessor.
[3] Determined by state of $\overline{\text{BARB}}$ input pin.
[4] Open drain output not actively driven under this condition.
[5] Not guaranteed to be logic 1 (i.e., physical mode) until approximately 66 clock cycles after $\overline{\text{RESET}}$ is negated.
[6] Not guaranteed to be in kernel mode until approximately 18 clock cycles after $\overline{\text{RESET}}$ is negated.

The following defines the PSW bit field.

Bits 0, 1 – Exception Type (ET). Bits 0 and 1 represent exceptions generated during operations. This read-only field is interpreted as:

Bit 1	Bit 0	Description
0	0	On reset exception
0	1	On process exception
1	0	On stack exception
1	1	On normal exception

Bit 2 – Trace Mask (TM). Bit 2 enables masking of a trace trap. This read-only field masks the trace enable (TE) bit for the duration of one instruction in order to avoid a trace trap. The TM bit is set at the beginning of every instruction and cleared as part of every microsequence (see Microsequences) that performs a context switch or a return from gate (RETG), or when any fault or interrupt is detected and responded to.

Bits 3 through 6 – Internal State Code (ISC). Bits 3 through 6 represent the microprocessor internal state code and distinguish between exceptions of the same exception type. This field is used in conjunction with the exception type (ET) field to determine which type of exception occurred. Traps, exceptions, and faults are equivalent with respect to the ISC. Normal exceptional conditions are decoded on a priority scheme if more than one occurs in a particular cycle. Table 12 lists the priority and ISC values.

Bits 7, 8 – Register-Initial Context (RI). Bits 7 and 8 control the microprocessor context switching strategy. The I bit (bit 7) determines if a process executes from initial (I=1) or intermediate saved context. The R bit (bit 8, read only) determines if the registers of a process should be saved during a process switch (R=1). It also controls block moves, which can be used to change MMU information. On external reset, the R bit is cleared.

Bits 9, 10 – Previous Execution Level (PM). Bits 9 and 10 represent the previous execution level and are interpreted as:

Bit 10	Bit 9	Description
0	0	Kernel level
0	1	Executive level
1	0	Supervisor level
1	1	User level

Bits 11, 12 – Current Execution Level (CE). Bits 11 and 12 represent the current execution level. The code for bits 11 and 12 is interpreted in the same manner as that of bits 9 and 10 (respectively) of the PM code.

Changes to the CM field via instructions that have the PSW as an explicit destination may affect the XMD pins during a prefetch access. Accordingly, only microsequence instructions should be used to change the CM field state.

Table 11. Simultaneously Asserted Exception Conditions

Simultaneously Asserted Signals	Behavior
Condition 1: $\overline{\text{RRREQ}}$, $\overline{\text{RETRY}}$, $\overline{\text{FAULT}}$	In this instance the relinquish and retry request ($\overline{\text{RRREQ}}$) is honored first. The microprocessor acknowledges this request by relinquishing the bus and then asserting the relinquish and retry request acknowledge output ($\overline{\text{RRRACK}}$). The access is retried once $\overline{\text{RRREQ}}$ and $\overline{\text{RETRY}}$ are negated by the requesting devices. If the fault input ($\overline{\text{FAULT}}$) is still asserted during the retried access, then the exception will be honored (recognized). The fault input will only be recognized during the retried access.
Condition 2: $\overline{\text{RRREQ}}$, $\overline{\text{RETRY}}$	In this instance the relinquish and retry request ($\overline{\text{RRREQ}}$) is honored first. The microprocessor 3-states the appropriate signals and then asserts the relinquish and retry acknowledge output ($\overline{\text{RRRACK}}$). The access is retried once $\overline{\text{RRREQ}}$ and $\overline{\text{RETRY}}$ are negated.
Condition 3: $\overline{\text{RRREQ}}$, $\overline{\text{FAULT}}$	Same as Condition 1.
Condition 4: $\overline{\text{RETRY}}$, $\overline{\text{FAULT}}$	In this instance the $\overline{\text{RETRY}}$ request is honored first. The $\overline{\text{FAULT}}$ will be recognized on the retried access if it is still asserted.

Note: This table applies only when the microprocessor is the bus master ($\overline{\text{BARB}}$ asserted).

Bits 13 through 16 – Interrupt Priority Level (IPL). Bits 13 through 16 represent the current interrupt priority level. Fifteen levels of interrupts are available. An interrupt, unless it is a nonmaskable interrupt, must have a higher priority level than the current IPL of the PSW in order to be acknowledged. Level 0000 indicates that any of the other fifteen interrupt priority levels (0001 through 1111) can interrupt the microprocessor, while level 1111 indicates that no interrupts except a nonmaskable interrupt can interrupt the microprocessor since 1111 is the highest interrupt priority level.

Bit 17 – Trace Enable (TE). Bit 17 enables the trace function. When set, it causes a trace trap to occur after execution of the next instruction. Debugging and analysis software use this facility for single-stepping a program. Changes to the TE field via instructions that have the PSW as an explicit destination may cause unpredictable trace-trap behavior, i.e., the instruction that changed the TE field in the PSW may or may not cause a trace trap. Accordingly, only microsequence instructions should be used to change the TE field state.

Bits 18 through 21 and bit 26 – Condition Codes (NZVCX). Bits 18 through 21 and bit 26 represent condition codes. These codes reflect the status of the last instruction execution to affect them. The codes are tested by using the conditional branch instructions and indicate the following when set:

N	— Negative	(bit 21)
Z	— Zero	(bit 20)
V	— Overflow	(bit 19)
C	— Carry	(bit 18)
X	— Extended Carry (for BCD only)	(bit 26)

Bit 22 – Enable Overflow Trap (OE). Bit 22, when set, enables overflow traps. It is cleared whenever an overflow trap is detected and handled.

Bit 23 – Cache Disable (CD). Bit 23 enables and disables the instruction cache. When the CD bit is cleared, the cache is used to store and read text. When the CD bit is set, the cache is not used. The instruction cache should be disabled only when its use could cause problems, e.g., when self-modifying code is executing. Changes to the CD field via

instructions that have the PSW as an explicit destination may corrupt the contents of the instruction cache. Therefore, only microsequence instructions should be used to change the CD field state.

Bit 24 – Quick Interrupt Enable (QIE). Bit 24 enables and disables the quick-interrupt facility. If QIE is set, an interrupt is handled via the quick-interrupt sequence. If QIE is cleared, the interrupt causes a process switch (full-interrupt sequence). Changes to the QIE field should be made only with microsequence instructions in order to prevent possible unpredictable QIE states.

Bit 25 – Cache Flush Disable (CFD). Bit 25, when set, disables instruction cache flushing (emptying of the cache's contents) when a new process is loaded via the XSWITCH_TWO microsequence (see Microsequences). When cleared, the contents of the cache are flushed when a new process is loaded via the XSWITCH_TWO microsequence. Changes to the CFD field via instructions that have the PSW as an explicit destination may cause unpredictable cache flushing. Therefore, only microsequence instructions should be used to change the CFD field state.

Bit 26 – Extend Carry/Borrow (X). Bit 26 represents the condition code for the BCD operations. The extended carry/borrow bit can be set only if there is a carry or borrow from a BCD arithmetic operation on two BCD digits. When there is no carry or borrow from such operations, the bit is reset.

It can also be set and cleared by two BCD flag setting instructions, SETX and CLRX, respectively, or by a write to the PSW in kernel mode.

Bit 27 – Additional Register Save (AR). Bit 27, when set, enables the additional 8 registers (r16—r23) to be saved during a process switch but only when no block moves are being executed. When reset, the CPU returns to *WE* 32100 CPU compatible mode and process switch reverts to *WE* 32100 CPU. On external reset, this bit is cleared.

Bit 28 – Normal Exception and User Call Process Option (EX/UC). Bit 28, when set, causes normal exception procedures to be identical to process switches, except that the PCBP is located at a predefined address.

Table 12. Exceptional Conditions

Exception Type	Priority	Exception	Internal State Code Bit 6 5 4 3
On normal exception (ET = 11)	1	Gate vector fault	0 1 1 0
	3	Illegal level change	0 1 1 1
	4	Breakpoint trap	1 1 1 0
	5	Privileged opcode	1 0 1 0
	6	Illegal opcode	0 0 1 0
	7	Reserved opcode	0 0 1 1
	8	Invalid descriptor	0 1 0 1*
	9	Reserved data type	1 0 0 0*
	10	Privileged register	1 1 1 1
	11	Integer zero-divide	0 0 0 0*
	13	External memory fault	0 1 0 1
	14	Trace trap	0 0 0 1
	15	Integer overflow trap	1 0 0 1
Stack exception (ET = 10)	0	Stack bound	0 0 0 0
	1	Stack fault	0 0 0 1
	3	Interrupt ID fetch	0 0 1 1
Process exception (ET = 01)	0	Old PCB fault	0 0 0 0
	1	Gate PCB fault	0 0 0 1
	4	New PCB fault	0 1 0 0
Reset exception (ET = 00)	0	Old PCB fault	0 0 0 0
	1	System data	0 0 0 1
	2	Interrupt stack fault	0 0 1 0
	3	External reset	0 0 1 1
	4	New PCB fault	0 1 0 0
	6	Gate vector fault	0 1 1 0

* Exceptional conditions that reset the PSW flags.

In addition, the UCALLPS instruction is enabled, which allows a user-initiated process switch. The RETPS instruction can be used to terminate both processes. When cleared, the option reverts to *WE* 32100 CPU mode, where the normal exception is gate-like. In addition, the UCALLPS instruction is disabled. If the UCALLPS instruction is then used, an illegal opcode exception is generated. On external reset, this bit is cleared.

Bit 29 – Arbitrary Byte Alignment Enable (EA). Bit 29 enables and disables the arbitrary byte alignment feature in the *WE* 32200 CPU. Arbitrary byte alignment is enabled when the EA bit is set, allowing the CPU to read or write word and half-word data from any byte boundary. If EA is cleared, arbitrary byte alignment is disabled and the alignment fault detection is enabled (i.e., compatible with the *WE* 32100 CPU). On external reset, the EA bit is cleared.

Bits 30 and 31 – Unused. Bits 30 and 31 are not used and must always be cleared.

Process Control Block (PCB)

The PCB contains the entire switchable process context, collected into a compact form for ease of movement between system memory and privileged internal registers. The process control block pointer (PCBP), r13, contains the address of the PCB currently in execution. The PCB is shown on Figures 4 and 5 and a summary of its contents appears in Table 13.

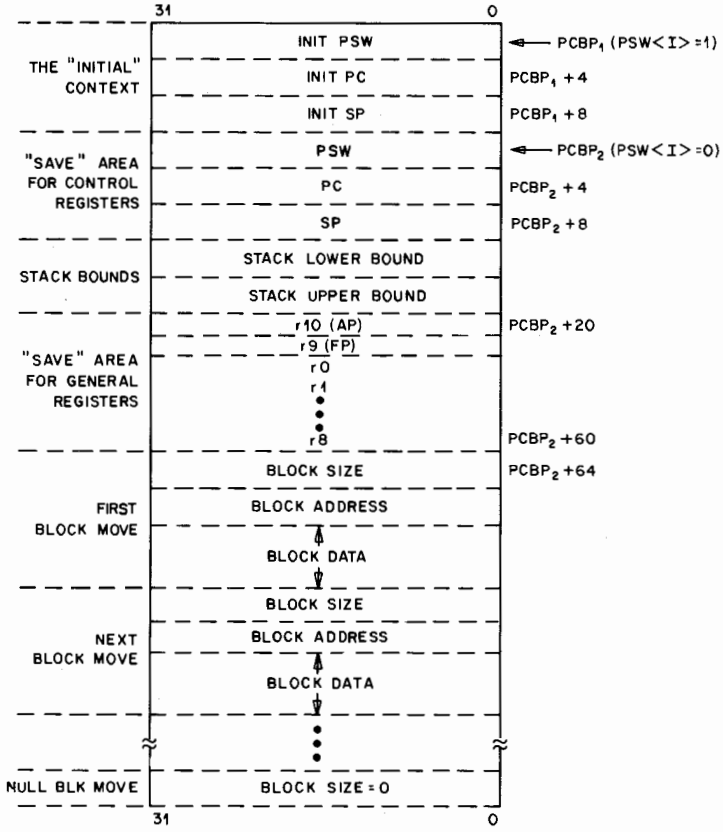


Figure 4. Process Control Block, PSW(R) = 1, PSW(AR) = 0

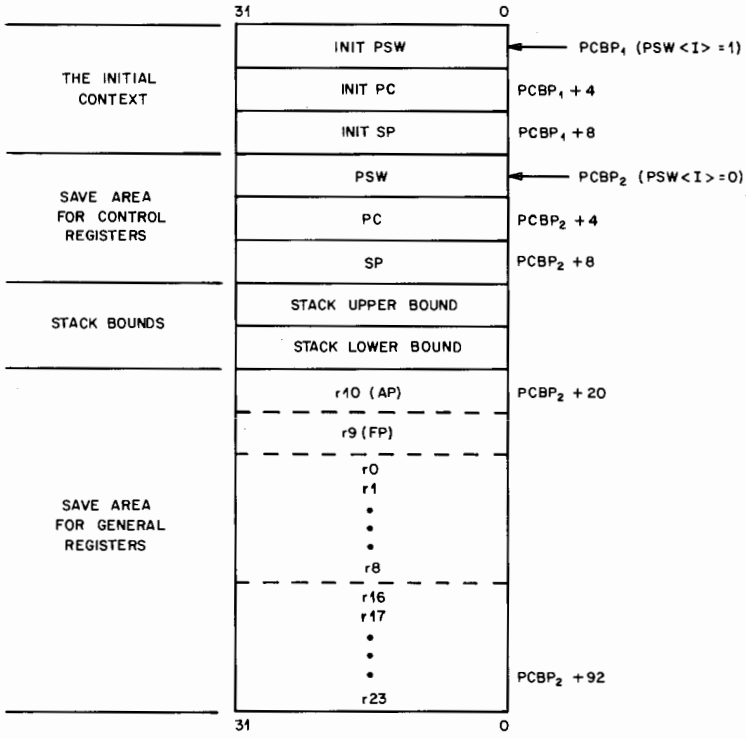


Figure 5. Process Control Block, PSW(R) = 1, PSW(AR) = 1

Table 13. Definitions of Process Control Block Sections

PCB Section	Description
Initial context	This section contains a set of initial values for the PSW, PC, and SP. This allows a process to be restarted with its initial values even though, at some point, its execution was suspended.
Save area for control registers	This section contains the PSW, PC, and SP values at the time the current process is switched.
Stack bounds	This section defines the upper and lower bounds of the stack used in the current process. It contains pointers to the upper and lower memory limits of the stack.
Save area for general registers	This section or area contains images of what the general registers were when the process was switched. The images are of registers 0 through 10 (r0—r10) if PSW(R) = 0; and of registers 0 through 10, with registers 16 through 23 (r0—r10, r16—r23), if PSW(R) = 1 and PSW(AR) = 1. This allows the process to continue in the same state that it was in when control switched to another process.
First block move	Moves data in PCB to an arbitrary memory region.
Next & nth block move	Moves a different set of data in PCB to a different memory region.
Null block move	Terminates block move sequence.

Flags/Conditions

The processor status word (r11) contains five flag bits: negative (N), zero (Z), overflow (V), carry/borrow (C), and extended carry (X) bits. For most instructions, these flags are set according to standard criteria. In the following flag descriptions, the term *logical instruction* refers to any nonarithmetic instruction. *Infinite precision result* refers to what the result would be if the operation were done in an "infinite precision" machine. The type of result is determined by the first operand in monadic operations and the second operand in dyadic and triadic operations. *Requested representation* refers to the actual data that is written into the desired location. The representation depends on the site requested (byte, halfword, or word) and whether the result is signed or unsigned. Loop control does not affect the flag bits. The flag bits are set as follows:

N – Negative. The negative flag is set if the sign of the infinite-precision result is negative for nonlogical instructions. Zero is considered positive in this case. For logical instructions, the negative flag is the most significant bit (bit 31 for words, bit 15 for halfwords, or bit 7 for bytes) of the requested representation. Note that this bit is valid even in unsigned operations.

Z – Zero. For nonlogical instructions, the zero flag is set if the infinite-precision result is equal to zero. For logical instructions, it is set if the requested representation is equal to zero.

V – Overflow. For arithmetic operations with signed destination, the V bit is set if any truncated bit of the infinite precision result is different from the sign bit, which is the most significant bit, of the requested representation. For arithmetic operations with an unsigned destination, the V bit is set if any truncated bit is nonzero. For arithmetic left-shift operations, bits are not shifted past bit 31 of the result, so the V bit will be set only if there is a truncation error. For logical instructions with a signed destination, the V bit is set if any truncated bit of the 32-bit result is different from the sign bit of the requested representation. For logical instructions with an unsigned destination, the V bit is set if any truncated bit is nonzero.

C – Carry/Borrow. The carry bit is the carry into the 33rd bit position (bit 32) for word operations, the 17th bit position (bit 16) for halfword operations, or the 9th bit position (bit 8) for byte operations. For subtracts, negates, decrements, and compares, it is the borrow from the 33rd, 17th, or 9th bit position. For example, consider A–B, where A and B are unsigned. If A $\geq$ B after both A and B are extended to 32 bits, then C is cleared. Otherwise the C flat is set. For logical instructions, the carry is cleared. For BCD arithmetic operations, the carry is set if there is a carry or a borrow from a BCD arithmetic operation on two 2-BCD digits.

X – Extended Carry/Borrow. The extended carry/borrow bit can be set and reset only by BCD instructions. The bit is set if there is a carry or borrow from a BCD arithmetic operation on two BCD digits. The bit is reset if there is no carry or borrow from such BCD operations. It can also be set and cleared by two BCD flag setting instructions, SETX and CLRX, respectively, or by a write to the PSW in the kernel mode.

Addressing Modes

The addressing mode of an operand in an instruction refers to the way in which the location of the operand is determined. A memory-based operand requires an effective address to be determined according to the operand descriptor. The operand descriptor for register operands specifies the register. Unless specified by the instruction, all operands are addressed by a descriptor.

The *WE* 32200 Microprocessor has extensively enhanced its addressing modes beyond that of the *WE* 32100 Microprocessor and thus provides two encoding formats for the first byte of the descriptor. If the byte is not equal to 0x5B (5 PSW), 0xAB (10 PSW), 0xBB (11 PSW), 0xCB (12 PSW), or 0xDB (13 PSW) which represents format 2, then the format (format 1) is the same as the *WE* 32100 Microprocessor addressing modes.

Format 1 Addressing Modes

In format 1, the format of the first byte of a descriptor is

mmmmrrrr

where rrrr (bit 0—3) represent a general-purpose register (r0—r15) and mmmm (bits 4—7) represent the addressing mode. The addressing modes are summarized in Tables

14 (format 1) and 15 (format 2). For format 1, the modes are interpreted as:

Mode Field	Description
0—3	**Positive Literal Mode.** The four bits rrrr (r0—r15) concatenated with the low-order two bits of mmmm constitute a positive 6-bit literal. If a literal is specified as a destination or its address is requested by the opcode, an illegal operand exception is generated.
4	**Register Mode.** Bits rrrr specify one of 15 registers (r0—r14). An illegal operand exception is generated if the instruction attempts to take this operand's address. If rrrr specifies PC, mode becomes "word immediate." Four bytes following mmmmrrrr byte provide a 32-bit immediate value. The first byte following the operand descriptor byte contains the low order-bits (0—7). If a word immediate is specified as a destination, an illegal operand exception is generated if the address of the word immediate operand is requested by the opcode.
5	**Register Deferred Mode.** The specified register (any one of r0—r14 except r11) contains operand address; r11 cannot be specified. The mmmm = 5, rrrr = 11 combination is used for format 2 addressing modes. If rrrr specifies PC, the mode becomes "halfword immediate." The two bytes following the mmmmrrrr byte provide a 16-bit immediate value (range –32768 to 32767). The first byte contains the low-order bits of the immediate value; the second contains the high-order bits. If a halfword immediate is specified as a destination, an illegal operand exception is generated if the address of the halfword immediate operand is requested by the opcode.
6	**FP Short Offset Mode.** This mode implicitly refers to the FP (r9). Bits rrrr are used as a literal (range 0 to 14) and added to the FP to produce the operand address. This mode, an optimization of the byte displacement mode for short FP offsets, is useful for referring to local variables of high-level language functions. If a "byte immediate" is specified as a destination, an illegal operand exception is generated if the address of the byte immediate operand is requested by the opcode. If rrrr specifies PC, the mode becomes byte immediate. The byte following the mmmmrrrr byte provides 8-bit immediate value (range –128 to 127).
7	**AP Short Offset Mode.** This mode implicitly refers to the AP (r10). Bits rrrr are used as a literal (range 0 to 14) and added to the AP to produce the operand address. This mode, an optimization of the byte displacement mode for short AP offsets, is useful for referring to high-level language function arguments. If rrrr specifies PC, the mode becomes "absolute." The four bytes following the mmmmrrrr byte provide the operand address for instruction. The first byte contains bits 0 through 7 (low-order); second, bits 8 through 15; third, bits 16 through 23; fourth, bits 24 through 31.
8	**Word Displacement Mode.** The four bytes following the operand descriptor byte are fetched from the instruction stream and added to the register specified by rrrr. Bits rrrr can range from 0 to 10 and 12 to 15. The use of r11 (PSW) causes an invalid descriptor exception. The sum is the address of the instruction operand. The first byte contains the low-order bits of the displacement value.
9	**Word Displacement Deferred Mode.** The four bytes following the operand descriptor byte are fetched from the instruction stream and added to the register specified by rrrr. Bits rrrr can range from 0 to 10 and 12 to 15. The use of r11 (PSW) causes an invalid descriptor exception. The first byte contains the low-order bits of the displacement value. The sum is the address of a 32-bit pointer, which is fetched and used as the instruction's operand address.
10	**Halfword Displacement Mode.** Two bytes following the operand descriptor byte are fetched from the instruction

stream, sign-extended to 32 bits, and added to the rrrr register. Bits rrrr can range from 0 to 10 and 12 to 15; r11 cannot be specified. The first byte contains the low-order bits of the displacement value. The sum is the instruction's operand address. The mmmm = 10, rrrr = 11 combination is used for format 2 addressing modes.

11 **Halfword Displacement Deferred Mode.** The two bytes following the operand descriptor byte are fetched from the instruction stream, sign-extended to 32 bits, and added to the rrrr register. Bits rrrr can range from 0 to 10 and 12 to 15; r11 cannot be specified. The first byte contains the low-order bits of the displacement value. The sum is the address of a 32-bit pointer, which is fetched and used as the instruction's operand address. The mmmm = 11, rrrr = 11 combination is reserved for format 2 addressing modes.

12 **Byte Displacement Mode.** The byte following the operand descriptor byte is fetched from the instruction stream, sign-extended to 32 bits, and added to the register specified by rrrr. Bits rrrr can range from 0 to 10 and 12 to 15; r11 cannot be specified. The sum is the instruction's operand address. The mmmm = 12, rrrr = 11 combination is reserved for format 2 addressing modes.

13 **Byte Displacement Deferred Mode.** The byte following the operand descriptor byte is fetched from the instruction stream, sign-extended to 32 bits, and added to the register specified by rrrr. Bits rrrr can range from 0 to 10 and 12 to 15; r11 cannot be specified. The sum is the address of a 32-bit pointer, which is fetched and used as the instruction's operand address. The mmmm = 13, rrrr = 11 combination is reserved for format 2 addressing modes.

14 **Expanded-Operand Type Mode.** Operand type and its address mode are determined by a combination of the mmmmrrrr byte (r0—r14) and the next byte in the instruction stream. This second byte can be decoded as format 1 or format 2, accordingly. This next byte is formatted as

nnnngggg

where nnnn (bits 4 to 7) defines actual address mode and gggg (bits 0 to 3) defines the actual register of the operand. For format 1, a value of 14 for nnnn generates an invalid descriptor exception, i.e., only one level of expansion is possible. The mode determination register is provided by the gggg field of the second byte. Operand type is determined by the rrrr of the first byte. Field type overrides instruction opcode type and may only be overridden by another expanded-type descriptor. It does not override width description of immediate operands. Byte immediates will still be 8 bits, halfword 16, and word 32. Valid types for rrrr are:

7	Signed byte
3	Unsigned byte
6	Signed halfword
2	Unsigned halfword
4	Signed word
0	Unsigned word

Types 1, 5, and 8 through 14 are not defined on the *WE* 32200 Microprocessor and generate a reserved data-type exception.

If r15 (PC) is specified by rrrr, this becomes "word absolute deferred" mode instead of expand mode. The four bytes following the mmmmrrrr byte provide the 32-bit absolute address of the instruction operand. The first byte contains bits 0 through 7; the second, bits 8 through 15; the third, bits 16 through 23; and the fourth, bits 24 through 31.

15 **Negative Literal Mode.** Bits rrrr (r0—r15) concatenated with mmmm constitute a negative 8-bit literal (range −1 to −16). If a literal is specified as a destination, or if the opcode requests the literal operand's address, an illegal operand exception is generated.

Table 14. Addressing Modes – Format 1

Mode	Syntax	Mode Field	Register Field	Total Bytes	Notes
Absolute					
Absolute	*$expr*	7	15	5	—
Absolute deferred	**$expr*	14	15	5	—
Displacement (From a Register)					
Byte displacement	*expr*(%r*n*)	12	0—10, 12—15	2	5
Byte displacement deferred	**expr*(%r*n*)	13	0—10, 12—15	2	5
Halfword displacement	*expr*(%r*n*)	10	0—10, 12—15	3	5
Halfword displacement deferred	**expr*(%r*n*)	11	0—10, 12—15	3	5
Word displacement	*expr*(%r*n*)	8	0—10, 12—15	5	6
Word displacement deferred	**expr*(%r*n*)	9	0—10, 12—15	5	6
AP short offset	*so*(%ap)	7	0—14	1	1
FP short offset	*so*(%fp)	6	0—14	1	1
Immediate					
Byte immediate	*&imm8*	6	15	2	2,3
Halfword immediate	*&imm16*	5	15	3	2,3
Word immediate	*&imm32*	4	15	5	2,3
Positive literal	*&lit*	0—3	0—15	1	2,3
Negative literal	*&lit*	15	0—15	1	2,3
Register					
Register	%r*n*	4	0—14	1	1,3
Register deferred	(%r*n*)	5	0—10, 12—14	1	1,5
Special Mode					
Expanded-operand type	*{type}opnd*	14	0—14	2—6	4

[1]Mode field has special meaning if register field is 15; see absolute or immediate mode.
[2]Mode may not be used for a destination operand.
[3]Mode may not be used if the instruction takes effective address of the operand.
[4]*type* overrides instruction type; *opnd* determines actual address mode. For total bytes, add 1 to byte count for address mode determined by *opnd*.
[5]In this mode, r11 refers to format 2 addressing mode.
[6]In this mode, r11 causes an invalid descriptor exception.

Format 2 Addressing Modes

The addressing modes for format 2 are summarized in Table 15.

Addressing modes are specified as format 2 if the first byte of the operand descriptor is 0x5B (5 PSW), 0xAB (10 PSW), 0xBB (11 PSW), 0xCB (12 PSW), or 0xDB (13 PSW). These addressing modes are unique to the *WE* 32200 Microprocessor.

Note: The PSW, register 11, should not be used for FP or AP short offset. Use of it will cause an invalid descriptor exception.

First Byte	Description
0x5B	**Auto Pre-Decrement.** The register specified is first decremented by the size of the operand: 1 for byte, 2 for halfword, or 4 for word. The decremented register contains the address of the operand. **Auto Post-Decrement.** The register specified is used as a pointer to the address of the operand. The register is then decremented by the size of the operand: 1 for byte, 2 for halfword, or 4 for word. **Auto Pre-Increment.** The specified register is first incremented by the size of the operand: 1 for byte, 2 for halfword, or 4 for word. The incremented register contains the address of the operand. **Auto Post-Increment.** The specified register is used as a pointer to the address of the operand. The register is then incremented by the size of the operand: 1 for byte, 2 for halfword, or 4 for word.
0xAB	**Indexed Register with Byte Displacement.** The CPU in this mode fetches the two bytes following the operand descriptor byte from the instruction stream to identify the operand. The least significant nibble of the first byte specifies register %r16 through %r31. The most significant nibble of the same byte specifies register %r0 through %r15. Assuming the least significant nibble to be y, the register specified is $y+16$. The second byte is sign-extended to 32 bits and added to the sum of the contents of the two registers. The total is the address of the operand to be used in the instruction. The use of the PSW or PC causes an invalid descriptor exception.
0xBB	**Indexed Register with Halfword Displacement.** In this mode, the CPU fetches the three bytes following the operand mode descriptor byte from the instruction stream to identify the operand. The most significant nibble of the first byte specifies register %r0 through %r15. The least significant nibble is y; the register specified is $y+16$. The remaining halfword (second and third bytes) is sign-extended to 32 bits and added to the sum of the contents of the two registers. The total is the address of the operand to be used in the instruction. The use of the PSW or PC causes an invalid descriptor exception.
0xCB	**Format 1 Addressing Modes for Registers 16—31.** The second byte following the operand mode descriptor is fetched from the instruction stream and is interpreted as the first byte of a format 1 operand descriptor. In other words, the byte is decoded the same as format 1. The byte cannot contain the first byte of a format 2 operand. Bits 0 through 3 of the byte represent a register, 16 to 31. Assuming bits 0 through 31 to be y, the register number is $y+16$. Bits 4 through 7 are to be decoded as shown in Table 14. **Note:** The expanded, literal, and short-offset modes cause an invalid descriptor exception. Operand descriptors for format 2 expand modes should start with expanded mode (format 1) and then be followed by any format 2 addressing modes.
0xDB	**Indexed Register with Scaling.** The CPU in this mode fetches the byte following the operand mode descriptor byte from the instruction stream to identify the operand. The most significant nibble of the byte specifies the index register (%rn, where n = 0—15). The least significant nibble of the byte specifies the base register %r16 through %r31. Assuming the least significant nibble to be y, the register specified is $y+16$. Depending on the operand size (byte, halfword, or word), the contents of the register (%r0—%r15) specified by the most significant nibble are first multiplied by 1, 2, or 4. This product is then added to the contents of the register (%r16—%r31) specified by the least significant nibble of the byte. Finally, the sum is the address of the operand to be used in the instructions. The use of the PSW or PC causes an invalid descriptor exception, as does the use of this mode with coprocessor instructions.

Table 15. Addressing Modes – Format 2

Operand Mode Descriptor		Name	Syntax	dest?	EA?	PC?
1st Byte	2nd Byte					
Auto Pre/Post Increment/Decrement Addressing Modes						
0x5B	000 rrrrr	Auto pre-decrement	— (%rx)	yes	yes	illegal
0x5B	010 rrrrr	Auto post-decrement	(%rx)—	yes	yes	illegal
0x5B	100 rrrrr	Auto pre-increment	+(%rx)	yes	yes	illegal
0x5B	110 rrrrr	Auto post-increment	(%rx)+	yes	yes	illegal
Indexed Register Addressing Modes						
0xAB	NA	Indexed with byte displacement	expr(%rn, %rm)	yes	yes	yes
0xBB	NA	Indexed with halfword displacement	expr(%rn, %rm)	yes	yes	yes
0xDB	NA	Indexed with scaling	%rm[%rn]	yes	yes	illegal
Format 1 Addressing Modes for r16—r31						
0xCB	0100 rrrr	Register mode	%rm	yes	no	NA
0xCB	0101 rrrr	Register deferred	(%rm)	yes	yes	NA
0xCB	1000 rrrr	Word displacement	expr(%rm)	yes	yes	NA
0xCB	1001 rrrr	Word displacement deferred	*expr(%rm)	yes	yes	NA
0xCB	1010 rrrr	Halfword displacement	expr(%rm)	yes	yes	NA
0xCB	1011 rrrr	Halfword displacement deferred	*expr(%rm)	yes	yes	NA
0xCB	1100 rrrr	Byte displacement	expr(%rm)	yes	yes	NA
0xCB	1101 rrrr	Byte displacement deferred	*expr(%rm)	yes	yes	NA

Legend:
dest = write access (destination) valid.
EA = effective address access valid.
PC = PC (r15) in register field.
NA = not applicable.
expr = user supplied expression that yields a byte, halfword, or word.
n= register number in range 0 to 15.
m = register number in range 16 to 31.
rrrr = r16—r31, where the value represented is 0—15, respectively.
rrrrr = register number in range 0 to 31.

Exceptional Conditions

There are four groups of exceptional conditions: normal, stack, process, and reset. Each group consists of several exceptional conditions. The exception type (ET) field in the processor status word (PSW) indicates which of the four groups caused an exception, and the internal state code (ISC) field distinguishes between exceptional conditions in each group. Refer to Table 12 for a list of ISC codes associated with each exception type.

Normal Exceptions. Normal exceptions consist of three types of events generated by the microprocessor: traps, internal faults, and external faults. When a trap is generated, the instruction that caused the trap is completed and the PC points to the next executable instruction. Note that integer overflows may or may not behave in this way due to pipelining. When an internal or external fault is generated, the PC points to the opcode of the instruction that caused the exception; this instruction may have been executed partially or not at all. Each

different trap or exception uses a different trap vector to branch to the corresponding exception handling software.

There are three kinds of traps:

- **Breakpoint Trap.** This trap is invoked whenever the breakpoint trap (BPT) instruction is executed.
- **Integer Overflow.** This trap is enabled when the overflow enable (OE) bit in the PSW is set. Overflow trapping has the following behavior:
  - Whenever an overflow trap occurs, the OE bit is cleared before the PSW is saved.
  - When an instruction causes an overflow trap to occur, the next instruction may or may not be executed before the microsequence is entered. Therefore, the saved PC may point to the instruction following the trapped instruction, or to the next instruction after that one. Even if the instruction following the trapped instruction is executed, flags may or may not be set.
  - If two consecutive instructions cause overflow traps, only one overflow trap will occur.
  - An overflow trap occurs if the OE bit is set and the execution of an instruction causes the V (overflow) bit in the PSW to be set after the instruction is completed. For return from gate (RETG), return to process (RETPS), return from quick interrupt (RETQINT) instructions or an explicit move to the PSW, if the new PSW has the OE bit and the V bit set, then an overflow trap occurs.
  - When an overflow trap occurs, the overflow flag may or may not be set in the PSW that is saved.
- **Trace Trap.** Trace trapping is enabled when the trace enable (TE) bit in the PSW is set. This causes a trace trap to occur after each instruction is executed, except for the RETPS, CALLPS, RETG, and RETQINT instructions.

The ten types of internal and external faults generated by the microprocessor are:

- **External Memory.** This exception occurs when alignment requirements are violated and when arbitrary byte alignment is disabled, if an external device asserts the $\overline{\text{FAULT}}$ input on an access, if an exception occurs during a coprocessor status fetch, or if no coprocessor responds to a coprocessor broadcast. The following list describes the alignment exception behavior when arbitrary byte alignment is disabled:
  - No alignment exception ever occurs on a byte access.
  - No alignment exception ever occurs on an instruction fetch access.
  - If arbitrary byte alignment is turned off (EA = 0), an alignment exception occurs if the access is a data access of word length and address bit 1 (ADDR01) and/or address bit 0 (ADDR00) is 1.
  - If arbitrary byte alignment is turned off (EA = 0), an alignment exception occurs if the access is a data access of halfword length and address bit 0 (ADDR00) is 1.
- **Gate Vector.** This exception is caused by a memory fault when reading gate tables during a GATE instruction.
- **Illegal Level Change.** This exception is caused by an attempt to increase the current execution privilege on a return from gate (RETG) instruction.
- **Illegal Opcode.** The opcode is not defined for the microprocessor.
- **Integer Zero-Divide.** This exception is caused by an attempt to divide by zero and is always enabled. This exception resets the PSW flags.
- **Invalid Descriptor.** Address mode generated cannot be used in the way specified below. This exception resets the PSW flags and may result from the following causes:
  - Literal or immediate used as destination
  - Effective address requested of literal or immediate
  - Effective address requested of a register.
- **Privileged Opcode.** The opcode is defined for kernel-execution level only. An attempt to execute it in another execution level causes this exception.
- **Privileged Register.** An attempt to write into a privileged register in an execution level other than kernel causes this exception. This exception resets the PSW flags.
- **Reserved Data Type.** The operand type described by the expanded-operand descriptor type is not implemented on the

microprocessor. This exception resets the PSW flags.

- **Reserved Opcode.** The opcode is not implemented on the microprocessor, but is reserved for future use.

Stack Exceptions. This exception may occur during a process switch or a gate sequence. The three types of stack exceptions are:

- **Interrupt ID Fetch.** A memory exception during the interrupt acknowledge.
- **Stack Bound.** A stack pointer (SP) value outside the upper or lower stack bound on a system call.
- **Stack Fault.** A memory exception when storing the PC or PSW on the execution stack during a system call.

Process Exceptions. These exceptions may occur during a process switch. The three types of process exceptions are:

- **Gate Process Control Block.** A memory exception when accessing the process control block (PCB) for a stack bounds check during a GATE.
- **New Process Control Block.** A memory exception when accessing the PCB for the new process during a process switch.
- **Old Process Control Block.** A memory exception when accessing the PCB for the exiting process on a process switch.

Reset Exceptions. These exceptions are triggered by an error condition in accessing critical system data and require restarting the system. Reset exceptions may occur during reset or during process and normal exceptions. The six types of reset exceptions are:

- **External Reset.** Normal response to an external (system) reset signal.
- **Gate Vector.** A memory exception when accessing a gate table while processing a normal exception.
- **Interrupt Stack.** A memory exception when accessing the interrupt stack.
- **New Process Control Block.** A memory exception when accessing the PCB of an exception-handler process.
- **Old Process Control Block.** A memory exception when accessing the PCB of a process exception-handler.
- **System Data.** A memory exception when accessing the PCBPs for stack, process, or reset exceptions.

Instruction Set Summary by Functional Group

Tables 16 through 23 summarize the instruction set by functional group. The entries in the conditions column refer to the condition flag code assignment cases listed in Table 24.

The *WE* 32200 Microprocessor possesses an extensive instruction set. The instruction set includes special coprocessor instructions that implement interfacing with coprocessors. Instructions may appear at any byte address in memory and consist of a one- or two-byte operation code (opcode) followed by zero or more operand descriptors. The opcode specifies the desired operation to be performed and the operand descriptors provide the information needed to locate the operand. The microprocessor supports four classes of instructions:

Monadic Instructions. These instructions contain one operand, which serves as source, destination, or both, e.g., clear, increment.

Dyadic Instructions. These instructions contain two operands. In one type, both operands are designated as sources, as is the case for the compare instruction. The contents of the operands will not be altered. A second type of dyadic instruction designates one of the operands as the source and the other as the destination, e.g., move instructions. A third type of dyadic instruction designates one of the operands as the source and the other as both source and destination, e.g., two-address arithmetic and logical instructions. In this case, the result of the operation is stored in the destination.

Triadic Instructions. These instructions contain three operands: two designated as sources (the first and second), and one (the third) as the destination. The result of the operation is stored in the destination, e.g., three-address arithmetic and logical operations.

Program Control and Miscellaneous Instructions. These instructions alter the order of instruction execution and have operands that are not described by operand descriptors. Program control instructions are of three types:

- Conditional, where the operation performed is based on the result of a test, e.g., conditional branch and conditional return instructions
- Unconditional, where the operation is always performed, e.g. jump and call instructions
- Loop control instructions, where the operation performed is based on the result of a test and the result of a counter that is decremented.

An example of an instruction with no operand descriptor is cache flush (CFLUSH).

Arithmetic Group

These instructions perform arithmetic operations on data in registers and memory.

Table 16. Arithmetic Group

Instruction	Mnemonic	Opcode	Conditions*
Add			
Add byte	ADDB2	0x9F	
Add halfword	ADDH2	0x9E	
Add word	ADDW2	0x9C	
Add byte, 3-address	ADDB3	0xDF	
Add halfword, 3-address	ADDH3	0xDE	
Add word, 3-address	ADDW3	0xDC	
Subtract			
Subtract byte	SUBB2	0xBF	
Subtract halfword	SUBH2	0xBE	
Subtract word	SUBW2	0xBC	Case 2
Subtract byte, 3-address	SUBB3	0xFF	
Subtract halfword, 3-address	SUBH3	0xFE	
Subtract word, 3-address	SUBW3	0xFC	
Increment			
Increment byte	INCB	0x93	
Increment halfword	INCH	0x92	
Increment word	INCW	0x90	
Decrement			
Decrement byte	DECB	0x97	
Decrement halfword	DECH	0x96	
Decrement word	DECW	0x94	
Multiply			
Multiply byte	MULB2	0xAB	
Multiply halfword	MULH2	0xAA	Case 3
Multiply word	MULW2	0xA8	
Multiply byte, 3-address	MULB3	0xEB	
Multiply halfword, 3-address	MULH3	0xEA	Case 4
Multiply word, 3-address	MULW3	0xE8	

* Refer to Table 24 for condition flag code assignments.

Table 16. Arithmetic Group (Continued)

Instruction	Mnemonic	Opcode	Conditions*
Divide			
Divide byte	DIVB2	0xAF	
Divide halfword	DIVH2	0xAE	Case 3
Divide word	DIVW2	0xAC	
Divide byte, 3-address	DIVB3	0xEF	
Divide halfword, 3-address	DIVH3	0xEE	Case 4
Divide word, 3-address	DIVW3	0xEC	
Modulo			
Modulo byte	MODB2	0xA7	
Modulo halfword	MODH2	0xA6	Case 3
Modulo word	MODW2	0xA4	
Modulo byte, 3-address	MODB3	0xE7	
Modulo halfword, 3-address	MODH3	0xE6	Case 4
Modulo word, 3-address	MODW3	0xE4	
Arithmetic Shift			
Arithmetic left shift word	ALSW3	0xC0	Case 5
Arithmetic right shift byte	ARSB3	0xC7	
Arithmetic right shift halfword	ARSH3	0xC6	Case 3
Arithmetic right shift word	ARSW3	0xC4	

* Refer to Table 24 for condition flag code assignments.

Data Transfer Group

These instructions transfer data to and from registers and memory.

Table 17. Data Transfer Group

Instruction	Mnemonic	Opcode	Conditions*
Move			
Move byte	MOVB	0x87	
Move halfword	MOVH	0x86	
Move word	MOVW	0x84	Case 1
Move address (word)	MOVAW	0x04	
Move complemented byte	MCOMB	0x8B	
Move complemented halfword	MCOMH	0x8A	
Move complemented word	MCOMW	0x88	
Move negated byte	MNEGB	0x8F	
Move negated halfword	MNEGH	0x8E	Case 2
Move negated word	MNEGW	0x8C	
Move version number	MVERNO	0x3009	Unchanged
Swap (Interlocked)			
Swap byte interlocked	SWAPBI	0x1F	
Swap halfword interlocked	SWAPHI	0x1E	Case 1
Swap word interlocked	SWAPWI	0x1C	

* Refer to Table 24 for condition flag code assignments.

Table 17. Data Transfer Group (Continued)

Instruction	Mnemonic	Opcode	Conditions*
Compare and Swap			
Compare and swap word interlocked	CASWI	0x09	Case 2
Block Operations			
Move block of words	MOVBLW	0x3019	Unchanged
Field Operations			
Extract field byte	EXTFB	0xCF	Case 1
Extract field halfword	EXTFH	0xCE	
Extract field word	EXTFW	0xCC	
Insert field byte	INSFB	0xCB	
Insert field halfword	INSFH	0xCA	
Insert field word	INSFW	0xC8	
String Operations			
String copy	STRCPY	0x3035	Unchanged
String end	STREND	0x301F	

* Refer to Table 24 for condition flag code assignments.

Logical Group

These instructions perform logical operations on data in registers and memory.

Table 18. Logical Group

Instruction	Mnemonic	Opcode	Conditions*
AND			
AND byte	ANDB2	0xBB	Case 1
AND halfword	ANDH2	0xBA	
AND word	ANDW2	0xB8	
AND byte, 3-address	ANDB3	0xFB	
AND halfword, 3-address	ANDH3	0xFA	
AND word, 3-address	ANDW3	0xF8	
Exclusive OR (XOR)			
Exclusive OR byte	XORB2	0xB7	
Exclusive OR halfword	XORH2	0xB6	
Exclusive OR word	XORW2	0xB4	
Exclusive OR byte 3-address	XORB3	0xF7	
Exclusive OR halfword 3-address	XORH3	0xF6	
Exclusive OR word, 3-address	XORW3	0xF4	
OR			
OR byte	ORB2	0xB3	
OR halfword	ORH2	0xB2	
OR word	ORW2	0xB0	
OR byte, 3-address	ORB3	0xF3	
OR halfword, 3-address	ORH2	0xF2	
OR word, 3-address	ORW3	0xF0	

* Refer to Table 24 for condition flag code assignments.

Table 18. Logical Group (Continued)

Instruction	Mnemonic	Opcode	Conditions*
Compare or Test			
Compare byte	CMPB	0x3F	
Compare halfword	CMPH	0x3E	Case 2
Compare word	CMPW	0x3C	
Test byte	TSTB	0x2B	
Test halfword	TSTH	0x2A	Case 6
Test word	TSTW	0x28	
Bit test byte	BITB	0x3B	
Bit test halfword	BITH	0x3A	Case 1
Bit test word	BITW	0x38	
Clear			
Clear byte	CLRB	0x83	
Clear halfword	CLRH	0x82	Case 2
Clear word	CLRW	0x80	
Rotate or Logical Shift			
Rotate word	ROTW	0xD8	
Logical left shift byte	LLSB3	0xD3	
Logical left shift halfword	LLSH3	0xD2	Case 1
Logical left shift word	LLSW3	0xD0	
Logical right shift word	LRSW3	0xD4	

* Refer to Table 24 for condition flag code assignments.

Binary Coded Decimal (BCD) Group

There are three types of BCD instructions: additive BCD arithmetic, BCD data type conversion, and the BCD flag setting and clearing.

These instructions perform BCD arithmetic, data conversion, and logical operations, respectively, on data in registers and memory.

Table 19. BCD Group

Instruction	Mnemonic	Opcode	Conditions*
Add			
Add packed byte	ADDPB2	0xA3	
Add packed byte, 3-address	ADDPB3	0xE3	Case 7
Subtract			
Subtract packed byte	SUBPB2	0x9B	
Subtract packed byte, 3-address	SUBPB3	0xDB	
Data Conversion			
Pack BCD halfword	PACKB	0x0E	Unchanged
Unpack BCD byte	UNPACKB	0x0F	
Flags			
Clear X bit in PSW	CLRX	0x0B	Only X bit
Set X bit in PSW	SETX	0x0A	in PSW affected

* Refer to Table 24 for condition flag code assignments.

Program Control Group

These instructions alter normal sequential program flow. Unconditional transfers (branch, jump, or return) change the contents of the PC (r15) to the value specified. Conditional transfers first examine the status of the microprocessor flags to determine if the specified transfer should be executed. Loop control instructions are similar to conditional transfers, but with the added feature that the contents of an operand are also examined to determine if a specified transfer should be executed. Subroutine and procedure transfer instructions save or restore registers so control can transfer to the subroutine or procedure and then return to the original program sequence.

Table 20. Program Control Group

Instruction	Mnemonic	Opcode	Conditions*
Unconditional Transfer			
Branch with byte displacement	BRB	0x7B	
Branch with halfword displacement	BRH	0x7A	
Jump	JMP	0x24	
Conditional Transfers			
Branch on carry clear byte	BCCB	0x53**	
Branch on carry clear halfword	BCCH	0x52**	
Branch on carry set byte	BCSB	0x5B**	
Branch on carry set halfword	BCSH	0x5A**	
Branch on overflow clear, byte displacement	BVCB	0x63	
Branch on overflow clear, halfword displacement	BVCH	0x62	Unchanged
Branch on overflow set, byte displacement	BVSB	0x6B	
Branch on overflow set, halfword displacement	BVSH	0x6A	
Branch on equal byte (duplicate)	BEB	0x6F	
Branch on equal byte	BEB	0x7F	
Branch on equal halfword (duplicate)	BEH	0x6E	
Branch on equal halfword	BEH	0x7E	
Branch on not equal byte (duplicate)	BNEB	0x67	
Branch on not equal byte	BNEB	0x77	
Branch on not equal halfword (duplicate)	BNEH	0x66	
Branch on not equal halfword	BNEH	0x76	

* Refer to Table 24 for condition flag code assignments.
** Indicates that another mnemonic has the same opcode.

Table 20. Program Control Group (Continued)

Instruction	Mnemonic	Opcode	Conditions*
Branch on less than byte (signed)	BLB	0x4B	Unchanged
Branch on less than halfword (signed)	BLH	0x4A	
Branch on less than byte (unsigned)	BLUB	0x5B**	
Branch on less than halfword (unsigned)	BLUH	0x5A**	
Branch on less than or equal byte (signed)	BLEB	0x4F	
Branch on less than or equal halfword (signed)	BLEH	0x4E	
Branch on less than or equal byte (unsigned)	BLEUB	0x5F	
Branch on less than or equal halfword (unsigned)	BLEUH	0x5E	
Branch on greater than byte (signed)	BGB	0x47	
Branch on greater than halfword (signed)	BGH	0x46	
Branch on greater than byte (unsigned)	BGUB	0x57	
Branch on greater than halfword (unsigned)	BGUH	0x56	
Branch on greater than or equal byte (signed)	BGEB	0x43	
Branch on greater than or equal halfword (signed)	BGEH	0x42	
Branch on greater than or equal byte (unsigned)	BGEUB	0x53**	
Branch on greater than or equal halfword (unsigned)	BGEUH	0x52**	
Return on carry clear	RCC	0x50**	
Return on carry set	RCS	0x58**	
Return on overflow clear	RVC	0x60	
Return on overflow set	RVS	0x68	
Return on equal (unsigned)	REQLU	0x6C†	
Return on equal	REQL	0x7C†	
Return on not equal (unsigned)	RNEQU	0x64††	
Return on not equal	RNEQ	0x74††	

* Refer to Table 24 for condition flag code assignments.
** Indicates that another mnemonic has the same opcode.
† REQLU and REQL perform the same operation.
†† RNEQU and RNEQ perform the same operation.

Table 20. Program Control Group (Continued)

Instruction	Mnemonic	Opcode	Conditions*
Return on less than (signed)	RLSS	0x48	Unchanged
Return on less than (unsigned)	RLSSU	0x58**	
Return on less than or equal (signed)	RLEQ	0x4C	
Return on less than or equal (unsigned)	RLEQU	0x5C	
Return on greater than (signed)	RGTR	0x44	
Return on greater than (unsigned)	RGTRU	0x54	
Return on greater than or equal (signed)	RGEQ	0x40	
Return on greater than or equal (unsigned)	RGEQU	0x50**	
Loop Control			Unchanged
Decrement, test, and branch byte (signed)	DTB	0x29	
Decrement, test, and branch halfword (signed)	DTH	0x19	
Test equal, decrement, and test byte	TEDTB	0x4D	
Test equal, decrement, and test halfword	TEDTH	0x0D	
Test greater, decrement, and test byte	TGDTB	0x6D	
Test greater, decrement, and test halfword	TGDTH	0x2D	
Test greater or equal, decrement, and test byte	TGEDTB	0x5D	
Test greater or equal, decrement, and test halfword	TGEDTH	0x1D	
Test not equal, decrement, and test byte	TNEDTB	0x7D	
Test not equal, decrement, and test halfword	TNEDTH	0x3D	
Subroutine Transfer			
Branch to subroutine, byte displacement	BSBB	0x37	
Branch to subroutine, halfword displacement	BSBH	0x36	
Jump to subroutine	JSB	0x34	
Return from subroutine	RSB	0x78	
Procedure Transfer			
Save registers	SAVE	0x10	
Restore registers	RESTORE	0x18	
Call procedure	CALL	0x2C	
Return from procedure	RET	0x08	

* Refer to Table 24 for condition flag code assignments.
** Indicates that another mnemonic has the same opcode.

Stack and Miscellaneous Groups

These instructions manipulate the stack and alter the machine state.

Table 21. Stack and Miscellaneous Groups

Instruction	Mnemonic	Opcode	Conditions*
Stack Operations			
Push address word	PUSHAW	0xE0	
Push word	PUSHW	0xA0	Case 1
Pop word	POPW	0x20	
Miscellaneous			
No operation, 1 byte	NOP	0x70	
No operation, 2 byte	NOP2	0x73	
No operation, 3 byte	NOP3	0x72	Unchanged
Breakpoint trap	BPT	0x2E	
Cache flush	CFLUSH	0x27	
Extended opcode	EXTOP	0x14	

* Refer to Table 24 for condition flag code assignments.

Coprocessor Group Instruction

These instructions implement the interface with coprocessors.

Table 22. Coprocessor Group

Instruction	Mnemonic	Opcode	Conditions*
Coprocessor operation	SPOP	0x32	
Coprocessor operation read single	SPOPRS	0x22	
Coprocessor operation read double	SPOPRD	0x02	
Coprocessor operation read triple	SPOPRT	0x06	
Coprocessor operation single 2-address	SPOPS2	0x23	Case 11
Coprocessor operation double 2-address	SPOPD2	0x03	
Coprocessor operation triple 2-address	SPOPT2	0x07	
Coprocessor operation write single	SPOPWS	0x33	
Coprocessor operation write double	SPOPWD	0x13	
Coprocessor operation write triple	SPOPWT	0x17	

* Refer to Table 24 for condition flag code assignments.

Operating System Instructions and Microsequences

These instructions manipulate, via a sequence of actions, the context (internal registers and pointers) of the microprocessor to permit changing the process that is currently in control.

The instruction or microsequence is classified as privileged or nonprivileged. Privileged instructions are executed only if the kernel execution level is in effect. Nonprivileged instructions do not depend upon the execution privilege level.

Table 23. Operating System Group

Instruction	Mnemonic	Opcode	Conditions*	Instruction	Mnemonic	Opcode	Conditions*
Nonprivileged Instructions Move translated word	MOVTRW	0x0C	Case 8	**Privileged Instructions** Call process	CALLPS	0x30AC	Case 9
				Return to process	RETPS	0x30C8	Case 10
Gate	GATE	0x3061	Case 9	Wait for interrupt	WAIT	0x2F	Unchanged
Return from gate	RETG	0x3045	Case 10	Interrupt acknowledge	INTACK	0x302F	
Privileged Instructions Enable virtual pin and jump	ENBVJMP	0x300D	Unchanged	Return from quick interrupt	RETQINT	0x98	Case 10
Disable virtual pin and jump	DISVJMP	0x3013		User Call Process	UCALLPS	0x30C0	Case 9

* Refer to Table 24 for condition flag code assignments.

Table 24. Condition Flag Code Assignments

Case	Condition Flags* (Negative) N	(Zero) Z	(Carry) C	(External Carry) X	(Overflow) V	Special Conditions**
1	MSB of dst	1 if dst = 0	0	No change	0	V flat is set when expanded-operand mode is used and the result is truncated when represented in dst.
2	1 if result < 0	1 if result = 0	1 on carry or borrow	No change	1 on integer overflow	—
3	1 if dst < 0	1 if dst = 0	0	No change	1 on integer overflow	—
4	1 if dst < 0	1 if dst = 0	0	No change	1 on integer overflow	V flag may not be set when dst is a signed word and bit 31 of absolute value of the result is 1 and bits 32—63 are 0s.

* MSB is most significant bit, dst is destination, and src is source.
** Cases 1 through 6. When PSW is the source, condition flags are unaffected. When PSW is destination, condition flags assume the value of bits 18—21 of the operation performed.

Table 24. Condition Flag Code Assignments (Continued)

Case	Condition Flags* (Negative) N	(Zero) Z	(Carry) C	(External Carry) X	(Overflow) V	Special Conditions**
5	1 if dst < 0	1 if dst = 0	0	No change	0	V flag is set if expanded-operand mode changes the type of dst and integer overflow occurs.
6	1 if scr < 0	1 if scr = 0	0	No change	0	N flag is affected if src is signed integer.
7	0	1 if result = 0	1 on BCD carry or borrow	1 on BCD carry or borrow	No change	—
8	MSB of word returned	1 if word returned = 0	0	No change	0	—
9	—	—	—	—	—	All flags determined by new PSW.
10	—	—	—	—	—	All flags determined by restored PSW.
11	—	—	—	—	—	When coprocessor status word is accepted, bits 18—21 of the word read are put into bits 18—21 of the PSW, respectively.

* MSB is most significant bit, dst is destination, and src is source.
** Cases 1 through 6. When PSW is the source, condition flags are unaffected. When PSW is destination, condition flags assume the value of bits 18—21 of the operation performed.

Other Microsequences

The microprocessor automatically executes a sequence of actions in response to interrupts, exceptions (normal, stack, process, or reset), or context switches. The following are the corresponding microsequences performed for each of these conditions:

On-Interrupt. Used on an interrupt to perform an implicit process switch from the interrupted process to the interrupt handler or to perform the quick-interrupt action.

On-Normal Exception. Performs a function similar to that of a GATE to transfer control to the appropriate exception-handler function (determined from the internal state code (ISC) field in the PSW), when the EX/UC bit in the PSW is cleared. With the EX/UC option in the PSW set, this exception performs a sequence similar to the process switch.

On-Stack Exception. Performed when a stack bounds violation is detected during a GATE instruction or a normal exception.

On-Process Exception. Performed when the microprocessor receives a memory fault signal on a PCB access or other process exception.

On-Reset Exception. Used to restart the system when an error condition in accessing critical system data is detected or when an external reset request occurs.

Context Switches. Three context switch microsequences available are: XSWITCH_ONE, XSWITCH_TWO, and XSWITCH_THREE. Various combinations of the three are used to save the context (control registers) for the current process, load the context of the new process, and execute block moves for the new process.

Instruction Set Summaries by Mnemonic and Opcode

Table 25 lists the instruction set mnemonics alphabetically. The hexadecimal opcode and a brief description of the instruction are presented for each mnemonic. Opcodes are either one or two bytes, depending on the instruction.

Table 26 lists instructions by opcode.

Opcodes are flagged with an asterisk (*) if the opcode matches another instruction and the operation performed is identical. Mnemonics are flagged with double asterisks (**) to indicate an operating system instruction.

Table 25. Instruction Set Summary by Mnemonic

Mnemonic	Opcode	Instruction
ADDB2	0x9F	Add byte
ADDB3	0xDF	Add byte, 3-address
ADDH2	0x9E	Add halfword
ADDH3	0xDE	Add halfword, 3-address
ADDPB2	0xA3	Add packed BCD
ADDPB3	0xE3	Add packed BCD, 3-address
ADDW2	0x9C	Add word
ADDW3	0xDC	Add word, 3-address
ALSW3	0xC0	Arithmetic left shift word
ANDB2	0xBB	AND byte
ANDB3	0xFB	AND byte, 3-address
ANDH2	0xBA	AND halfword
ANDH3	0xFA	AND halfword, 3-address
ANDW2	0xB8	AND word
ANDW3	0xF8	AND word, 3-address
ARSB3	0xC7	Arithmetic right shift byte
ARSH3	0xC6	Arithmetic right shift halfword
ARSW3	0xC4	Arithmetic right shift word
BCCB	0x53*	Branch on carry clear byte
BCCH	0x52*	Branch on carry clear halfword
BCSB	0x5B*	Branch on carry set byte
BCSH	0x5A*	Branch on carry set halfword
BEB	0x6F	Branch on equal byte (duplicate)
BEB	0x7F	Branch on equal byte
BEH	0x6E	Branch on equal halfword (duplicate)
BEH	0x7E	Branch on equal halfword
BGB	0x47	Branch on greater than byte (signed)
BGEB	0x43	Branch on greater than or equal byte (signed)
BGEH	0x42	Branch on greater than or equal halfword (signed)
BGEUB	0x53*	Branch on greater than or equal byte (unsigned)
BGEUH	0x52*	Branch on greater than or equal halfword (unsigned)
BGH	0x46	Branch on greater than halfword (signed)
BGUB	0x57	Branch on greater than byte (unsigned)
BGUH	0x56	Branch on greater than halfword (unsigned)
BITB	0x3B	Bit test byte
BITH	0x3A	Bit test halfword
BITW	0x38	Bit test word

* Another mnemonic has the same opcode.

Table 25. Instruction Set Summary by Mnemonic (Continued)

Mnemonic	Opcode	Instruction
BLB	0x4B	Branch on less than byte (signed)
BLEB	0x4F	Branch on less than or equal byte (signed)
BLEH	0x4E	Branch on less than or equal halfword (signed)
BLEUB	0x5F	Branch on less than or equal byte (unsigned)
BLEUH	0x5E	Branch on less than or equal halfword (unsigned)
BLH	0x4A	Branch on less than halfword (signed)
BLUB	0x5B*	Branch on less than byte (unsigned)
BLUH	0x5A*	Branch on less than halfword (unsigned)
BNEB	0x67	Branch on not equal byte (duplicate)
BNEB	0x77	Branch on not equal byte
BNEH	0x66	Branch on not equal halfword (duplicate)
BNEH	0x76	Branch on not equal halfword
BPT	0x2E	Breakpoint trap
BRB	0x7B	Branch with byte displacement
BRH	0x7A	Branch with halfword displacement
BSBB	0x37	Branch to subroutine, byte displacement
BSBH	0x36	Branch to subroutine, halfword displacement
BVCB	0x63	Branch on overflow clear, byte displacement
BVCH	0x62	Branch on overflow clear, halfword displacement
BVSB	0x6B	Branch on overflow set, byte displacement
BVSH	0x6A	Branch on overflow set, halfword displacement
CALL	0x2C	Call procedure
CALLPS**	0x30AC	Call process
CASWI	0x09	Compare and swap word interlocked
CFLUSH	0x27	Cache flush
CLRB	0x83	Clear byte
CLRH	0x82	Clear halfword
CLRW	0x80	Clear word
CLRX	0x0B	Clear X bit in PSW
CMPB	0x3F	Compare byte
CMPH	0x3E	Compare halfword
CMPW	0x3C	Compare word
DECB	0x97	Decrement byte
DECH	0x96	Decrement halfword
DECW	0x94	Decrement word
DISVJMP**	0x3013	Disable virtual pin and jump
DIVB2	0xAF	Divide byte
DIVB3	0xEF	Divide byte, 3-address
DIVH2	0xAE	Divide halfword
DIVH3	0xEE	Divide halfword, 3-address
DIVW2	0xAC	Divide word
DIVW3	0xEC	Divide word, 3-address
DTB	0x29	Decrement, test, and branch byte
DTH	0x19	Decrement, test, and branch halfword

* Another mnemonic has the same opcode.
** Operating system instruction.

Table 25. Instruction Set Summary by Mnemonic (Continued)

Mnemonic	Opcode	Instruction
ENBVJMP**	0x300D	Enable virtual pin and jump
EXTFB	0xCF	Extract field byte
EXTFH	0xCE	Extract field halfword
EXTFW	0xCC	Extract field word
EXTOP	0x14	Extended opcode
GATE**	0x3061	Gate
INCB	0x93	Increment byte
INCH	0x92	Increment halfword
INCW	0x90	Increment word
INSFB	0xCB	Insert field byte
INSFH	0xCA	Insert field halfword
INSFW	0xC8	Insert field word
INTACK**	0x302F	Interrupt acknowledge
JMP	0x24	Jump
JSB	0x34	Jump to subroutine
LLSB3	0xD3	Logical left shift byte
LLSH3	0xD2	Logical left shift halfword
LLSW3	0xD0	Logical left shift word
LRSW3	0xD4	Logical right shift word
MCOMB	0x8B	Move complemented byte
MCOMH	0x8A	Move complemented halfword
MCOMW	0x88	Move complemented word
MNEGB	0x8F	Move negated byte
MNEGH	0x8E	Move negated halfword
MNEGW	0x8C	Move negated word
MODB2	0xA7	Modulo byte
MODB3	0xE7	Modulo byte, 3-address
MODH2	0xA6	Modulo halfword
MODH3	0xE6	Modulo halfword, 3-address
MODW2	0xA4	Modulo word
MODW3	0xE4	Modulo word, 3-address
MOVAW	0x04	Move address (word)
MOVB	0x87	Move byte
MOVBLW	0x3019	Move block of words
MOVH	0x86	Move halfword
MOVTRW**	0x0C	Move translated word
MOVW	0x84	Move word
MULB2	0xAB	Multiply byte
MULB3	0xEB	Multiply byte, 3-address
MULH2	0xAA	Multiply halfword
MULH3	0xEA	Multiply halfword, 3-address
MULW2	0xA8	Multiply word
MULW3	0xE8	Multiply word, 3-address
MVERNO	0x3009	Move version number

** Operating system instruction.

Table 25. Instruction Set Summary by Mnemonic (Continued)

Mnemonic	Opcode	Instruction
NOP	0x70	No operation, 1 byte
NOP2	0x73	No operation, 2 byte
NOP3	0x72	No operation, 3 byte
ORB2	0xB3	OR byte
ORB3	0xF3	OR byte, 3-address
ORH2	0xB2	OR halfword
ORH3	0xF2	OR halfword, 3-address
ORW2	0xB0	OR word
ORW3	0xF0	OR word, 3-address
PACKB	0x0E	Pack BCD
POPW	0x20	Pop word
PUSHAW	0xE0	Push address word
PUSHW	0xA0	Push word
RCC	0x50*	Return on carry clear
RCS	0x58*	Return on carry set
REQLU	0x6C†	Return on equal (unsigned)
REQL	0x7C†	Return on equal (signed)
RESTORE	0x18	Restore registers
RET	0x08	Return from procedure
RETG**	0x3045	Return from gate
RETPS**	0x30C8	Return to process
RETQINT**	0x98	Return from quick interrupt
RGEQ	0x40	Return on greater than or equal (signed)
RGEQU	0x50*	Return on greater than or equal (unsigned)
RGTR	0x44	Return on greater than (signed)
RGTRU	0x54	Return on greater than (unsigned)
RLEQ	0x4C	Return on less than or equal (signed)
RLEQU	0x5C	Return on less than or equal (unsigned)
RLSS	0x48	Return on less than (signed)
RLSSU	0x58*	Return on less than (unsigned)
RNEQU	0x64††	Return on not equal (unsigned)
RNEQ	0x74††	Return on not equal (signed)
ROTW	0xD8	Rotate word
RSB	0x78	Return from subroutine
RVC	0x60	Return on overflow clear
RVS	0x68	Return on overflow set
SAVE	0x10	Save registers
SETX	0x0A	Set X bit in PSW
SPOP	0x32	Coprocessor operation
SPOPD2	0x03	Coprocessor operation double, 2-address
SPOPRD	0x02	Coprocessor operation read double
SPOPRS	0x22	Coprocessor operation read single

* Another mnemonic has the same opcode.
** Operating system instruction.
† REQLU and REQL perform the same operation.
†† RNEQU and RNEQ perform the same operation.

Table 25. Instruction Set Summary by Mnemonic (Continued)

Mnemonic	Opcode	Instruction
SPOPRT	0x06	Coprocessor operation read triple
SPOPS2	0x23	Coprocessor operation single, 2-address
SPOPT2	0x07	Coprocessor operation triple, 2-address
SPOPWD	0x13	Coprocessor operation write double
SPOPWS	0x33	Coprocessor operation write single
SPOPWT	0x17	Coprocessor operation write triple
STRCPY	0x3035	String copy
STREND	0x301F	String end
SUBB2	0xBF	Subtract byte
SUBB3	0xFF	Subtract byte, 3-address
SUBH2	0xBE	Subtract halfword
SUBH3	0xFE	Subtract halfword, 3-address
SUBPB2	0x9B	Subtract packed BCD
SUBPB3	0xDB	Subtract packed BCD, 3-address
SUBW2	0xBC	Subtract word
SUBW3	0xFC	Subtract word, 3-address
SWAPBI	0x1F	Swap byte interlocked
SWAPHI	0x1E	Swap halfword interlocked
SWAPWI	0x1C	Swap word interlocked
TEDTB	0x4D	Test equal, decrement and test byte
TGEDTB	0x5D	Test greater or equal, decrement and test byte
TGDTB	0x6D	Test greater, decrement and test byte
TNEDTB	0x7D	Test not equal, decrement and test byte
TEDTH	0x0D	Test equal, decrement and test halfword
TGEDTH	0x1D	Test greater or equal, decrement and test halfword
TGDTH	0x2D	Test greater, decrement and test halfword
TNEDTH	0x3D	Test not equal, decrement and test halfword
TSTB	0x2B	Test byte
TSTH	0x2A	Test halfword
TSTW	0x28	Test word
UCALLPS**	0x30C0	User call process
UNPACKB	0x0F	Unpack BCD
WAIT**	0x2F	Wait for interrupt
XORB2	0xB7	Exclusive OR byte
XORB3	0xF7	Exclusive OR byte, 3-address
XORH2	0xB6	Exclusive OR halfword
XORH3	0xF6	Exclusive OR halfword, 3-address
XORW2	0xB4	Exclusive OR word
XORW3	0xF4	Exclusive OR word, 3-address

** Operating system instruction.

Table 26. Instruction Set Summary by Opcode

Opcode	Mnemonic	Instruction
0x02	SPOPRD	Coprocessor operation read double
0x03	SPOPD2	Coprocessor operation double, 2-address
0x04	MOVAW	Move address (word)
0x06	SPOPRT	Coprocessor operation read triple
0x07	SPOPT2	Coprocessor operation triple, 2-address
0x08	RET	Return from procedure
0x09	CASWI	Compare and swap word interlock
0x0A	SETX	Set X bit in PSW
0x0B	CLRX	Clear X bit in PSW
0x0C	MOVTRW**	Move translated word
0x0D	TEDTH	Test equal, decrement and test halfword
0x0E	PACKB	Pack BCD
0x0F	UNPACKB	Unpack BCD
0x10	SAVE	Save registers
0x13	SPOPWD	Coprocessor operation write double
0x14	EXTOP	Extended opcode
0x17	SPOPWT	Coprocessor operation write triple
0x18	RESTORE	Restore registers
0x19	DTH	Decrement, test, and branch halfword
0x1C	SWAPWI	Swap word interlocked
0x1D	TGEDTH	Test greater or equal, decrement and test halfword
0x1E	SWAPHI	Swap halfword interlocked
0x1F	SWAPBI	Swap byte interlocked
0x20	POPW	Pop word
0x22	SPOPRS	Coprocessor operation read single
0x23	SPOPS2	Coprocessor operation single, 2-address
0x24	JMP	Jump
0x27	CFLUSH	Cache flush
0x28	TSTW	Test word
0x29	DTB	Decrement, test, and branch byte
0x2A	TSTH	Test halfword
0x2B	TSTB	Test byte
0x2C	CALL	Call procedure
0x2D	TGDTH	Test Greater, decrement and test halfword
0x2E	BPT	Breakpoint trap
0x2F	WAIT**	Wait for interrupt
0x3009	MVERNO	Move version number
0x300D	ENBVJMP**	Enable virtual pin and jump
0x3013	DISVJMP**	Disable virtual pin and jump
0x3019	MOVBLW	Move block of words
0x301F	STREND	String end
0x302F	INTACK**	Interrupt acknowledge
0x3035	STRCPY	String copy
0x3045	RETG**	Return from gate
0x3061	GATE**	Gate
0x30AC	CALLPS**	Call process
0x30C0	UCALLPS**	User call process
0x30C8	RETPS**	Return to process

** Operating system instruction.

Table 26. Instruction Set Summary by Opcode (Continued)

Opcode	Mnemonic	Instruction
0x32	SPOP	Coprocessor operation
0x33	SPOPWS	Coprocessor operation write single
0x34	JSB	Jump to subroutine
0x36	BSBH	Branch to subroutine, halfword displacement
0x37	BSBB	Branch to subroutine, byte displacement
0x38	BITW	Bit test word
0x3A	BITH	Bit test halfword
0x3B	BITB	Bit test byte
0x3C	CMPW	Compare word
0x3D	TNEDTH	Test not equal, decrement and test halfword
0x3E	CMPH	Compare halfword
0x3F	CMPB	Compare byte
0x40	RGEQ	Return on greater than or equal (signed)
0x42	BGEH	Branch on greater than or equal halfword (signed)
0x43	BGEB	Branch on greater than or equal byte (signed)
0x44	RGTR	Return on greater than (signed)
0x46	BGH	Branch on greater than halfword (signed)
0x47	BGB	Branch on greater than byte (signed)
0x48	RLSS	Return on less than (signed)
0x4A	BLH	Branch on less than halfword (signed)
0x4B	BLB	Branch on less than byte (signed)
0x4C	RLEQ	Return on less than or equal (signed)
0x4D	TEDTB	Test equal, decrement and test byte
0x4E	BLEH	Branch on less than or equal halfword (signed)
0x4F	BLEB	Branch on less than or equal byte (signed)
0x50*	RCC	Return on carry clear
0x50*	RGEQU	Return on greater than or equal (unsigned)
0x52*	BCCH	Branch on carry clear halfword
0x52*	BGEUH	Branch on greater than or equal halfword (unsigned)
0x53*	BCCB	Branch on carry clear byte
0x53*	BGEUB	Branch on greater than or equal byte (unsigned)
0x54	RGTRU	Return on greater than (unsigned)
0x56	BGUH	Branch on greater than halfword (unsigned)
0x57	BGUB	Branch on greater than byte (unsigned)
0x58*	RCS	Return on carry set
0x58*	RLSSU	Return on less than (unsigned)
0x5A*	BCSH	Branch on carry set halfword
0x5A*	BLUH	Branch on less than halfword (unsigned)
0x5B*	BCSB	Branch on carry set byte
0x5B*	BLUB	Branch on less than byte (unsigned)
0x5C	RLEQU	Return on less than or equal (unsigned)
0x5D	TGEDTB	Test greater or equal, decrement and test byte
0x5E	BLEUH	Branch on less than or equal halfword (unsigned)
0x5F	BLEUB	Branch on less than or equal byte (unsigned)
0x60	RVC	Return on overflow clear
0x62	BVCH	Branch on overflow clear, halfword displacement

* Another mnemonic has the same opcode.

Table 26. Instruction Set Summary by Opcode (Continued)

Opcode	Mnemonic	Instruction
0x63	BVCB	Branch on overflow clear, byte displacement
0x64†	RNEQU	Return on not equal (unsigned)
0x66	BNEH	Branch on not equal halfword (duplicate)
0x67	BNEB	Branch on not equal byte (duplicate)
0x68	RVS	Return on overflow set
0x6A	BVSH	Branch on overflow set, halfword displacement
0x6B	BVSB	Branch on overflow set, byte displacement
0x6C††	REQLU	Return on equal (unsigned)
0x6D	TGDTB	Test greater, decrement and test byte
0x6E	BEH	Branch on equal halfword (duplicate)
0x6F	BEB	Branch on equal byte (duplicate)
0x70	NOP	No operation, 1 byte
0x72	NOP3	No operation, 3 byte
0x73	NOP2	No operation, 2 byte
0x74†	RNEQ	Return on not equal (signed)
0x76	BNEH	Branch on not equal halfword
0x77	BNEB	Branch on not equal byte
0x78	RSB	Return from subroutine
0x7A	BRH	Branch with halfword displacement
0x7B	BRB	Branch with byte displacement
0x7C††	REQL	Return on equal (signed)
0x7D	TNEDTB	Test not equal, decrement and test byte
0x7E	BEH	Branch on equal halfword
0x7F	BEB	Branch on equal byte
0x80	CLRW	Clear word
0x82	CLRH	Clear halfword
0x83	CLRB	Clear byte
0x84	MOVW	Move word
0x86	MOVH	Move halfword
0x87	MOVB	Move byte
0x88	MCOMW	Move complemented word
0x8A	MCOMH	Move complemented halfword
0x8B	MCOMB	Move complemented byte
0x8C	MNEGW	Move negated word
0x8E	MNEGH	Move negated halfword
0x8F	MNEGB	Move negated byte
0x90	INCW	Increment word
0x92	INCH	Increment halfword
0x93	INCB	Increment byte
0x94	DECW	Decrement word
0x96	DECH	Decrement halfword
0x97	DECB	Decrement byte
0x98	RETQINT**	Return from quick interrupt
0x9B	SUBPB2	Subtract packed BCD, 2-address
0x9C	ADDW2	Add word
0x9E	ADDH2	Add halfword
0x9F	ADDB2	Add byte

** Operating system instruction.
† RNEQU and RNEQ perform the same operation.
†† REQLU and REQL perform the same operation.

Table 26. Instruction Set Summary by Opcode (Continued)

Opcode	Mnemonic	Instruction
0xA0	PUSHW	Push word
0xA3	ADDPB2	Add packed BCD, 2-address
0xA4	MODW2	Modulo word
0xA6	MODH2	Modulo halfword
0xA7	MODB2	Modulo byte
0xA8	MULW2	Multiply word
0xAA	MULH2	Multiply halfword
0xAB	MULB2	Multiply byte
0xAC	DIVW2	Divide word
0xAE	DIVH2	Divide halfword
0xAF	DIVB2	Divide byte
0xB0	ORW2	OR word
0xB2	ORH2	OR halfword
0xB3	ORB2	OR byte
0xB4	XORW2	Exclusive OR word
0xB6	XORH2	Exclusive OR halfword
0xB7	XORB2	Exclusive OR byte
0xB8	ANDW2	AND word
0xBA	ANDH2	AND halfword
0xBB	ANDB2	AND byte
0xBC	SUBW2	Subtract word
0xBE	SUBH2	Subtract halfword
0xBF	SUBB2	Subtract byte
0xC0	ALSW3	Arithmetic left shift word
0xC4	ARSW3	Arithmetic right shift word
0xC6	ARSH3	Arithmetic right shift halfword
0xC7	ARSB3	Arithmetic right shift byte
0xC8	INSFW	Arithmetic left shift word
0xCA	INSFH	Insert field halfword
0xCB	INSFB	Insert field byte
0xCC	EXTFW	Extract field word
0xCE	EXTFH	Extract field halfword
0xCF	EXTFB	Extract field byte
0xD0	LLSW3	Logical left shiftword
0xD2	LLSH3	Logical left shift halfword
0xD3	LLSB3	Logical left shift byte
0xD4	LRSW3	Logical right shift word
0xD8	ROTW	Rotate word
0xDB	SUBPB3	Subtract packed BCD, 3-address
0xDC	ADDW3	Add word, 3-address
0xDE	ADDH3	Add halfword, 3-address
0xDF	ADDB3	Add byte, 3-address
0xE0	PUSHAW	Push address word
0xE3	ADDPB3	Add packed BCD, 3-address
0xE4	MODW3	Modulo word, 3-address
0xE6	MODH3	Modulo halfword, 3-address

Table 26. Instruction Set Summary by Opcode (Continued)

Opcode	Mnemonic	Instruction
0xE7	MODB3	Modulo byte, 3-address
0xE8	MULW3	Multiply word, 3-address
0xEA	MULH3	Multiply halfword, 3-address
0xEB	MULB3	Multiply byte, 3-address
0xEC	DIVW3	Divide word, 3-address
0xEE	DIVH3	Divide halfword, 3-address
0xEF	DIVB3	Divide byte, 3-address
0xF0	ORW3	OR word, 3-address
0xF2	ORH3	OR halfword, 3-address
0xF3	ORB3	OR byte, 3-address
0xF4	XORW3	Exclusive OR word, 3-address
0xF6	XORH3	Exclusive OR halfword, 3-address
0xF7	XORB3	Exclusive OR byte, 3-address
0xF8	ANDW3	AND word, 3-address
0xFA	ANDH3	AND halfword, 3-address
0xFB	ANDB3	AND byte, 3-address
0xFC	SUBW3	Subtract word, 3-address
0xFE	SUBH3	Subtract halfword, 3-address
0xFF	SUBB3	Subtract byte, 3-address

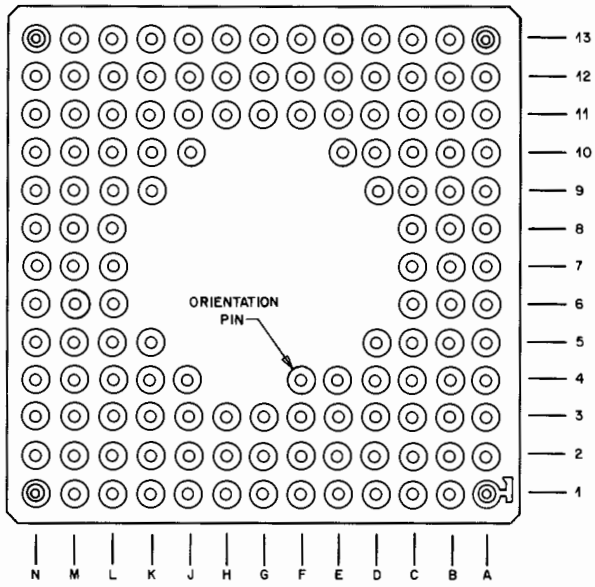
Pin Descriptions

The *WE* 32200 Microprocessor is available in a 133-pin square, ceramic PGA package. Figure 6 and Tables 27—35 describe the pin assignments.

The term *asserted* in the pin function descriptions means that the signal is driven to its active state either by the microprocessor (outputs) or an external device (inputs). The term *negated* means that the signal is driven to its inactive state.

A bar over a signal name (e.g., $\overline{AS}$) indicates that the signal is active low, logic 0. The 0 bit is the least significant bit for signals that occupy two or more pins (e.g., SAS0). In the signal type column, I indicates an input, O an output, and I/O a bidirectional signal.

Table 27 lists the signals in numerical order. Tables 28 through 35 describe the signals by functional group and list the pin numbers for the PGA package.



Bottom View

Figure 6. WE® 32200 Microprocessor 133-Pin Square, Ceramic PGA Package

Numerical Order

Table 27. Pin Descriptions – 133-Pin Square PGA Package

Pin	Name	Type	Description
A1	DATA23	I/O	Microprocessor data 23
A2	DATA24	I/O	Microprocessor data 24
A3	DATA26	I/O	Microprocessor data 26
A4	ADDR28	O	Microprocessor address 28
A5	IQS0	O	Instruction queue status 0
A6	ADDR30	O	Microprocessor address 30
A7	XMD1	O	Execution mode 1
A8	IQS1	O	Instruction queue status 1
A9	SAS3	O	Access status code 3
A10	XMD0	O	Execution mode 0
A11	SAS2	O	Access status code 2
A12	DSIZE1	O	Data size 1
A13	SAS1	O	Access status code 1
B1	DATA22	I/O	Microprocessor data 22
B2	DATA28	I/O	Microprocessor data 28
B3	DATA29	I/O	Microprocessor data 29
B4	DATA31	I/O	Microprocessor data 31
B5	ADDR29	O	Microprocessor address 29
B6	ADDR31	O	Microprocessor address 31
B7	$\overline{SOI}$	O	Start of instruction

Table 27. Pin Descriptions – 133-Pin Square PGA Package (Continued)

Pin	Name	Type	Description
B8	$\overline{\text{BRACK}}$	I/O	Bus request acknowledge
B9	$\overline{\text{BUSRQ}}$	I/O	Bus request
B10	SAS0	O	Access status code 0
B11	R/$\overline{\text{W}}$	O	Read/write
B12	DSIZE0	O	Data size 0
B13	$\overline{\text{RRREQ}}$	I	Relinquish and retry request
C1	ADDR19	O	Microprocessor address 19
C2	ADDR26	O	Microprocessor address 26
C3	DATA25	I/O	Microprocessor data 25
C4	ADDR27	O	Microprocessor address 27
C5	DATA30	I/O	Microprocessor data 30
C6	+5V	—	Power
C7	GND	—	Ground
C8	+5V	—	Power
C9	GND	—	Ground
C10	$\overline{\text{DONE}}$	I	Coprocessor done
C11	$\overline{\text{HIGHZ}}$	I	High impedance
C12	$\overline{\text{RRRACK}}$	O	Relinquish and retry request acknowledge
C13	$\overline{\text{FAULT}}$	I	Fault
D1	ADDR24	O	Microprocessor address 24
D2	ADDR20	O	Microprocessor address 20
D3	ADDR25	O	Microprocessor address 25
D4	DATA27	I/O	Microprocessor data 27
D5	GND	—	Ground
D9	DSIZE2	O	Data size 2
D10	$\overline{\text{SRDY}}$	I	Synchronous ready
D11	$\overline{\text{RETRY}}$	I	Retry
D12	$\overline{\text{BARB}}$	I	Bus arbiter
D13	$\overline{\text{RESET}}$	O	Reset acknowledge
E1	ADDR18	O	Microprocessor address 18
E2	ADDR23	O	Microprocessor address 23
E3	+5V	—	Power
E4	GND	—	Ground
E10	GND	—	Ground
E11	+5V	—	Power
E12	$\overline{\text{DTACK}}$	I	Data transfer acknowledge
E13	$\overline{\text{BLKFTCH}}$	I	Block (double-word) fetch
F1	ADDR21	O	Microprocessor address 21
F2	ADDR22	O	Microprocessor address 22
F3	GND	—	Ground
F4	GND	—	Ground
F11	CLK23	I	Input clock 23
F12	$\overline{\text{DSHAD}}$	I	Data bus shadow
F13	$\overline{\text{RESETR}}$	I	Reset request

Table 27. Pin Descriptions – 133-Pin Square PGA Package (Continued)

Pin	Name	Type	Description
G1	DATA21	I/O	Microprocessor data 21
G2	ADDR17	O	Microprocessor address 17
G3	DATA20	I/O	Microprocessor data 20
G11	CLK34	I	Input clock 34
G12	$\overline{\text{CYCLEI}}$	O	Cycle initiate
G13	$\overline{\text{STOP}}$	I	Microprocessor stop
H1	ADDR15	O	Microprocessor address 15
H2	DATA18	I/O	Microprocessor data 18
H3	GND	—	Ground
H11	GND	—	Ground
H12	$\overline{\text{AS}}$	O	Address strobe
H13	$\overline{\text{DS}}$	O	Data strobe
J1	DATA19	I/O	Microprocessor data 19
J2	DATA17	I/O	Microprocessor data 17
J3	+5V	—	Power
J4	GND	—	Ground
J10	GND	—	Ground
J11	+5V	—	Power
J12	$\overline{\text{ABORT}}$	O	Access abort
J13	$\overline{\text{DRDY}}$	O	Data ready
K1	ADDR16	O	Microprocessor address 16
K2	ADDR13	O	Microprocessor address 13
K3	DATA15	I/O	Microprocessor data 15
K4	DATA12	I/O	Microprocessor data 12
K5	GND	—	Ground
K9	GND	—	Ground
K10	DATA00	I/O	Microprocessor data 00
K11	GND	—	Microprocessor ground
K12	$\overline{\text{DYN16}}$	I	Dynamic 16-bit port
K13	$\overline{\text{NMINT}}$	I	Nonmaskable interrupt
L1	ADDR14	O	Microprocessor address 14
L2	DATA16	I/O	Microprocessor data 16
L3	DATA11	I/O	Microprocessor data 11
L4	ADDR11	O	Microprocessor address 11
L5	GND	—	Microprocessor ground
L6	ADDR08	O	Microprocessor address 08
L7	+5V	—	Power
L8	GND	—	Ground
L9	+5V	—	Power
L10	ADDR00	O	Microprocessor address 00
L11	$\overline{\text{INTOPT}}$	I	Interrupt option
L12	IPL3	I	Interrupt priority level 3
L13	$\overline{\text{AVEC}}$	I	Autovector
M1	ADDR12	O	Microprocessor address 12

Table 27. Pin Descriptions – 133-Pin Square PGA Package (Continued)

Pin	Name	Type	Description
M2	DATA14	I/O	Microprocessor data 14
M3	DATA13	I/O	Microprocessor data 13
M4	DATA08	I/O	Microprocessor data 08
M5	ADDR07	O	Microprocessor address 07
M6	DATA05	I/O	Microprocessor data 05
M7	ADDR03	O	Microprocessor address 03
M8	ADDR05	O	Microprocessor address 05
M9	DATA03	I/O	Microprocessor data 03
M10	DATA02	I/O	Microprocessor data 02
M11	DATA01	I/O	Microprocessor data 01
M12	IPL2	I	Interrupt priority level 2
M13	$\overline{\text{VAD}}$	O	Virtual address
N1	DATA10	I/O	Microprocessor data 10
N2	DATA09	I/O	Microprocessor data 09
N3	ADDR10	O	Microprocessor address 10
N4	ADDR09	O	Microprocessor address 09
N5	ADDR06	O	Microprocessor address 06
N6	DATA07	I/O	Microprocessor data 07
N7	ADDR02	O	Microprocessor address 02
N8	DATA06	I/O	Microprocessor data 06
N9	ADDR01	O	Microprocessor address 01
N10	ADDR04	O	Microprocessor address 04
N11	DATA04	I/O	Microprocessor data 04
N12	IPL0	I	Interrupt priority level 0
N13	IPL1	I	Interrupt priority level 1

Functional Groups

Table 28. Address and Data

Name	Pins	Type	Description
ADDR00—ADDR31	L10, N9, N7, M7, N10, M8, N5, M5, L6, N4, N3, L4, M1, K2, L1, H1, K1, G2, E1, C1, D2, F1, F2, E2, D1, D3, C2, C4, A4, B5, A6, B6	O	**Address.** These pins are used by the microprocessor to issue 32-bit addresses for off-chip accesses.
DATA00—DATA31	K10, M11, M10, M9, N11, M6, N8, N6, M4, N2, N1, L3, K4, M3, M2, K3, L2, J2, H2, J1, G3, G1, B1, A1, A2, C3, A3, D4, B2, B3, C5, B4	I/O	**Data.** These bidirectional pins are used to convey data to and from the microprocessor.

Table 29. Interface and Control Signals

Name	Pin	Type	Description
$\overline{AS}$	H12	O	**Address Strobe.** When low, this signal indicates the presence of a valid physical address on the address pins. If the address is virtual, the falling edge of $\overline{AS}$ indicates a valid address, and the address pins are 3-stated subsequent to the falling edge of $\overline{AS}$.
$\overline{CYCLEI}$	G12	O	**Cycle Initiate.** This signal is asserted at the beginning of a bus transaction and negated two clock cycles later. $\overline{CYCLEI}$ is asserted in both the read and write halves of an interlocked-read transaction.
$\overline{DONE}$	C10	I	**Coprocessor Done.** This input is recognized during a coprocessor instruction. It informs the microprocessor that a coprocessor has completed its operation.
$\overline{DRDY}$	J13	O	**Data Ready.** When asserted, this signal indicates the end of a bus transaction that has no bus exceptions. ($\overline{FAULT}$, $\overline{RETRY}$, $\overline{RRREQ}$ signals).
$\overline{DS}$	H13	O	**Data Strobe.** During a read operation this signal, when low, indicates that a slave device can place data on the data bus. During a write operation this signal, when low, indicates that the *WE* 32200 Microprocessor has placed valid data on the data bus.
$\overline{DTACK}$	E12	I	**Data Transfer Acknowledge.** This signal is used to handshake between the *WE* 32200 Microprocessor and a slave device. During a read operation, the microprocessor latches data present on the data bus and terminates the bus transaction one cycle after $\overline{DTACK}$ is driven low by a slave device. During a write operation the transaction is terminated one cycle after a slave device drives $\overline{DTACK}$ low. If $\overline{DTACK}$ is high, wait states are inserted in the current cycle. $\overline{DTACK}$ is ignored if the data bus shadow signal ($\overline{DSHAD}$) is asserted. The $\overline{DTACK}$ is an asynchronous input and is double-latched to avoid metastability.
$\overline{DYN16}$	K12	I	**Dynamic 16-Bit Port.** This signal informs the CPU that the port that it is writing to or reading from is a 16-bit port. The memory system has 4 different options when asserting $\overline{DYN16}$: • **$\overline{DYN16}$ asserted with $\overline{DTACK}$.** The $\overline{DYN16}$ pin can be qualified with the asynchronous acknowledge to indicate a 16-bit port. The signal is internally double-latched. Assertion of $\overline{DYN16}$ looks identical to the assertion of $\overline{DTACK}$. • **$\overline{DYN16}$ asserted after $\overline{DTACK}$.** This is very similar to case 1 above, except that $\overline{DYN16}$ may be asserted at the synchronous sampling point.

Table 29. Interface and Control Signals (Continued)

Name	Pin	Type	Description
			• **$\overline{\text{DYN16}}$ by itself.** The $\overline{\text{DYN16}}$ can be used as an acknowledge by itself (i.e. $\overline{\text{DYN16}}$ can terminate a memory transaction). The signal is internally double-latched. $\overline{\text{DYN16}}$ has to be asserted at the asynchronous sampling point and looks exactly like a $\overline{\text{DTACK}}$ assertion. • **$\overline{\text{DYN16}}$ asserted with $\overline{\text{SRDY}}$.** The $\overline{\text{DYN16}}$ signal can be qualified with the synchronous acknowledge to indicate a 16-bit port. The assertion of $\overline{\text{DYN16}}$ looks exactly like the assertion of $\overline{\text{SRDY}}$. $\overline{\text{DYN16}}$ is ignored if $\overline{\text{DSHAD}}$ is asserted.
$\overline{\text{SRDY}}$	D10	I	**Synchronous Ready.** This signal is a synchronous input that begins the termination of a read or write operation when asserted. It is sampled only once on the falling edge of the fourth clock state during read and write operations. If $\overline{\text{SRDY}}$ is not asserted at this time and $\overline{\text{DTACK}}$ was not asserted during the previous clock state, then wait state cycles are inserted until either signal is asserted. $\overline{\text{SRDY}}$ is ignored if the data bus shadow input ($\overline{\text{DSHAD}}$) was previously asserted.

Table 30. Access Status Signals

Name	Pin(s)	Type	Description
$\overline{\text{BLKFTCH}}$	E13	I	**Block (Double-Word) Fetch.** This input indicates to the microprocessor that the memory system can perform a double-word (8-byte) program block fetch. On all instruction fetches, the data size (DSIZE0—DSIZE2) pins show a double-word access. If the memory system can handle a double word access, it activates this input. Otherwise, the input is left inactive, and the microprocessor fetches a block of instruction by two consecutive reads.
DSIZE0—DSIZE2	B12, A12, D9	O	**Data Size.** This three-bit output is used to indicate whether the microprocessor is transferring 0 bytes, 1 byte, halfword, 3 bytes, word, or double-word data in the current bus transaction. On all instruction fetches the DSIZE0—DSIZE2 pins have the value for double-word:

DSIZE2	DSIZE1	DSIZE0	Description
0	0	0	3 byte transaction
0	0	1	0 byte transaction
0	1	0	Reserved
0	1	1	Reserved
1	0	0	Word transaction
1	0	1	Double-word transaction
1	1	0	Halfword transaction
1	1	1	Byte transaction

Table 30. Access Status Signals (Continued)

<table>
<tr><th>Name</th><th>Pin(s)</th><th>Type</th><th>Description</th></tr>
<tr><td>R/$\overline{W}$</td><td>B11</td><td>O</td><td>Read/Write. This signal indicates whether the bus transaction is a read or a write. When low, the operation is a write. When high, the operation is a read. This pin is valid during the time the address strobe ($\overline{AS}$) is active.</td></tr>
<tr><td>SAS0—
SAS3</td><td>B10, A13,
A11, A9</td><td>O</td><td>Access Status Codes. These pins describe the type of bus transaction being executed:
<table>
<tr><th>SAS3</th><th>SAS2</th><th>SAS1</th><th>SAS0</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>0</td><td>0</td><td>Move translated word</td></tr>
<tr><td>0</td><td>0</td><td>0</td><td>1</td><td>Coprocessor data write</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>0</td><td>Autovector interrupt acknowledge</td></tr>
<tr><td>0</td><td>0</td><td>1</td><td>1</td><td>Coprocessor data fetch</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>0</td><td>Stop acknowledge</td></tr>
<tr><td>0</td><td>1</td><td>0</td><td>1</td><td>Coprocessor broadcast</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>0</td><td>Coprocessor status fetch</td></tr>
<tr><td>0</td><td>1</td><td>1</td><td>1</td><td>Read interlocked</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>0</td><td>Address fetch</td></tr>
<tr><td>1</td><td>0</td><td>0</td><td>1</td><td>Operand fetch</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>0</td><td>Write</td></tr>
<tr><td>1</td><td>0</td><td>1</td><td>1</td><td>Interrupt acknowledge</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>0</td><td>Instruction fetch after PC discontinuity</td></tr>
<tr><td>1</td><td>1</td><td>0</td><td>1</td><td>Instruction prefetch</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>0</td><td>Instruction fetch</td></tr>
<tr><td>1</td><td>1</td><td>1</td><td>1</td><td>No operation</td></tr>
</table></td></tr>
<tr><td>$\overline{VAD}$</td><td>M13</td><td>O</td><td>Virtual Address. When low, this signal indicates that the address is virtual. When it is high, the address is a physical address. $\overline{VAD}$ is a level signal. It is asserted by execution of the enable virtual pin and jump (ENBVJMP) instruction and negated by execution of the disable virtual pin and jump (DISVJMP) instruction.</td></tr>
<tr><td>XMD0,
XMD1</td><td>A10, A7</td><td>O</td><td>Execution Mode. These two outputs indicate the present execution mode of the microprocessor:
<table>
<tr><th>XMD1</th><th>XMD0</th><th>Description</th></tr>
<tr><td>0</td><td>0</td><td>Kernel mode</td></tr>
<tr><td>0</td><td>1</td><td>Executive mode</td></tr>
<tr><td>1</td><td>0</td><td>Supervisor mode</td></tr>
<tr><td>1</td><td>1</td><td>User mode</td></tr>
</table>
If an MMU is present in the system, it may latch and use a spurious execution mode value if XMD0 or XMD1 changes during an access. Since XMD0 and XMD1 reflect the state of the current execution level (CM) bits in the PSW, changes to the CM field via nonmicrosequence instructions must be avoided.</td></tr>
</table>

Table 31. Interrupt Signals

Name	Pin(s)	Type	Description
$\overline{\text{AVEC}}$	L13	I	**Autovector.** When this input is asserted with the interrupt priority level input, the microprocessor supplies its own vector. The vector number value is the inverted interrupt option ($\overline{\text{INTOPT}}$) input concatenated with the interrupt priority level value. When autovector is not asserted, the interrupting device supplies the vector (see Interrupts).
$\overline{\text{INTOPT}}$	L11	I	**Interrupt Option.** This asynchronous input is latched, along with the interrupt priority level inputs (IPL0—IPL3), inverted, and output on ADDR06 during an interrupt acknowledge transaction.
IPL0—IPL3	N12, N13, M12, L12	I	**Interrupt Priority Level.** These asynchronous inputs indicate the level of the pending interrupt. The code is based on a decreasing priority scheme, with 0000 having the highest priority and 1110 the lowest. Level 1111 indicates that no interrupts are pending. IPL0 is the least significant bit of the interrupt priority level code. To be acknowledged, the requesting level on the pins inverted must be greater than the present interrupt priority level (IPL field) in the PSW. The exception to this is a nonmaskable interrupt, which can interrupt the microprocessor regardless of the present IPL field priority level.
$\overline{\text{NMINT}}$	K13	I	**Nonmaskable Interrupt.** When asserted, this asynchronous input indicates that a nonmaskable interrupt is being requested. The microprocessor acknowledges this interrupt with an autovector interrupt acknowledge cycle (see Interrupts). During this acknowledge cycle, the microprocessor address bus contains all zeros.

Table 32. Arbitration Signals

Name	Pin	Type	Description
$\overline{\text{BARB}}$	D12	I	**Bus Arbiter.** When this input is strapped low, the microprocessor is the arbiter of the bus. As arbiter, the microprocessor need not request the bus to obtain access to the bus. When the pin is strapped high, the microprocessor is not the arbiter and must request the bus to use it. The following outputs are 3-stated when the microprocessor is not the bus arbiter until the CPU does a bus transaction:

$\overline{\text{ABORT}}$	$\overline{\text{DS}}$
ADDR00—ADDR31	DSIZE0—DSIZE2
$\overline{\text{AS}}$	R/$\overline{\text{W}}$
$\overline{\text{CYCLEI}}$	SAS0—SAS3
DATA00—DATA31	$\overline{\text{VAD}}$
$\overline{\text{DRDY}}$	XMD0, XMD1

Table 32. Arbitration Signals (Continued)

Name	Pin	Type	Description
$\overline{\text{BRACK}}$	B8	I/O	**Bus Request Acknowledge.** This signal is an output if the microprocessor is the arbiter of the bus and an input if it is not. As an output, this pin indicates that the bus request ($\overline{\text{BUSRQ}}$) has been recognized and that the microprocessor has 3-stated the bus for the requesting bus master. The bus signals that are 3-stated when the $\overline{\text{BRACK}}$ is issued are: $\overline{\text{ABORT}}$ — $\overline{\text{DS}}$ ADDR00—ADDR31 — DSIZE0—DSIZE2 DATA00—DATA31 — R/$\overline{\text{W}}$ $\overline{\text{AS}}$ — SAS0—SAS3 $\overline{\text{CYCLEI}}$ — $\overline{\text{VAD}}$ $\overline{\text{DRDY}}$ — XMD0, XMD1 As an input, when $\overline{\text{BRACK}}$ is asserted, this pin indicates that the microprocessor's bus request has been recognized and that the microprocessor may take possession of the bus.
$\overline{\text{BUSRQ}}$	B9	I/O	**Bus Request.** This asynchronous signal is an input if the microprocessor is the arbiter of the bus and an output if it is not. As an input, this signal indicates that an external device is requesting the bus. As an output, the signal indicates that the microprocessor is requesting the bus.

Table 33. Bus Exception Signals

Name	Pin	Type	Description
$\overline{\text{ABORT}}$	J12	O	**Access Abort.** This pin is asserted on an access that is to be ignored by the memory system. The two cases for the CPU abort of a memory access are an instruction cache hit on program counter discontinuity and on an alignment fault.
$\overline{\text{DSHAD}}$	F12	I	**Data Bus Shadow.** The DATA00—DATA31, $\overline{\text{DRDY}}$, DSIZE0—DSIZE2, and R/$\overline{\text{W}}$ pins are 3-stated when this input is asserted. This input is used by the memory management unit (MMU) to remove the microprocessor from the data bus.
$\overline{\text{FAULT}}$	C13	I	**Fault.** This input notifies the microprocessor that a fault condition has occurred. It is an asynchronous input and is double latched prior to the assertion of $\overline{\text{DTACK}}$, and synchronous after the assertion of $\overline{\text{DTACK}}$ (latched once). The $\overline{\text{FAULT}}$ signal is ignored if $\overline{\text{DSHAD}}$ is asserted.
$\overline{\text{RESET}}$	D13	O	**Reset Acknowledge.** This signal indicates that the microprocessor has recognized an external reset request or that it has generated an internal reset (i.e., reset exception). The microprocessor executes its reset routine once it negates $\overline{\text{RESET}}$.

Table 33. Bus Exception Signals (Continued)

Name	Pin	Type	Description
$\overline{\text{RESETR}}$	F13	I	**Reset Request.** This asynchronous signal is used to reset the microprocessor. $\overline{\text{RESETR}}$ is sampled every clock cycle and must be asserted for 2 consecutive clock cycles in order to be acknowledged. The microprocessor acknowledges the request by immediately asserting $\overline{\text{RESET}}$. (See Reset.)
$\overline{\text{RETRY}}$	D11	I	**Retry.** When this signal is asserted, the microprocessor terminates the current bus transaction and retries it when $\overline{\text{RETRY}}$ is negated.
$\overline{\text{RRRACK}}$	C12	O	**Relinquish and Retry Request Acknowledge.** This signal is asserted in response to a relinquish and retry bus exception when the microprocessor has relinquished the bus (3-stated the bus). This open-drain output is passively pulled high by an external pull-up resistor when the bus transaction terminated by the relinquish and retry bus exception is retried.
$\overline{\text{RRREQ}}$	B13	I	**Relinquish and Retry Request.** This signal is used to preempt a bus transaction, so that the microprocessor bus may be used. The signal causes the microprocessor to terminate the current bus transaction and to 3-state the following pins: $\overline{\text{ABORT}}$ — $\overline{\text{DS}}$ ADDR00—ADDR31 — DSIZE0—DSIZE2 $\overline{\text{AS}}$ — SAS0—SAS3 $\overline{\text{CYCLEI}}$ — $\overline{\text{VAD}}$ DATA00—DATA31 — R/$\overline{\text{W}}$ $\overline{\text{DRDY}}$ — XMD0, XMD1 The $\overline{\text{RRRACK}}$ signal is asserted after all the above-mentioned pins are 3-stated. During this 3-state phase, the bus master requesting the relinquish and retry may take possession of the bus. No external bus arbitration signals are acknowledged during the assertion of relinquish and retry request. When $\overline{\text{RRREQ}}$ is negated, the preempted bus transaction is retried.
$\overline{\text{STOP}}$	G13	I	**Stop.** When asserted, this asynchronous signal halts the execution of any further instructions or microsequences beyond those already started. At most, one more instruction or one microsequence may be executed beyond the instruction during which $\overline{\text{STOP}}$ was asserted before the microprocessor comes to halt.

Table 34. Development Support Signals

Name	Pin(s)	Type	Description
$\overline{\text{HIGHZ}}$	C11	I	**High Impedance.** This signal puts all output pins on the microprocessor into the high-impedance state when asserted. This pin is intended for testing purposes.
IQS0, IQS1	A5, A8	O	**Instruction Queue Status.** This two-bit code indicates the activity on the microprocessor instruction queue: IQS1 IQS0 Description 0 0 Discard 4 bytes 0 1 Discard 1 byte 1 0 Discard 2 bytes 1 1 No discard this cycle
$\overline{\text{SOI}}$	B7	O	**Start of Instruction.** When asserted, this signal indicates that the microprocessor's internal control has fetched the opcode for the next instruction from the internal instruction queue. Since the instructions are pipelined, it does not always mean the end of the previous instruction execution.

Table 35. Clock Signals

Name	Pin	Type	Description
CLK34	G11	I	**Input Clock.** The falling edge of this clock signifies the beginning of a machine cycle. This clock input has same frequency as CLK23 and lags it by 90°.
CLK23	F11	I	**Input Clock.** This clock input has the same frequency as CLK34 and leads it by 90°.

Characteristics

V_{CC} = 5.0 V ± 5%, V_{SS} = 0 V, C_L = 130 pF, T_A = 0 to 70 °C

Timing Characteristics

All TTL timing specifications are referenced to and from 0.8 V for a low voltage and 2.0 V for a high voltage. CMOS clock references are to and from $V_{CC}/2$. All timing information is subject to change. All min and max values are in ns.

Table 36. Timing Characteristics

Num	Symbol	Description	Figure(s)	24 MHz	
				Min	Max
1	tDATVC34H	Data set-up time	8	3	—
2A	tDSBHDATX	Data hold time	8	0	—
2B	tC34HDATX	Data hold time	8	3	—
3	tSRYLC34L	Synchronous ready set-up time	8	3	—
4	tC34LSRYH	Synchronous ready hold time	8	3	—
5	tDTALC34H	Data transfer acknowledge set-up time*	8	3	—
6	tDSBHDTAH	Data transfer acknowledge hold time	8	0	—
7	tDSHLG01Z	Outputs 3-stated time	—	—	29
8	tDSHHG01V	Outputs valid time	—	—	29
9	tIPLVC23L	Interrupt priority level set-up time*	8	3	—
10	tIPLVSASV	Interrupt priority level valid time	—	0	—
11	tIOPLC23L	Interrupt option set-up time*	8	3	—
12	tIOPLSASV	Interrupt option assertion time	—	0	—
13	tNMILC23L	Nonmaskable interrupt set-up time*	8	3	—
14	tNMILSASV	Nonmaskable interrupt assertion time	—	0	—
15	tBFTLC34L	Block (double-word) fetch set-up time*	11	3	—
16A	tDSBHBFTH	Block (double-word) fetch asserted time**	11	0	—
16B	tC34LBFTH	Block (double-word) fetch hold time	11	3	—
17	tRTYLC34L	Synchronous retry set-up time	10	3	—
18A	tDSBHRTYH	Synchronous retry hold time**	18	0	—
18B	tC34LRTYH	Synchronous retry hold time	10	3	—
19	tRRRLC34L	Synchronous relinquish and retry request set-up time	10	3	—
20A	tRRALRRRH	Delayed relinquish and retry request hold time*	19	0	—
20B	tC34LRRRH	Synchronous relinquish and retry request hold time	10	3	—

* These are the asynchronous signals; set-up times are specified for testing and information purposes. The set-up times guarantee recognition at the next clock edge.

** Required only in systems using the *WE* 32201 Memory Management Unit.

Table 36. Timing Characteristics (Continued)

Num	Symbol	Description	Figure(s)	24 MHz	
				Min	Max
21	tFATLC34L	Synchronous fault set-up time	10	3	—
22A	tDSBHFATH	Delayed fault hold time*	20	0	—
22B	tC34LFATH	Synchronous fault hold time	10	3	—
23	tBRQLC23H	Bus request set-up time**	14	3	—
24	tBRALBRQH	Bus request hold time	14	0	—
25	tRSRLC23L	Reset request set-up time**	—	3	—
26	tC23LRSRH	Reset request valid time to guarantee reset	14	3Tc	—
27	tRTYLC34H	Asynchronous retry set-up time**	21	3	—
28	tRRRLC34H	Asynchronous relinquish and retry request set-up time**	21	3	—
29	tFATLC34H	Asynchronous fault set-up time**	21	3	—
30	tSTPLC23L	CPU stop set-up time**	13	3	—
31	tSASVSTPH	CPU stop valid time	13	0	—
32	tAVCLC23L	Autovector set-up time**	8	3	—
33	tAVCVSASV	Autovector valid time	—	0	—
34	tBRALC23H	Bus request acknowledge set-up time**,†	—	3	—
35	tBRALBRQH	Bus request acknowledge hold time†	—	0	—
36	tDONLC23H	Slave processor done set-up time	17	3	—
37	tSASVDONH	Slave processor done hold time	17	0	—
38	tDSBHRTYH	Asynchronous retry hold time	21	0	—
39	tDSBHRRRH	Asynchronous relinquish and retry request hold time	21	0	—
40	tDSBHFATH	Asynchronous fault hold time	21	0	—
41	tC34LADDV	Address assertion time	8	—	19
42	tC23HADDZ	Virtual address 3-state time	8	—	20
43	tC34LADDZ	Address 3-state time	14	—	20
44	tC34LDATV	Data assertion time	9	—	29
45	tC34LDATZ	Data 3-state time	14	—	20
46	tC34LSASV	Access status code assertion time	8	—	30
46A	tBRQHSASV	Access status code driven time	14	0.75Tc	—

* Required only in systems using the *WE* 32200 Memory Management Unit.
** These are asynchronous signals: set-up times are specified for testing and information purposes. The set-up times guarantee recognition at the next clock edge.
† These specifications are valid when the CPU is the slave ($\overline{BARB}$ = 1).

Table 36. Timing Characteristics (Continued)

Num	Symbol	Description	Figure(s)	24 MHz	
				Min	Max
47	tC34LSASZ	Access status code 3-state time	14	—	20
48	tC34LDSZV	Data size assertion time	8	—	26
49	tC34LDSZZ	Data size 3-state time	14	—	20
50	tC34HDRYL	Data ready assertion time	8	—	24
51	tC34HDRYH	Data ready negation time	8	—	24
52A	tC34HASBL	Address strobe assertion time	8	—	24
52B	tADDVASBL	Address set-up time	8	22	—
52C	tASBLADDV	Virtual address hold time	8	26	—
53A	tC34HASBH	Address strobe negation time	8	—	24
53B	tASBHADDX	Address hold time	8	17	—
54A	tC34HDSBL	Data strobe assertion time	8,9	—	24
54B	tDATVDSBL	Data set-up time	9	17	—
54C	tDSBLDSBH	Data strobe assertion (width) time	9	0.85Tc	—
55	tC34HDSBH	Data strobe negation time	8	—	24
55B	tDSBHDATZ	Data hold time	9	15	—
55C	tDSBHASBL	Address strobe negation time	—	-4	—
55D	tDSBHDSBL	Data strobe negated between first and second fetches time	11	0.9Tc	—
58	tC34LCYCL	Cycle initiate assertion time	8	—	22
57	tC34LCYCH	Cycle initiate negation time	8	—	22
60	tC34LR/WL	Read/write assertion time	9	—	24
61	tR/WLASBL	Read/write set-up time	9	17	—
61A	tR/WLDSBL	Read/write set-up time	9	13	—
62	tC34LBRAL	Bus request acknowledge assertion time	14	—	26
63	tC34LBRAH	Bus request acknowledge negation time	14	—	26
64	tC34LASBZ	Address strobe 3-state time	14	—	20
65	tC34LDSBZ	Data strobe 3-state time	14	—	20
66	tC34LR/WZ	Read/write 3-state time	14	—	20
67	tC34LDRYZ	Data ready 3-state time	14	—	20
68	tC34LCYCZ	Cycle initiate 3-state time	14	—	20
69	tC34LVADL	Virtual address assertion time	—	—	25
70	tC34HVADH	Virtual address negation time	—	—	26
71	tC34HABTL	Access abort assertion time	15	—	24

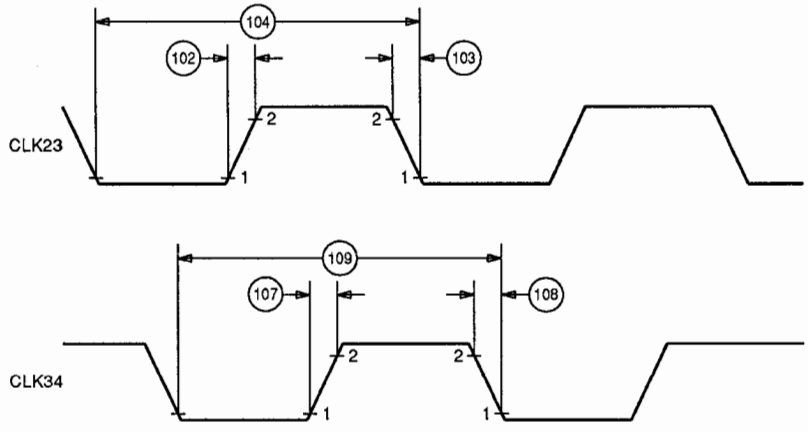
Table 36. Timing Characteristics (Continued)

Num	Symbol	Description	Figure(s)	24 MHz	
				Min	Max
72	tC34HABTH	Access abort negation time	15	—	24
73	tC34LSOIL	Start of instruction time assertion	12	—	25
74	tC34LSOIH	Start of instruction negation time	12	—	26
75	tC34LIQSV	Instruction queue status assertion time	12	—	26
76	tC34LRSTL	Reset acknowledge assertion time	16	—	28
77	tC34LRSTH	Reset acknowledge negation time	16	—	20
78	tC34LXMDV	Execution mode assertion time	14	—	26
79	tC34LXMDZ	Execution mode 3-state time	14	—	20
80	tC34LRRAL	Relinquish and retry request acknowledge assertion time	19	—	28
81	tC34LRRAH	Relinquish and retry request acknowledge assertion time	19	—	30
82	tC34HBRQL	Bus request assertion time*	—	—	20
83	tC34HBRQH	Bus request negation time*	—	—	29
84	tHIGZOUTZ	All outputs to 3-state time	—	—	41
90	tDTALRTYL	Delayed retry time	18	—	Tc/2
91	tDTALRRRL	Delayed relinquish and retry request time	19	—	Tc/2
92	tDTALFATL	Delayed fault time	20	—	Tc/2
93	tDTALBFTL	Delayed blockfetch time	—	—	Tc/2
94	tDYNVC34L	Synchronous $\overline{\text{DYN16}}$ set-up time	23	3	—
95	tDSBHDYNX	Synchronous $\overline{\text{DYN16}}$ hold time**	24	0	—
96	tC34LDYNH	Synchronous $\overline{\text{DYN16}}$ hold time	23	I3	—
97	tADDVDYNH	Asynchronous $\overline{\text{DYN16}}$ hold time	22	0	—
98	tDTAVDYNL	Delayed $\overline{\text{DYN16}}$ set-up time	24	—	Tc/2
99	tDYNVC34H	Asynchronous $\overline{\text{DYN16}}$ set-up time	22	3	—
100	tC23J	Clock 23 jitter	—	—	0.5
101	tC23E	Clock 23 duty cycle error	—	—	2
102	tC23L1C23H2	Clock 23 rise time	7	—	4
103	tC23H2C23L1	Clock 23 fall time	7	—	4
104	tC23L1C23L1	Clock 23 period (T)	7	41	—
105	tC34J	Clock 34 jitter	—	—	0.5
106	tC34E	Clock 34 duty cycle error	—	—	2
107	tC34L1C34H2	Clock 34 rise time	7	—	4
108	tC34H2C34L1	Clock 34 fall time	7	—	4
109	tC34L1C34L1	Clock 34 period (T)	7	41	—
110	tSKEW	Clock skew	—	—	2

* These specifications are valid when the CPU is the slave ($\overline{\text{BARB}}$ = 1).
** Required only in systems using the *WE* 32201 Memory Management Unit.

Timing Diagrams

These timing diagrams represent a subset of all possible transactions and are intended only for the purpose of displaying and clarifying timing relationships.

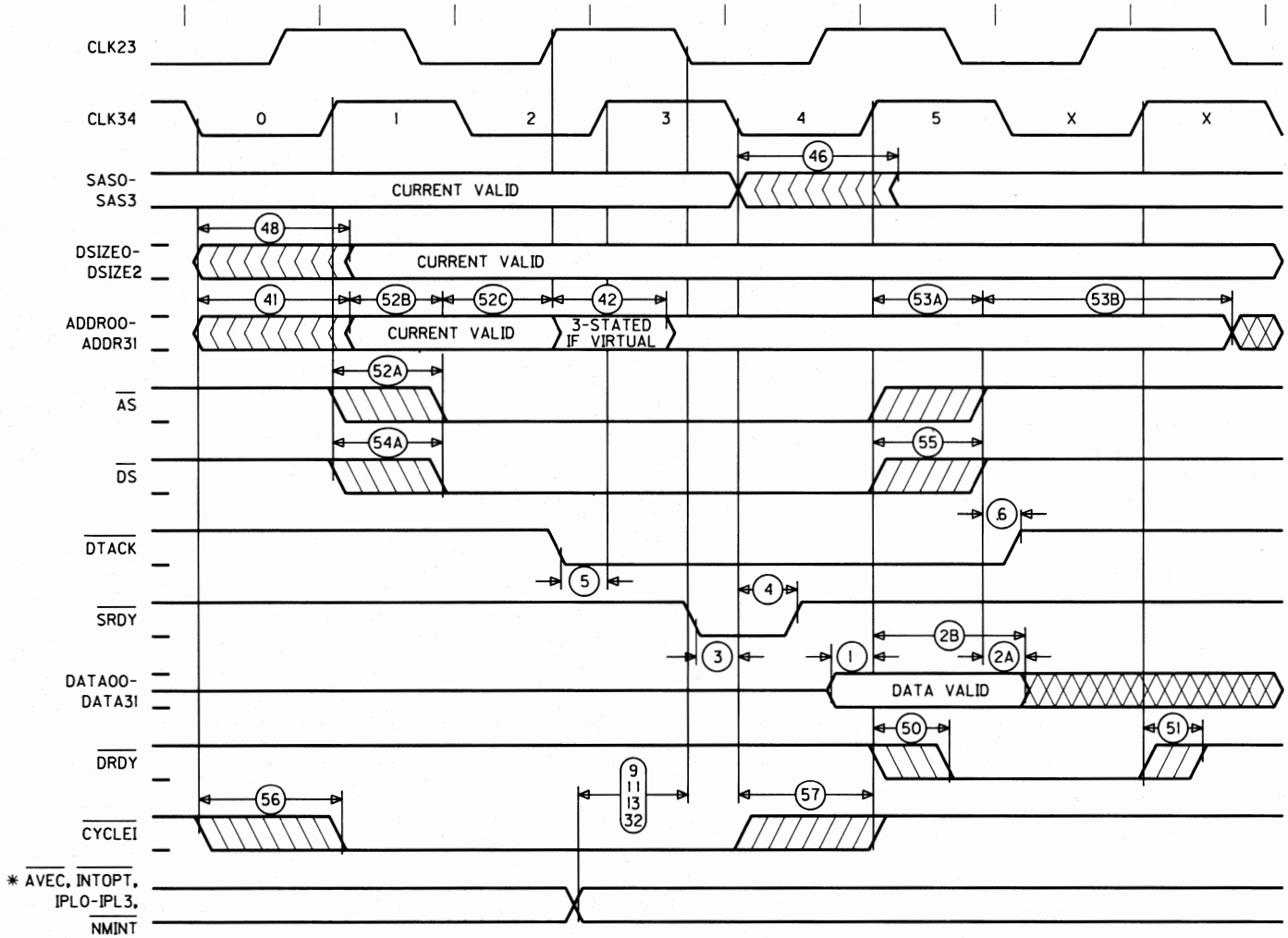


Notes:
Duty Cycle Error – The duty cycle of each clock input may deviate from 50% but should not exceed timing specification numbers 101 and 106 for CLK23 and CLK34, respectively.

Skew – CLK23 nominally leads CLK34 by 90° (1/4 clock period). This phase lead should never exceed timing specification 110.

Jitter – The period of each clock input may deviate from its nominal value but should not exceed timing specification numbers 100 or 105 for CLK23 and CLK34, respectively.

Figure 7. Clock Inputs



* These signals must be held valid until the processor acknowledges their reception via the SAS codes assigned for this purpose.

Figure 8. Read Timing

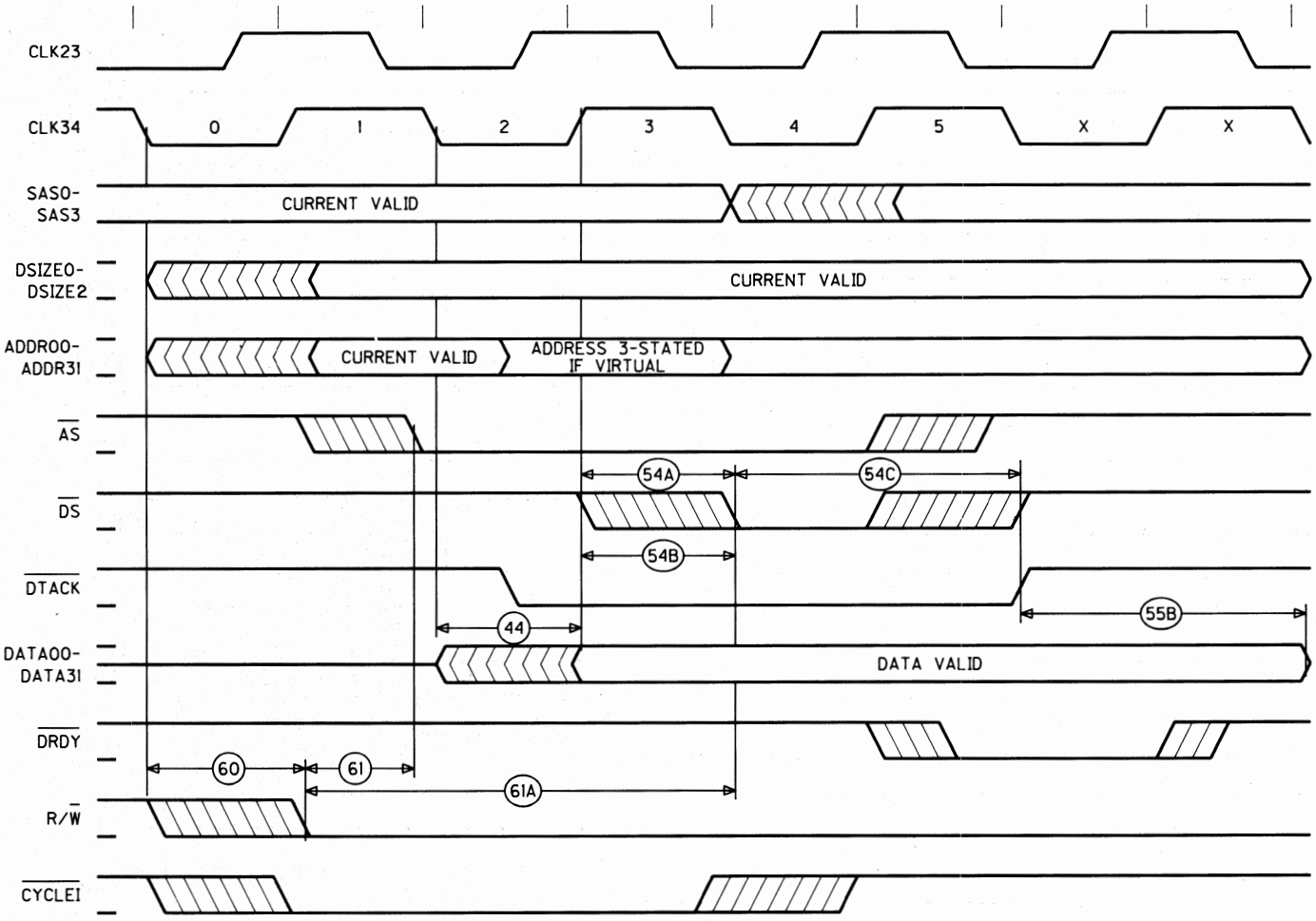


Figure 9. Write Timing

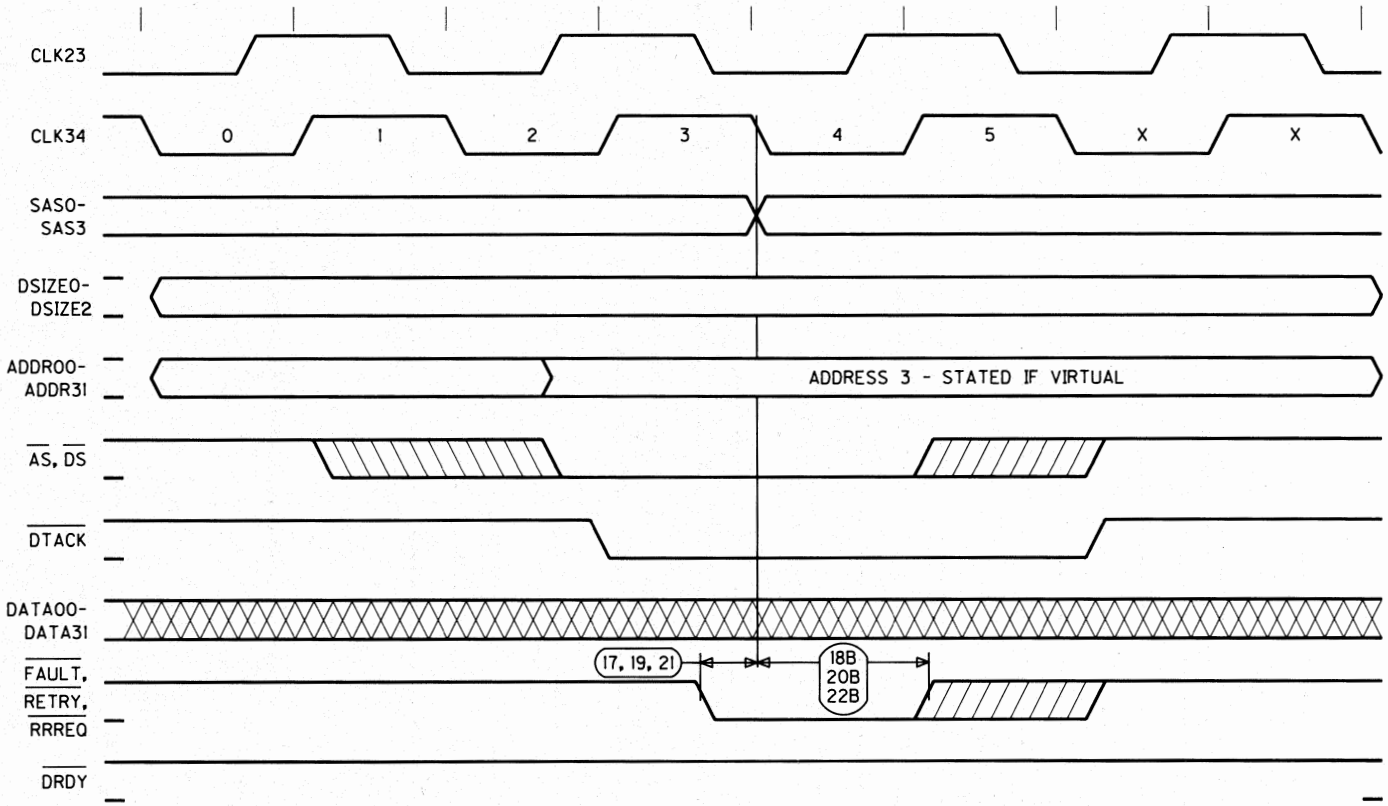


Figure 10. Bus Exceptions ($\overline{\text{FAULT}}$, $\overline{\text{RETRY}}$, $\overline{\text{RREQ}}$) Timing

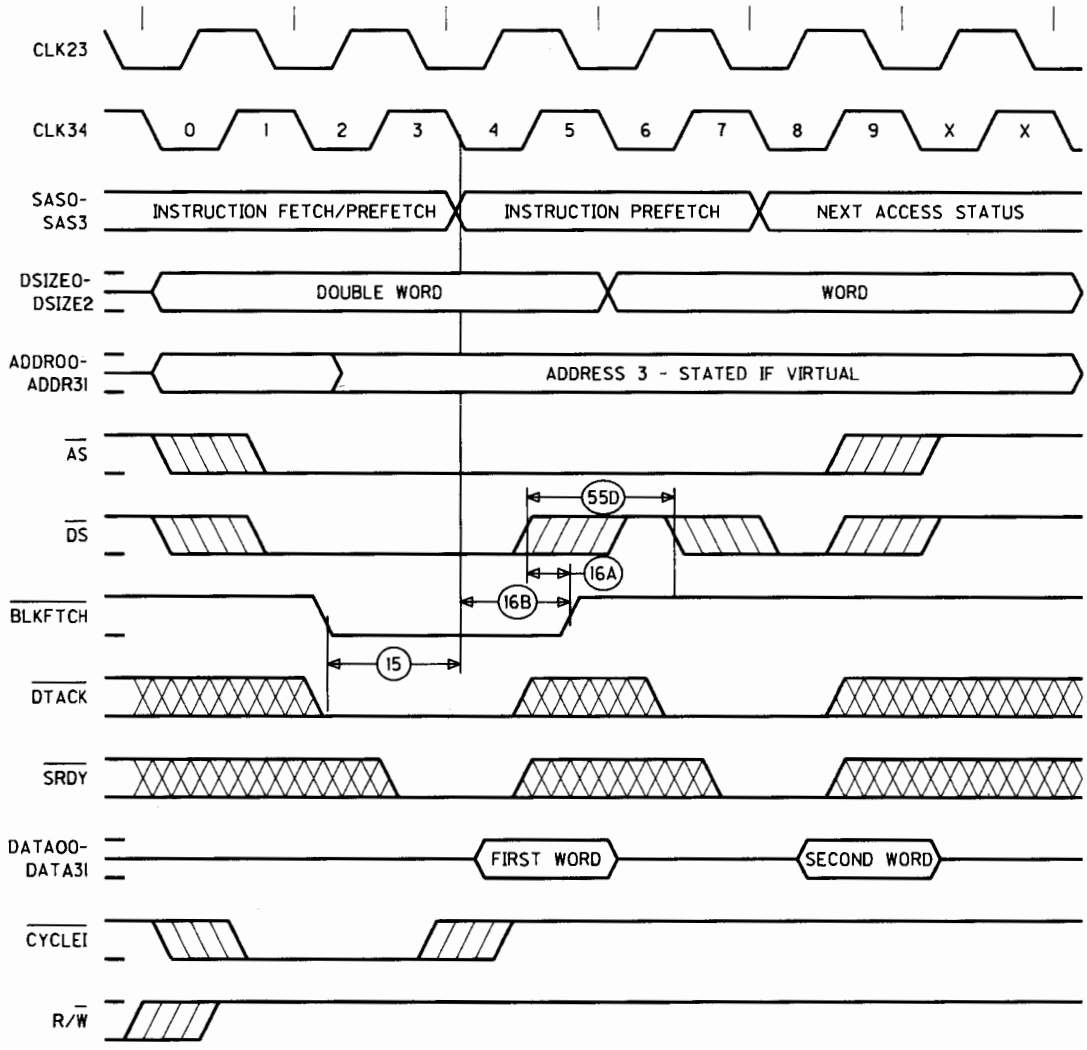


Figure 11. Block (Double-Word) Fetch Timing

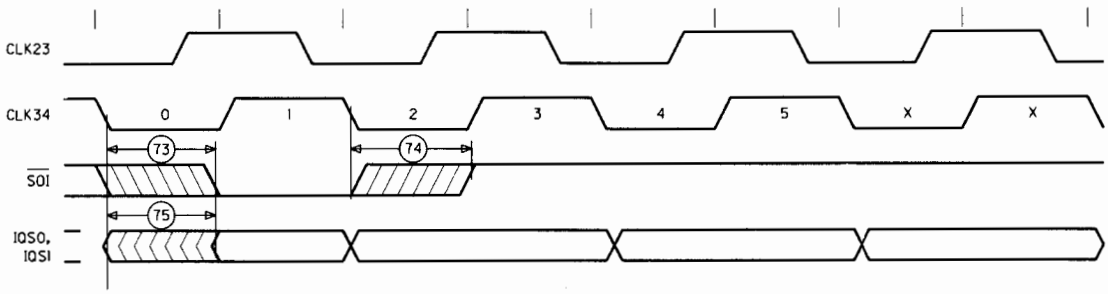
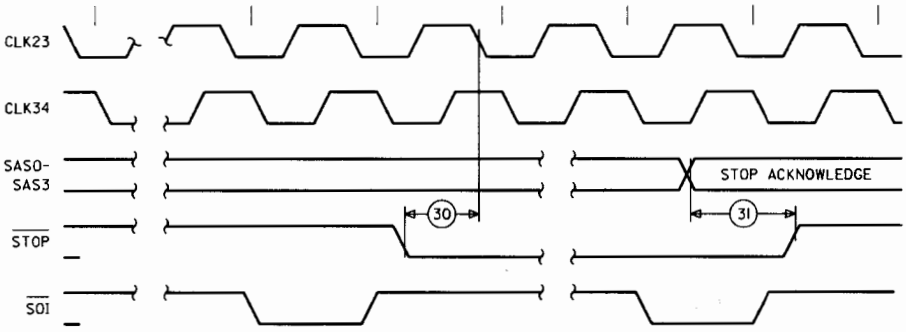


Figure 12. Start of Instruction and Instruction Queue Status Timing



Notes:
There will be at most one assertion of $\overline{SOI}$ before $\overline{STOP}$ is acknowledged on SAS0—SAS3 pins.
$\overline{STOP}$ must be asserted with the $\overline{STOP}$ acknowledge status valid.
$\overline{STOP}$ must be inactive for at least one cycle for the CPU to proceed to the next access.

Figure 13. Assertion of $\overline{STOP}$ after $\overline{SOI}$ Timing

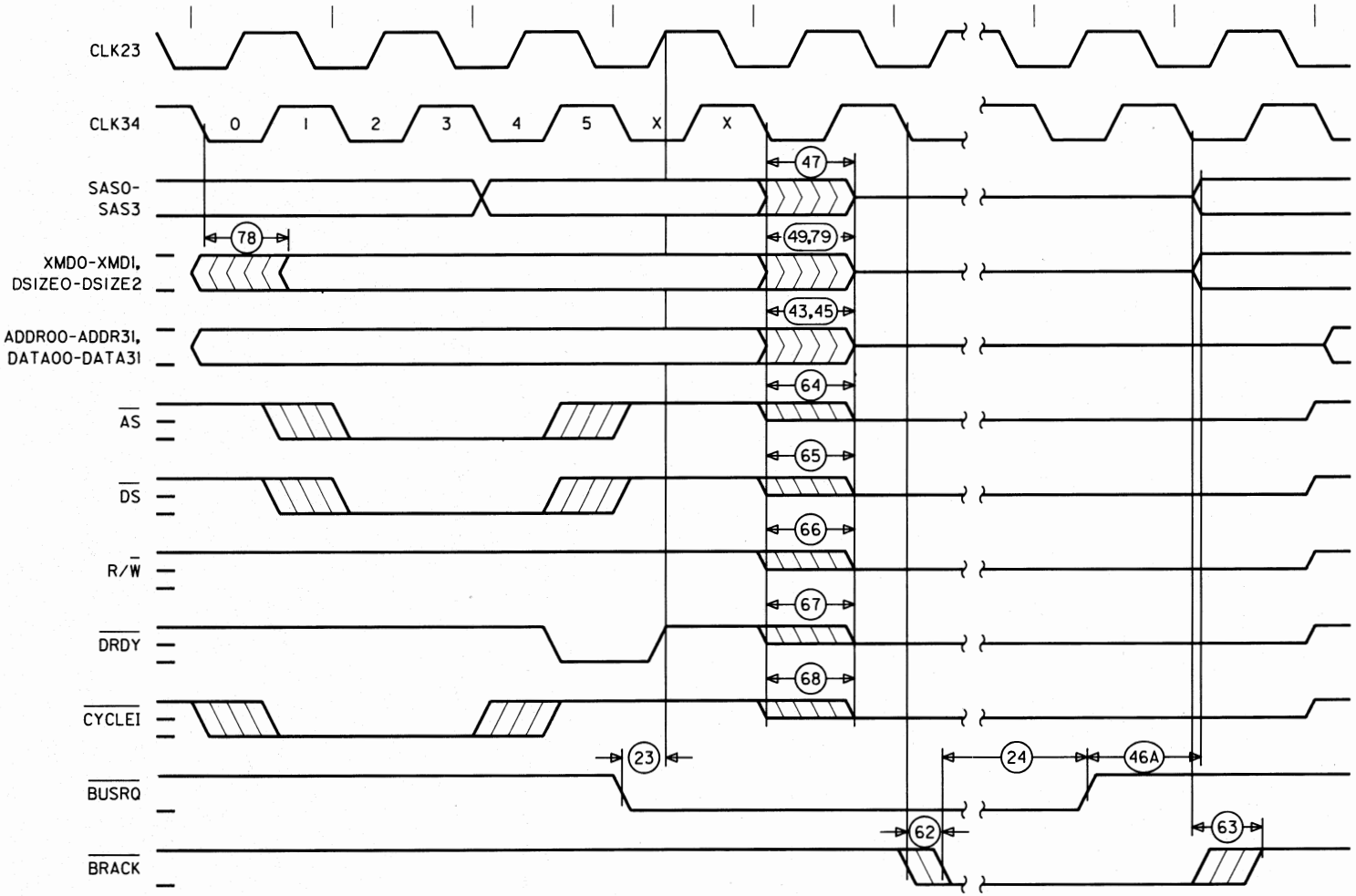


Figure 14. Bus Arbitration Timing

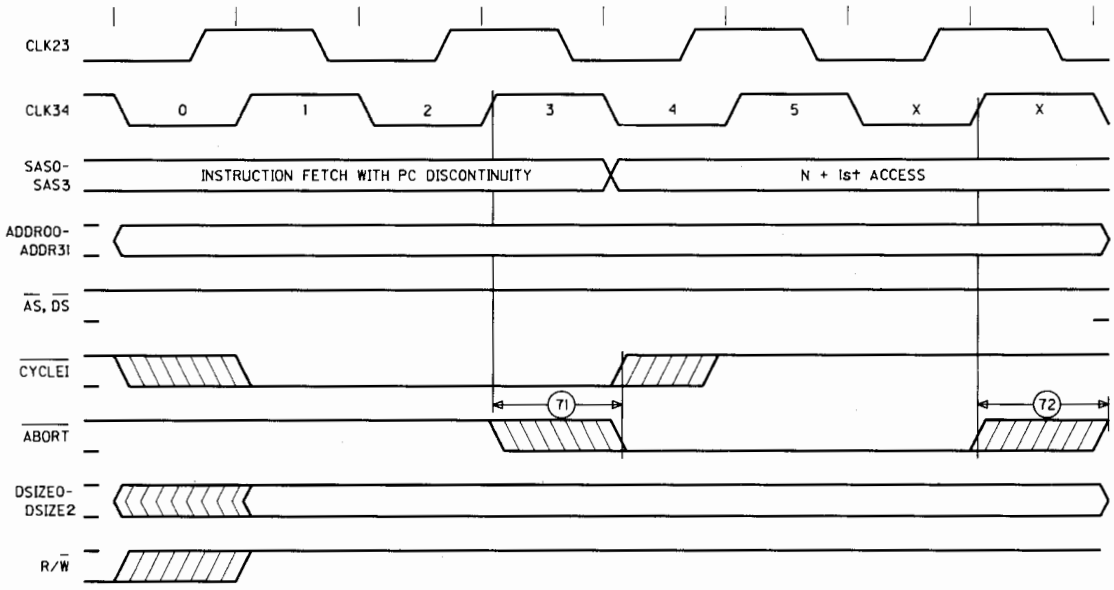
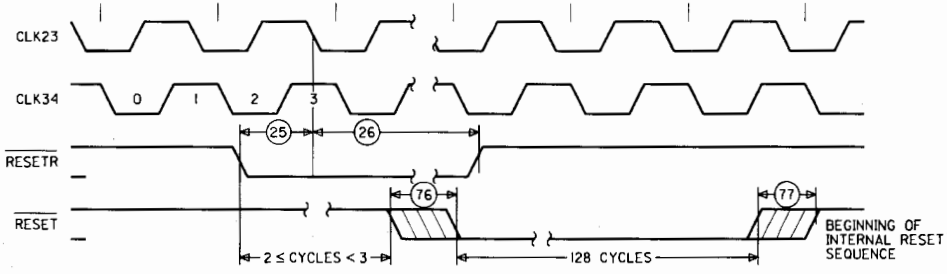


Figure 15. Aborted Access Timing



Notes:
RESETR must be asserted for at least 3 cycles to be recognized.
RESET is negated 128 cycles after negation of RESETR.

Figure 16. Reset Timing

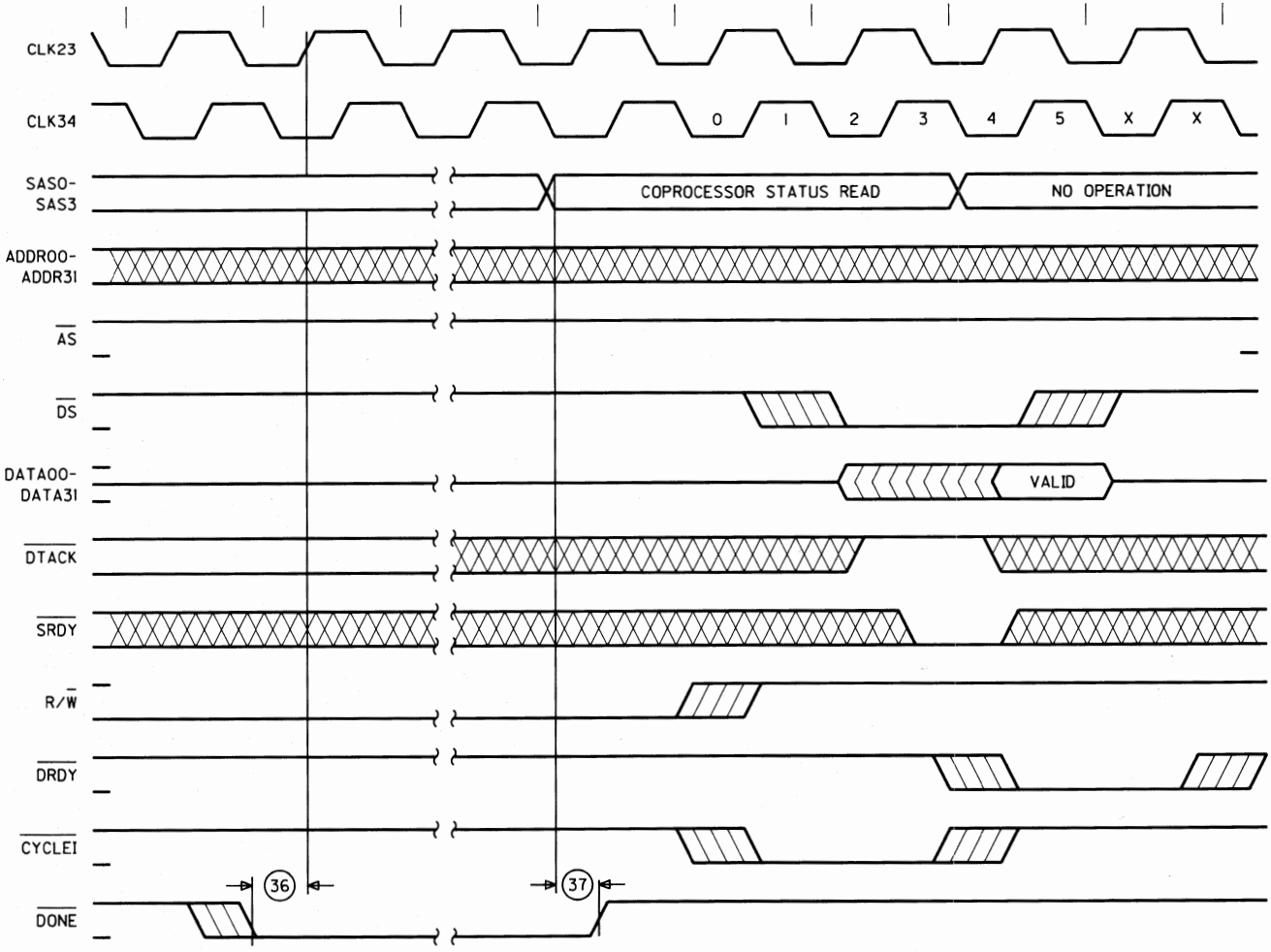
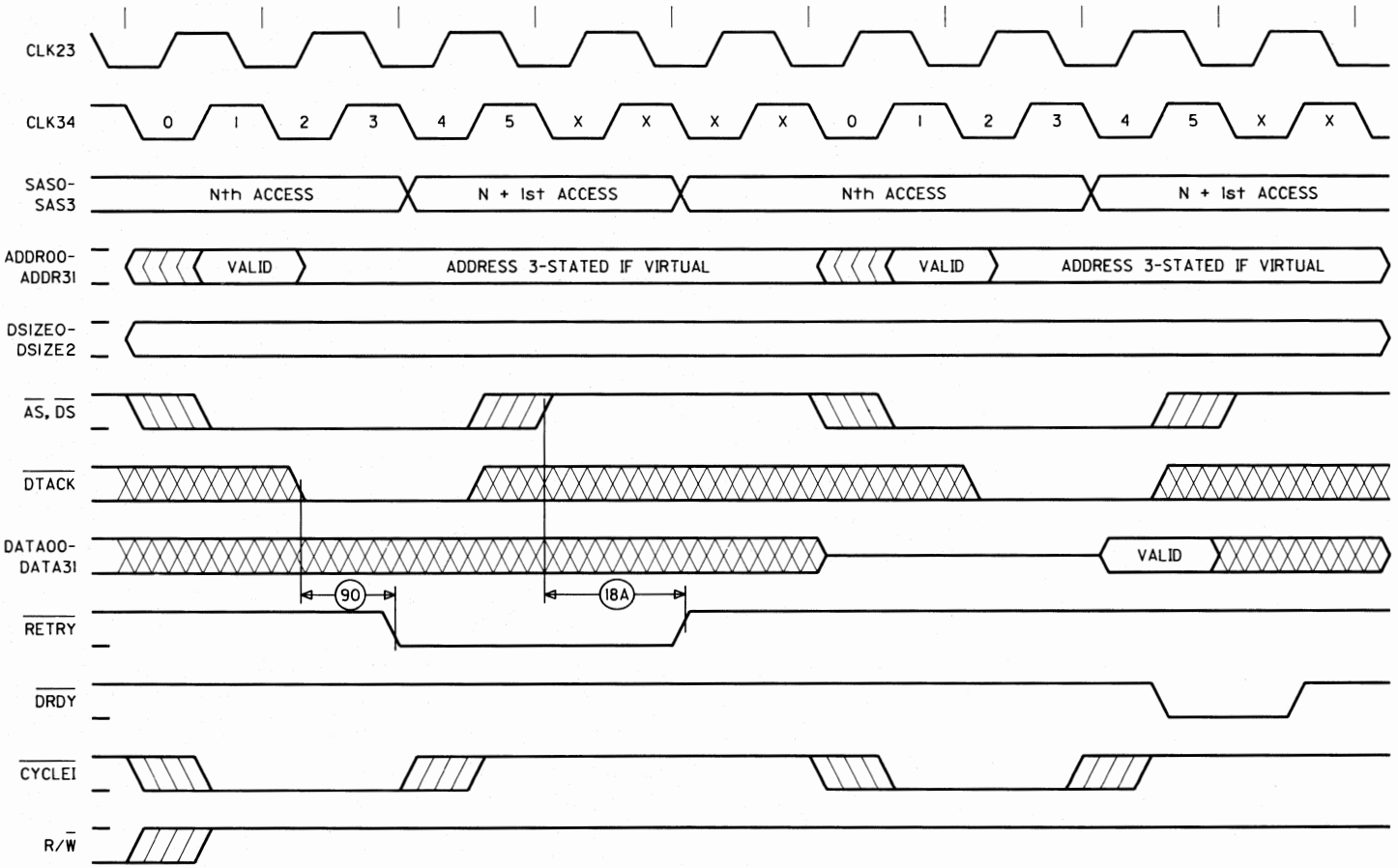
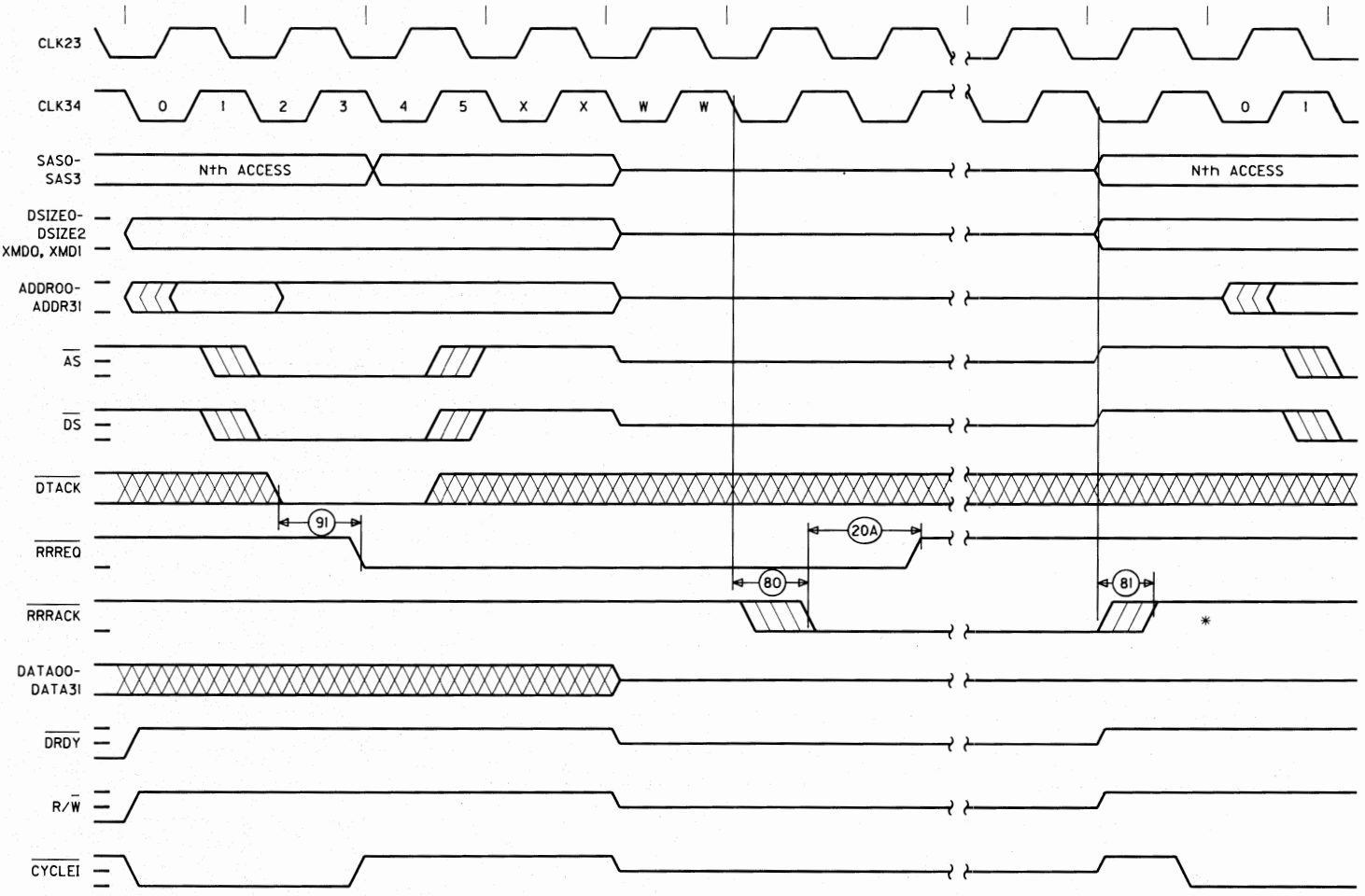


Figure 17. Coprocessor Status Read After $\overline{\text{DONE}}$ Timing



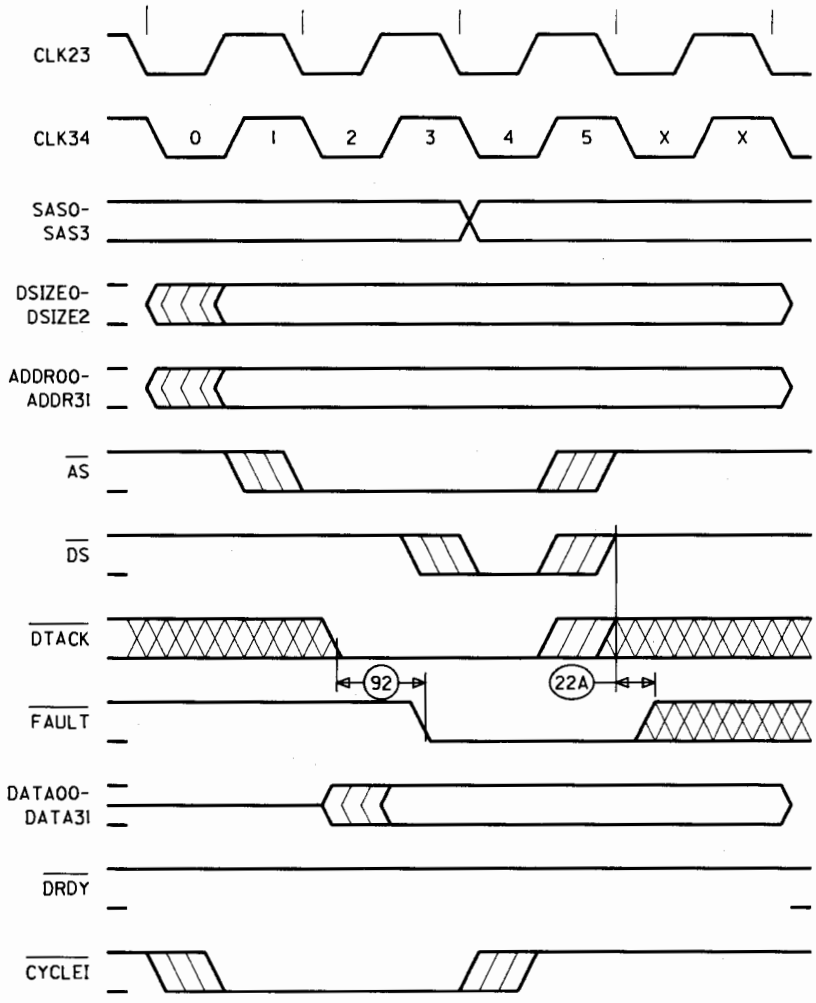
Note: Between the time $\overline{\text{RETRY}}$ is negated and state zero of the retried access, 2.5 to 3.5 cycles will elapse.

Figure 18. Retry After $\overline{\text{DTACK}}$ Timing (Read Transaction is Shown)



* Shown pulled high by external resistor.

Figure 19. Relinquish and Retry after $\overline{\text{DTACK}}$ Timing



Notes:
FAULT must meet set-up time with respect to 3, 4 edge, after DTACK.
SRDY is don't care at zero wait states.

Figure 20. Fault After DTACK Timing (Write Transaction is Shown)

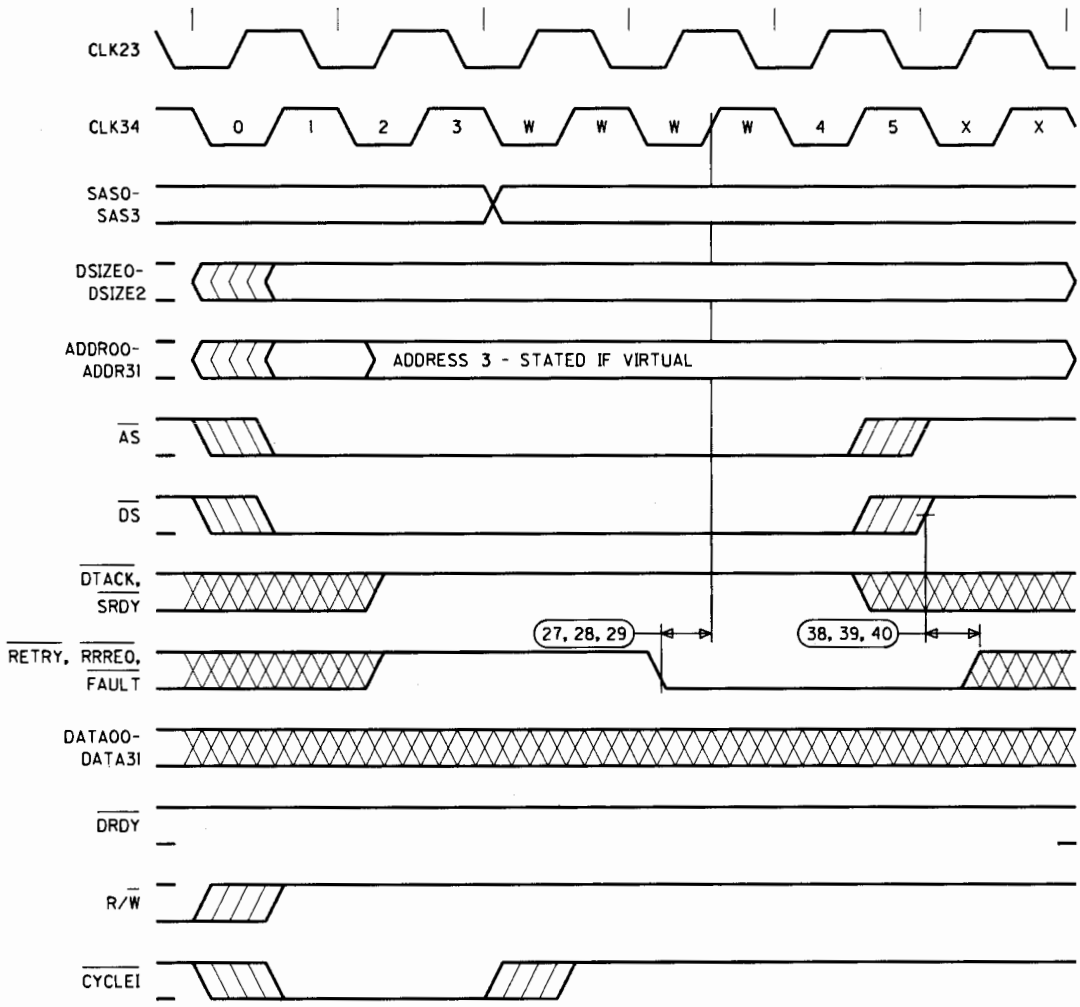


Figure 21. Bus Exception (RETRY, FAULT, or RRREQ) with no DTACK or SRDY Timing (Read Transaction is Shown)

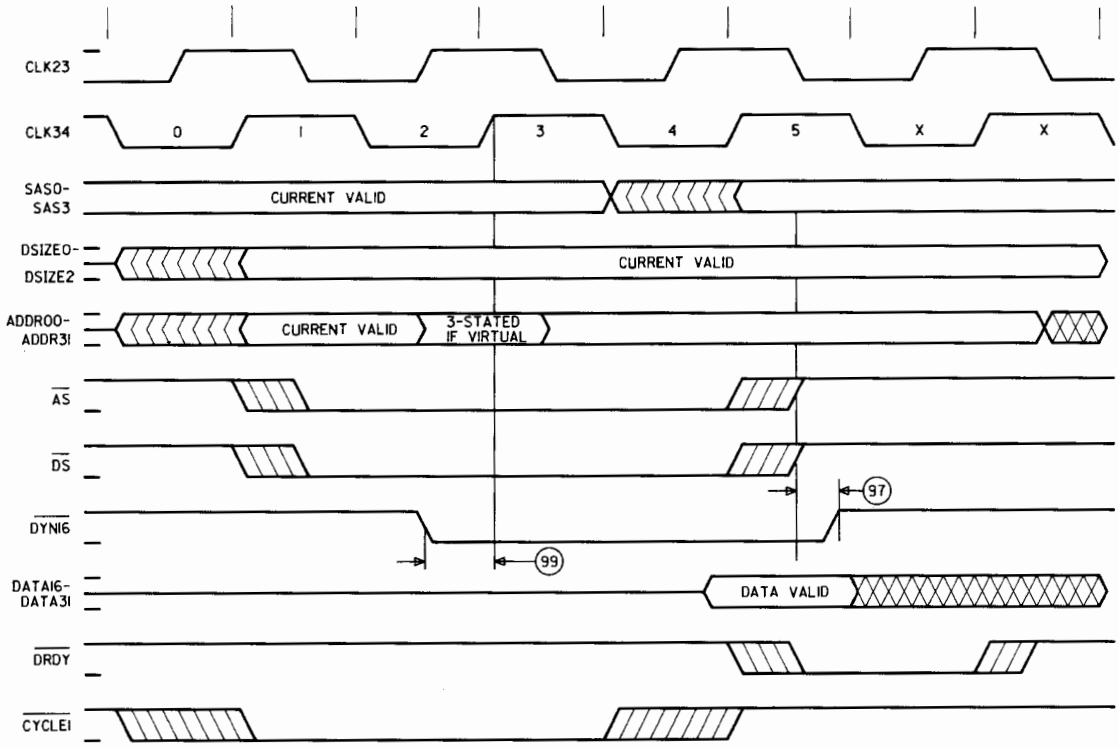


Figure 22. Dynamic Bus Sizing with $\overline{\text{DYN16}}$ by Itself (Read Transaction)

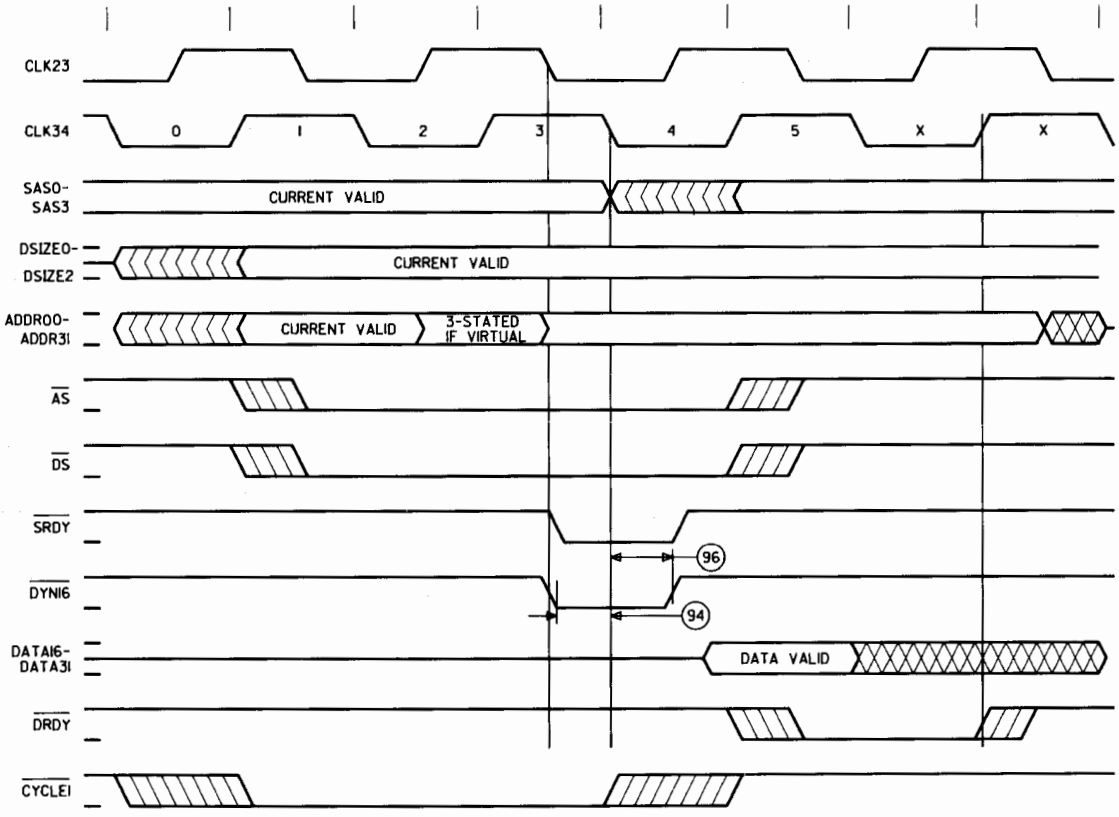
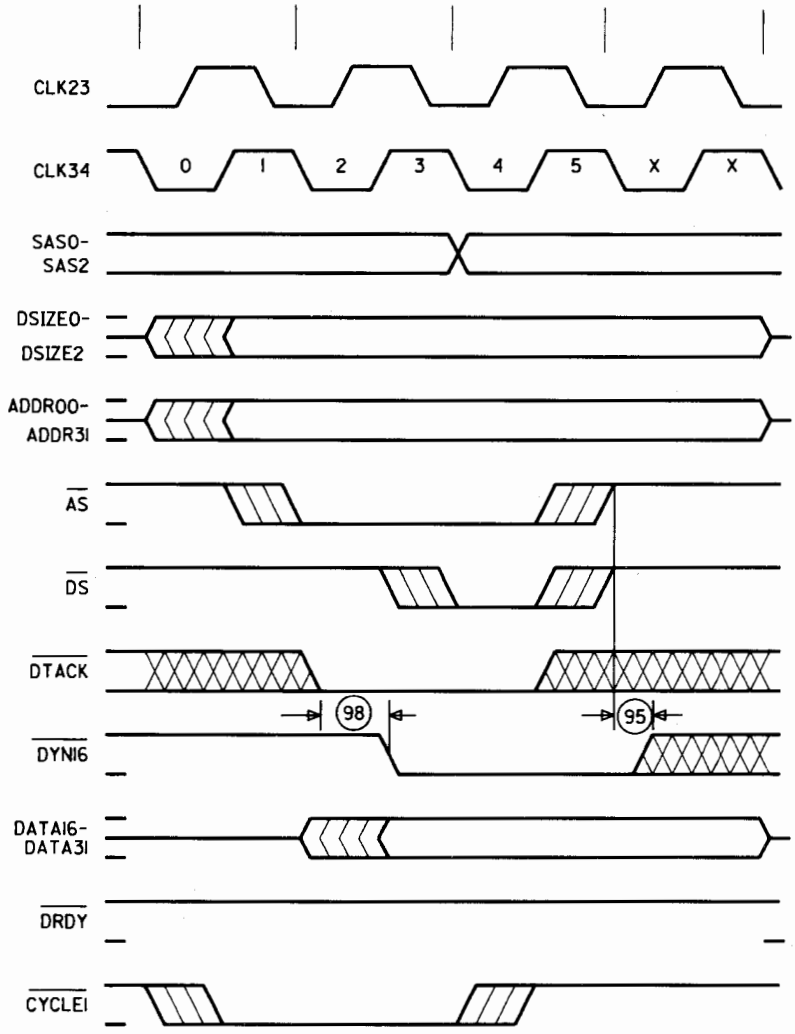


Figure 23. Dynamic Bus Sizing where $\overline{\text{DYN16}}$ is Asserted with $\overline{\text{SRDY}}$ (Read Transaction)



Notes:
DYN16 must meet set-up time with respect to 3, 4 edge, after DTACK.
SRDY is don't care at zero wait states.

Figure 24. DYN16 After DTACK Timing (Write Transaction is Shown)

Electrical Characteristics

Inputs

All inputs, except the two CMOS input clocks, are TTL compatible. The TTL inputs and clock inputs are specified separately below.

Table 37. DC Input Parameters

Inputs		Min	Nom	Max	Unit
TTL input voltage	high-level	2.0	—	V_{CC} + 0.5	V
	low-level	–0.5	—	0.8	V
CMOS clocks input voltage	high-level	V_{CC} – 1.3	—	V_{CC} + 0.5	V
	low-level	0	—	0.8	V
TTL input loading current: For V_{IH} (2.0 V ≤ V_{IH} ≤ V_{CC})	high-level	0	—	0.01	mA
For V_{IL} (0 V ≤ V_{IL} ≤ 0.8 V)	low-level	–0.01	—	0	mA
CMOS clocks input loading current: For V_{IH} (V_{CC} – 1.3 V ≤ V_{IH} ≤ V_{CC})	high-level	0	—	0.01	mA
For V_{IL} (0 V ≤ V_{IL} ≤ 0.8 V)	low-level	–0.01	—	0	mA

Outputs

The four classes of outputs that can support TTL input voltage levels are:

Class 1: This class is capable of driving one TTL load or 8 PNP Schottky TTL loads and has current allowance for an external holding resistor employed in 3-state buffers. The minimum resistor value is 2.7 kΩ.

Class 2: This class is the same as Class 1, except it does not have current allowance for a holding resistor.

Class 3: The signal in this class is an open drain output used for wired-logic operations, allowing more than one device to drive a node without conflict. An external resistor is required to pull this signal high. The minimum pull-up resistor value is 510 Ω.

Class 4: This class is the same as Class 1; however, its minimum holding resistor value is 1.8 kΩ.

The following lists the outputs assigned to each class:

Class 1		Class 2	Class 3	Class 4
$\overline{ABORT}$	DSIZE0—DSIZE2	ADDR00—ADDR31	$\overline{RRRACK}$	SAS0—SAS3
$\overline{AS}$	IQS0,IQS1	DATA00—DATA31		
$\overline{BRACK}$	R/$\overline{W}$			
$\overline{BUSRQ}$	$\overline{RESET}$			
$\overline{CYCLEI}$	$\overline{SOI}$			
$\overline{DRDY}$	$\overline{VAD}$			
$\overline{DS}$	XMD0, XMD1			

Table 38. DC Output Parameters

Outputs		Min	Max	Unit
Output sink current (I_{OL}) ($V_{OL} \leq 0.4$ V)	Class 1	—	5.5	mA
	Class 2	—	3.5	mA
	Class 3*	—	10.0	mA
	Class 4	—	6.5	mA
Output source current (I_{OH}) ($V_{OH} \geq 2.4$ V)	Class 1	—	–5.5	mA
	Class 2	—	–3.5	mA
	Class 3*	—	–10.0	μA
	Class 4	—	–5.5	mA
Output logic levels	high-level	2.4	—	V
	low-level	—	0.4	V

* See explanation of Class 3.

Operating Conditions

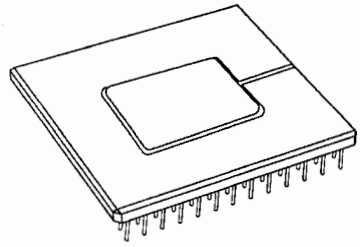
Table 39. Operating Conditions

Parameter		Symbol	Min	Nom	Max	Unit
Supply voltage		V_{CC}	4.75	5.00	5.25	V
Input load capacitance	TTL inputs	C_{IN}	—	—	12	pF
	CMOS clocks		—	—	7	pF
Total output load capacitance	Class 1	C_L	—	—	130	pF
	Class 2		—	—	130	pF
	Class 3		—	—	130	pF
	Class 4		—	—	130	pF
Ambient temperature at the microprocessor pins		T_A	0	—	70	°C
Storage temperature		—	–55	—	125	°C
Humidity range		—	5%	—	95%	RNCH
Power dissipation @ 24 MHz		PD	—	—	1.92	W
Operating frequency		F	24	—	—	MHz
Capacitive derating factor @ 24 MHz ($25pF \leq C_L \leq 225pF$)		dt/dC	—	0.8	—	ns/pF

WE® 32101 Memory Management Unit

Description

The *WE* 32101 Memory Management Unit (MMU) is a 32-bit bus-structured device that provides logical-to-physical address translation, memory organization, control, and access protection for the *WE* 32100 Microprocessor. The MMU performs address translation by mapping virtual memory addresses to physical memory addresses, which allows 2^{32} bytes of virtual memory and up to 2^{32} bytes of physical memory per process. It supports both demand paged and demand segmented virtual memory systems, which allows large programs to efficiently use physical memory space. The MMU also allows the use of shared segments for intertask communication. Access privileges for each segment provide system protection. The *WE* 32101 Memory Management Unit is available in 10-, 14-, and 18-MHz versions; requires only a single 5 V power source; and is in a 125-pin square, ceramic pin grid array (PGA) package.



Features

- Manages 4 Gbytes of virtual address and up to 4 Gbytes of physical address space
- Paged and nonpaged segmentation support
- On-chip segment descriptor cache and two-way, set-associative page descriptor cache
- On-chip cache miss-processing
- Four types of access protection at four execution levels
- Hardware support for *UNIX* System demand paging with automatic R & M bit update
- Shared segments managed by indirect segment descriptors
- Extensive fault detection and resolution capability
- Fully supports multiple MMU configuration
- Low-power CMOS technology

Description 145
Features 145
User Information 146
Architectural Summary 146
Memory Segmentation 146
Virtual Address Fields 146
Descriptors 146
Access Permission Field (Acc) 148
MMU Internal Elements 148
Descriptor Caches 149
Section RAMs 151
MMU Registers 151
Configuration Register (CR) 151
Virtual Address Register (VAR) 153
Fault Code Register (FLTCR) 153
Fault Address Register (FLTAR) 153
Memory-Mapped Peripheral Mode 154
Faults 154
MMU Reset 156
Pin Descriptions 157
Numerical Order 157
Functional Group 159
MMU-CPU Interconnections 163
Single MMU Configuration 163
Multiple MMU Configuration 163
Operating System Considerations 165
Characteristics 166
Timing Characteristics 166
Electrical Characteristics 180
Inputs 180
Outputs 180
Operating Conditions 181

User Information

Architectural Summary

The *WE* 32101 Memory Management Unit (MMU) may be used when virtual memory storage is used for a system containing the *WE* 32100 Microprocessor.

Note: Those who will be using the *WE* 321DS Microprocessor Development System to develop applications with the *WE* 32101 MMU should read the *WE* 321DS Microprocessor Development System User Manual prior to designing the application.

Memory Segmentation

The MMU divides the virtual address space into four sections which, in turn, may be subdivided into as many as 8K segments per section. These segments may be either *contiguous* or *paged* and are mapped into the physical address space by the MMU. A contiguous segment can be as large as 128 Kbytes; a paged segment can contain up to sixty-four 2 Kbyte pages. Contiguous and paged segments must start at an address in physical memory that is a multiple of 32 bytes. The following access permissions are associated with segments (a separate value may be assigned for each execution level):

- Execute only
- Read/execute
- Read/write/execute
- No access

Virtual Address Fields

The MMU requires the division of virtual addresses into three fields for contiguous segments and four fields for paged segments. A virtual address referencing a contiguous segment is divided into three fields: a section identifier (SID) field, which specifies the section of virtual address space; a segment select (SSL) field, which specifies the segment within the section; and a segment offset (SOT) field, which specifies the byte within the segment. The format of a virtual address in a contiguous segment is shown on Figure 1.

For paged segments, the SOT field is subdivided into a page select (PSL) field and a page offset (POT) field. The PSL field specifies the page in the segment; the POT field specifies the byte in the page. The format of a virtual address in a paged segment is shown on Figure 2.

Descriptors

The MMU performs address translation using descriptors that contain the information necessary for segment and page mapping. The MMU has two types of descriptors — segment descriptors (SD) for mapping contiguous and paged segments and page descriptors (PD) for mapping pages. An SD contains a segment base address that is added to an offset from the virtual address (SOT) to form the physical address. The PD contains a page base address that is concatenated with a page offset from the virtual address (POT) to form the physical address.

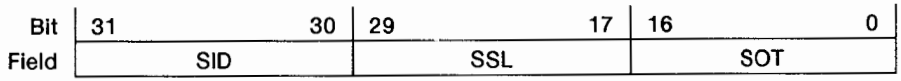


Figure 1. Virtual Address Fields for a Contiguous Segment

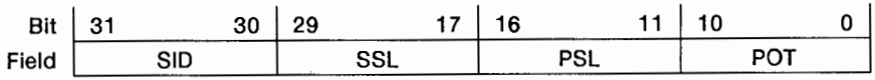


Figure 2. Virtual Address Fields for a Paged Segment

The SDs for each of the four sections of virtual memory are located in physical memory in segment descriptor tables (SDTs). There is one SDT associated with each section. The PDs for each segment are located in physical memory in page descriptor tables (PDTs), and there is one PDT associated with each paged segment. Contiguous segments are represented by an SDT entry only, while paged segments are represented by both an SDT and PDT entry. The SDT entry for a paged segment includes the physical base address of the PDT. The SD entry in the SDT is 8 bytes in length and occupies two successive words in physical memory. The SD format and a description of each field is given in Tables 1 and 2.

The PD entry in the PDT is 4 bytes in length. The PD format and a description of each field is shown in Table 5.

The MMU minimizes address translation times by storing a number of segment and page descriptors in its on-chip cache. It fetches these descriptors from physical memory, as needed, without CPU intervention. Also, the MMU contains other registers and data structures that support the operating system.

Each execution level (kernel, executive, supervisor, and user) in the access permission field contains a 2-bit code defining the type of access permission associated with each level. These codes are defined in Table 3.

The MMU checks the type of access requested by the CPU and determines which permissions are needed to allow the access. If the permissions do not allow the access, an access fault occurs. The permissions needed for each access type are shown in Table 4.

Table 1. First Word of a Segment Descriptor Field

Bit	31 24	23 10	9 8	7	6	5	4	3	2	1	0
Field	Acc	Max Off	Res	I	V	R	T	$	C	M	P

Bit	Field	Contents	Description
0	P	Present	Specifies if a segment is present in physical memory. If P=1, segment is present; if P=0, segment is not present.
1	M	Modified	When M=1, the MMU configuration register mod bit is set and the segment has been modified (see MMU Registers).
2	C	Contiguous	Specifies if the segment is paged (C=0) or contiguous (C=1).
3	$	Cacheable	Determines state of $\overline{\text{CABLE}}$ during miss-processing and updating of R & M bits. If $=0, then $\overline{\text{CABLE}}$=1; if $=1 then $\overline{\text{CABLE}}$=0.

Table 1. First Word of a Segment Descriptor Field (Continued)

Bit	Field	Contents	Description
4	T	Object trap	Object trap fault occurs if T is set (1), access rights are not violated, and C=1.
5	R	Referenced	When R=1, the MMU configuration register referenced field is a 1 and the segment has been referenced (see MMU Registers).
6	V	Valid	If V=1, SD is valid. If V=0, SD is not valid.
7	I	Indirect	If I=1, this SD points to another segment.
8—9	Res	Reserved	Reserved for future use. Must always be cleared (0).
10—23	Max off	Maximum offset	Used to calculate the maximum offset from the start of the segment that a virtual address may specify. Used for checking segment length.
24—31	Acc	Access permissions	Specifies the access type and permissions for each execution level (see Access Permission Field).

Table 2. Second Word of a Segment Descriptor Field

Bit	31 — 3	2 — 0
Field	Address (high-order 27 or 29 bits)	Soft

Bit	Field	Contents	Description
0—2	Soft	Software	Reserved for software use. The MMU does not change the value of this field at any time.
3—31	Address	Address	Pointer (high-order 27 bits) to address in main memory for contiguous segments and to page descriptor table for paged segments. Pointer (29 bits) to physical address of another SD when indirection feature is used.

Access Permission Field (Acc). The MMU uses this field during address translations. Its format is shown on Figure 3.

Bit	31 — 30	29 — 28	27 — 26	25 — 24
Field	Kernel	Exec	Super	User

Figure 3. Access Permission Field Format

Table 3. Access Permission Codes

Value	Permission	Description
00	NA	No access
01	EO	Execute only
10	RE	Read/execute
11	RWE	Read/write/execute

MMU Internal Elements

Figure 4 is a programming mode for the MMU. The MMU is functionally divided into descriptor caches, section RAMs, and MMU registers that can be address only in peripheral mode (see Memory Mapped Peripheral Mode).

Table 4. Permissions Needed for Each Access Type

Access Type	Permission Needed	Equivalent Operation
Interlocked read	RWE	Read and write
Address fetch	RWE or RE	Read
Operand fetch	RWE or RW	Read
Write	RWE	Write
Instruction fetch	EO, RWE, or RE	Execute
Instruction fetch after discontnuity	EO, RWE, or RE	Execute
Instruction prefetch	EO, RWE, or RE	Execute
Move translated (MT)	RWE or RE	Read

Descriptor Caches

The MMU uses two descriptor caches. The segment descriptor cache (SDC) consists of 32 descriptors, each 64 bits in length, and is divided into four parts which correspond to the four sections. The format of the cached descriptors is different from that of the descriptors that are part of the SDTs in memory. The SDC fields are described in Table 6.

The page descriptor cache (PDC) consists of sixty-four 64-bit descriptors organized in a two-way set-associative configuration. It is divided into four parts, which correspond to the four sections of virtual memory. The PDC fields are described in Table 7.

Table 5. Page Descriptor Field

Bit	31 — 11	10 — 8	7 — 6	5	4	3	2	1	0
Field	Page Address	Soft	Res	R	W	Res	L	M	P

Bit	Field	Contents	Description
0	P	Present	Specifies if the page is present in primary memory. If P=1, page is present; if P=0, page is absent.
1	M	Modified	When M=1, the page associated with the descriptor has been modified.
2	L	Last	Indicates (L=1) that this is the last page in the segment, and that it is not a multiple of 2-Kbytes. L=0 for all 2-Kbyte pages.
3	Res	Reserved	Reserved for future use. Must always be cleared.
4	W	Fault on write	If W=1, a page-write fault occurs. If W=0, translation continues normally. This bit is not examined during miss-processing.
5	R	Referenced	If R=1, the page associated with the descriptor has been referenced.
6—7	Res	Reserved	Reserved for future use. Must always be cleared.
8—10	Soft	Software	Reserved for software use. The MMU does not change the value of this field at any time.
11—31	Page address	Page address	Used to find the address of the first byte of a page in physical memory. Field multiplied by 2K equals page base address.

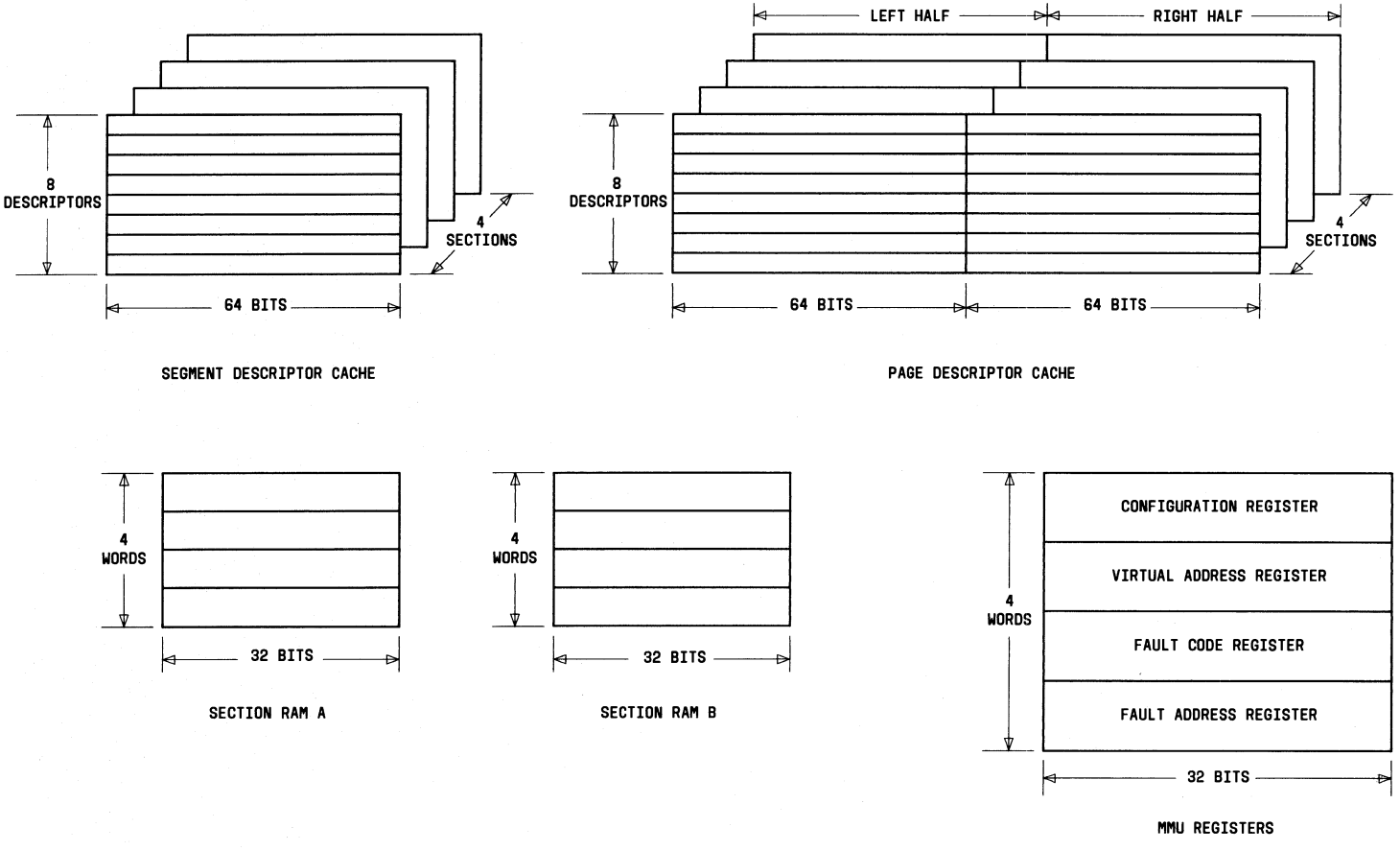


Figure 4. MMU Programming Model

Table 6. Segment Descriptor Cache Fields

Bit	63 37	36	35	34	33	32	31 24	23 10	9 0
Field	Address	T	$	C	M	G	Acc	Max Off	Tag

Bit	Field	Contents	Description
0—9	Tag	Identifying tag	Used to uniquely identify the descriptor (bits 20—29 of the virtual address).
10—23	Max off	Maximum offset	Used to calculate the maximum offset from the start of the segment that a virtual address may specify. (Used for checking segment length.)
24—31	Acc	Access permissions	Specifies the access type and permissions for each execution level.
32	G	Good	Specifies if cache entry contains a descriptor (G=1) or if the location is empty (G=0).
33	M	Modified	When M=1, the MMU configuration register M bit is set and the segment has been modified.
34	C	Contiguous	Specifies if the segment is paged (C=0) or contiguous (C=1).
35	$	Cacheable	Determines state of $\overline{\text{CABLE}}$ during miss-processing and updating of R & M bits. If $=0, then $\overline{\text{CABLE}}$=1; if $=1, then $\overline{\text{CABLE}}$=0.
36	T	Object trap	Object trap fault occurs if T is set, access rights are not violated, and C=1.
37—63	Address	Address	Pointer to address in main memory for contiguous segments or to page descriptor table for paged segments.

Section RAMs

The MMU contains two RAM areas called section RAM A (SRAMA) and section RAM B (SRAMB). Both of these areas contain four 32-bit words that describe the base address of the SDT for each section (contained in SRAMA) and the length (i.e., number of entries) of the SDT (contained in SRAMB) for each section. The SRAMA and SRAMB entry formats are shown on Figure 5. SRAMA entries are used during miss-processing (a segment descriptor needed for address translation is not present in the MMU descriptor cache) to access descriptor tables in memory to fetch the necessary descriptors into the descriptor caches. SRAMB entries contain one less than the number of segments in each section and are used for checking the SDT bounds during miss-processing.

MMU Registers

The MMU contains four 32-bit registers that provide information pertaining to the present state of the MMU.

Configuration Register (CR). This register is used by the operating system to determine whether the referenced (R) and modified (M) bits for segment descriptors are to be updated in memory. The format and field descriptions for the CR are shown in Table 8.

Table 7. Page Descriptor Cache Fields

Bit	63 43	42 39	38	37	36	35	34	33	32	31 24	23 16	15 0
Field	Address	Res	U	R	W	$	L	M	G	Acc	Res	Tag

Bit	Field	Contents	Description
0—15	Tag	Identifying tag	Used to uniquely identify the descriptor (bits 13—16 and bits 18—29 of the virtual address).
16—23	Res	Reserved	Reserved for future use. Must always be cleared.
24—31	Acc	Access permissions	Specifies the access type and permissions for each execution level.
32	G	Good	Specifies if cache entry contains a descriptor (G=1) or if the location is empty (G=0).
33	M	Modified	When M=1, the page associated with the descriptor has been modified.
34	L	Last	Indicates (L=1) that this is the last page in the segment and that it is not a multiple of 2 Kbytes. L=0 for all 2 Kbyte pages.
35	$	Cacheable	Determines state of $\overline{\text{CABLE}}$ during miss-processing and updating of R & M bits. If $=0, then $\overline{\text{CABLE}}$=1; if $=1, then $\overline{\text{CABLE}}$=0.
36	W	Fault on write	If W=1, a page-write fault occurs. If W=0, translation continues normally. This bit is not examined during miss-processing.
37	R	Referenced	If R=1, the page associated with the descriptor has been referenced.
38	U	Used	Specifies whether the right-hand (U=1) or left-hand (U=0) entry in a 2-entry set was replaced more recently. This bit exists only in left-hand PDC.
39—42	Res	Reserved	Reserved for future use. Must always be cleared.
43—63	Address	Address	A pointer to the start of a page in physical memory.

Bit	31 5	4 0
Field	SDT Address (high-order 27 bits)	Reserved

SRAMA Entry Format

Bit	31 23	22 10	9 0
Field	Reserved	SDT Length	Reserved

SRAMB Entry Format

Figure 5. SRAMA and SRAMB Entry Formats

Virtual Address Register (VAR). The VAR contains the virtual address to be translated by the MMU. The VAR is overwritten each time a translation is performed or when the CPU writes to it in the peripheral mode. (See Memory-Mapped Peripheral Mode.)

Fault Code Register (FLTCR). This register keeps a record of the last fault and operational states in the MMU. The FLTCR format and a description of each field are given in Table 9.

Fault Address Register (FLTAR). This register contains the virtual address that was being processed when the last fault that caused a write to the FLTCR and FLTAR occurred. The contents are changed when the CPU writes to it in peripheral mode (see Memory-Mapped Peripheral Mode).

Table 8. Configuration Register Fields

Bit	31 3	2	1	0
Field	Res	$	Ref	Mod

Bit	Field	Contents	Description
0	Mod	Modified	If Mod=1, the SD M bit is updated on the first write to a segment.
1	Ref	Referenced	The R bit in the SD is set (R=1) when SD is brought into the SDC as a result of miss-processing. When R=0, the R bit in the SD is not updated.
2	$	Cacheable	Determines state of $\overline{\text{CABLE}}$ during miss-processing and updating of R & M bits. If $=0, then $\overline{\text{CABLE}}$=1; if $=1, then $\overline{\text{CABLE}}$=0.
3—31	Res	Reserved	Reserved for future use. If read, zeros are returned.

Table 9. Fault Code Register Fields

Bit	31 11	10 7	6 5	4 0
Field	Reserved	Access Requested	Access Xlevel	Fault Type

Bit	Field	Contents	Description
0—4	Fault type	Fault type	Contains value of fault type that occurred (see Faults).
5—6	Access Xlevel	Access execution level	Records execution level of access that was requested when fault occurred. Bit 5 is the least significant bit. **Access Xlevel** / **Execution Level** 00 Kernel 01 Executive 10 Supervisor 11 User

Table 9. Fault Code Register Fields (Continued)

Bit	Field	Contents	Description
7—10	Access request	Access requested	Access type requested by CPU when a fault occurred. Bit 7 is the least significant bit.

Access Requested Value	Access Type Description
0000	Move translated (MT)
0001	Support processor data write
0011	Support processor data fetch
0111	Interlocked read
1000	Address fetch
1001	Operand fetch
1010	Write
1100	Instruction fetch after discontinuity
1110	Instruction fetch

11—31	Reserved	Reserved	Reserved for future use. If read, zeros are returned.

Memory-Mapped Peripheral Mode

This mode of operation allows many of the elements of the MMU to be addressed as memory locations, i.e., the MMU is treated as a memory-mapped peripheral. Each of the elements shown on Figure 4 can be read from and written to, and all occupy some of the physical address space. A peripheral mode access occurs when the MMU chip-select is asserted ($\overline{MCS}$=0). The MMU then interprets physical addresses, as shown in Table 10 and on Figure 6.

Faults

A fault occurs when the address translation process cannot be completed. Specifically, it is an error condition associated with read and write operations. The MMU handles faults with the FLTCR and FLTAR registers. The fault type is indicated in the FLTCR fault-type field and the virtual address of the fault is in the FLTAR. The FLTCR fault-type field is described in Table 11.

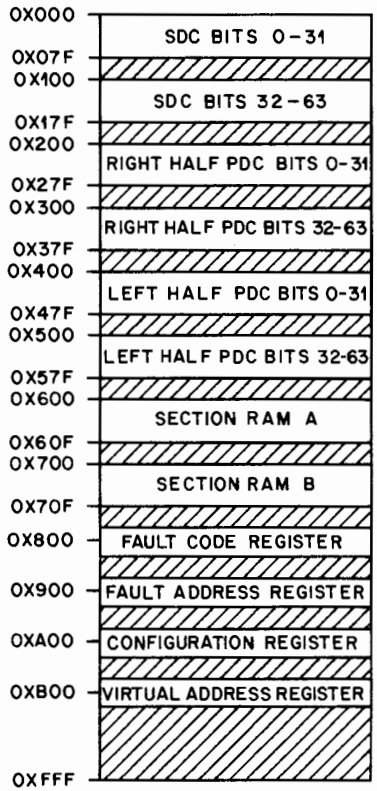


Figure 6. Peripheral Mode Memory Map

Table 10. Peripheral Mode Address Fields

Bit	31 12	11 8	7	6 2	1 0
Field	Res	Entity	Res	Index	Res

Bit	Field	Contents	Description
0—1	Res	Reserved	Ignored by MMU and treated as zeros.
2—6	Index	Index	Used to index each addressable entity. Bits 2—6 used when caches are accessed; bits 2—3 used when section RAMs are accessed; and bits 2—6 ignored when registers are accessed.
7	Res	Reserved	Ignored by MMU and treated as zeros.
8—11	Entity	Entity	Selects which entity is to be accessed as per entity value specified. Bit 8 is the least significant bit.
12—31	Res	Reserved	Ignored by MMU and treated as zeros.

Entity Value	Entity Addressed
0000	SDC bits 0—31
0001	SDC bits 32—63
0010	Right half of PDC bits 0—31
0011	Right half of PDC bits 32—63
0100	Left half of PDC bits 0—31
0101	Left half of PDC bits 32—63
0110	Section RAM A
0111	Section RAM B
1000	Fault code register
1001	Fault address register
1010	Configuration register
1011	Virtual address register

Note: All peripheral mode accesses must be word accesses.

Table 11. FLTCR Fault-Type Field Description

Fault Type	Fault Name	Description
00000	No fault	No fault occurred.
00001	Miss-processing memory	Occurs when a memory fault is signaled to the MMU during a descriptor read for miss-processing purposes.
00010	R & M update memory	Occurs when a memory fault is signaled to the MMU during a descriptor read or write for R & M bit update purposes.
00011	SDT length	Occurs when the SSL of the virtual address exceeds the SDT length field of SRAMB.
00100	Page write	If translation for a write or interlocked read access uses a PD, and the access and segment offset (L bit) check causes no faults, the W bit of the PD is checked. If W=1, this fault occurs.

Table 11. FLTCR Fault-Type Field Description (Continued)

Fault Type	Fault Name	Description
00101	PDT length	Occurs when the lower 17 bits of a virtual address specify a reference beyond the address range in the maximum offset field of the corresponding cached SD. Occurs only when fetching a PD or a paged segment.
00110	Invalid SD	Occurs if SD is marked invalid (V=0).
00111	Segment not present	Occurs if P bit in SD is 0 and V=1, C=1, and I=0.
01000	Object trap	Occurs if there is a valid translation (i.e., V=1, I=0 and access permissions are not violated) using an SD, C=1, and the T bit in the SD is set.
01001	PDT not present	Occurs if P bit in SD is 0, V=1, C=0, and I=0. In this case, the PDT for the segment is not present in physical memory.
01010	Page not present	Occurs when P bit in PD is 0.
01011	Too many indirections	Occurs if six consecutive SDs with I=1 are encountered during miss-processing.
01101	Access	Occurs if access requested is not permitted by access permissions field of the SD or PD used.
01110	Segment offset	Occurs when value in SOT specifies a reference beyond the valid address range indicated by max off field of the SD for contiguous segment. For a paged segment, it occurs when there is a PD cache hit (L=1) and the page segment boundary is violated.
01111	Access and segment offset	Occurs when both an access violation and a segment offset violation occurs.
11111	Double page hit	Occurs if the MMU attempts to translate a virtual address and finds two PDs corresponding to it in the PD cache.

Note: All fault type values not listed are unassigned.

MMU Reset

The MMU is reset by both system and CPU-initiated resets. When $\overline{\text{RESET}}$ is activated, the MMU sets all fault pins to the inactive state. The contents of all MMU elements readable in peripheral mode are unchanged, with the exception of the FLTCR fault-type field, which is made 0. The MMU then enters a state in which it can be directed to perform any of its usual functions. When the MMU $\overline{\text{RESET}}$ pin is asserted, the MMU output pins are set to the nonasserted state, as shown in Table 12.

Note: Reset the MMU after every power-up to eliminate random register, cache, or section RAM contents.

Table 12. MMU Output Signal States After Reset

Signal	State	Normally Tied High
ADDR00—ADDR31	Z	No
$\overline{\text{CABLE}}$	Z	Yes
$\overline{\text{CONTIG}}$	Z	Yes
DATA00—DATA31	Z	No
$\overline{\text{DSHAD}}$	Z	Yes
DSIZE0, DSIZE1	Z	No
$\overline{\text{EPAS}}$	Z	Yes
$\overline{\text{MCYCLEI}}$	Z	Yes
$\overline{\text{EMCYCLEI}}$	Z	Yes
$\overline{\text{MDRDY}}$	Z	Yes
$\overline{\text{MDS}}$	Z	Yes
$\overline{\text{MFAULT}}$ (Output)	Logic 1	No
$\overline{\text{MFAULT}}$ (Input)	Z	No
$\overline{\text{MPAS}}$	Z	Yes
MR/$\overline{\text{W}}$	Z	Yes
$\overline{\text{RMW}}$	Z	Yes

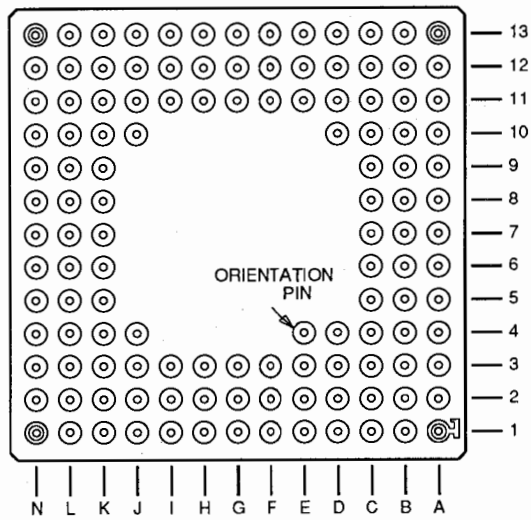
Note: Z = High-impedance.

Pin Descriptions

The *WE* 32101 Memory Management Unit is available in a 125-pin square, ceramic PGA. Figure 7 and Tables 13—19 describe the pin assignments.

The term *asserted* in the pin function descriptions means that the signal is driven to its active state either by the microprocessor (outputs) or an external device (inputs). The term *negated* means that the signal is driven to its inactive state. A bar over a signal name (e.g., $\overline{\text{AS}}$) indicates that the signal is active low, logic 0. The 0 bit is the least significant bit for signals which occupy two or more pins (e.g., DSIZE0—DSIZE1). In the signal type column, I indicates an input, O an output, and I/O a bidirectional signal.

Table 13 lists signals in numerical order by pin number for the 125-pin square PGA packages; Tables 14—19 describe the signals by functional group.



Bottom View

Figure 7. WE 32101 Memory Management Unit 125-Pin Square, Ceramic PGA Package

Numerical Order

Table 13. Pin Descriptions – 125-Pin Square PGA Package

Pin	Name	Type	Description
A1	$\overline{\text{MRESET}}$	—	Reset
A2	XMD1	I	Execution level 1
A3	DATA00	I/O	Data 00
A4	ADDR00	I/O	Address 00
A5	ADDR01	I/O	Address 01
A6	ADDR03	I/O	Address 03
A7	DATA03	I/O	Data 03
A8	DATA06	I/O	Data 06
A9	DATA04	I/O	Data 04
A10	ADDR08	I/O	Address 08
A11	ADDR05	I/O	Address 05
A12	ADDR09	I/O	Address 09
A13	ADDR11	I/O	Address 11
B1	$\overline{\text{CONTIG}}$	O	Contiguous segment
B2	SAS2	I	Access status 2
B3	XMD0	I	Execution level 0
B4	DATA01	I/O	Data 01
B5	DATA02	I/O	Data 02
B6	ADDR02	I/O	Address 02

Table 13. Pin Descriptions – 125-Pin Square PGA Package (Continued)

Pin	Name	Type	Description
B7	ADDR04	I/O	Address 04
B8	DATA05	I/O	Data 05
B9	DATA07	I/O	Data 07
B10	ADDR06	I/O	Address 06
B11	ADDR07	I/O	Address 07
B12	ADDR10	I/O	Address 10
B13	DATA11	I/O	Data 11
C1	$\overline{\text{VAD}}$	I	Virtual address
C2	SAS3	I	Access status 3
C3	SPARE	—	Reserved
C4	GND	—	Ground
C5	+5V	—	Power
C6	GND	—	Ground
C7	+5V	—	Power
C8	GND	—	Ground
C9	+5V	—	Power
C10	GND	—	Ground
C11	DATA08	I/O	Data 08
C12	DATA12	I/O	Data 12
C13	DATA13	I/O	Data 13
D1	$\overline{\text{MCS}}$	I	MMU chip select
D2	SAS1	I	Access status 1
D3	GND	—	Ground
D4	SAS0	I	Access status 0
D10	DATA10	I/O	Data 10
D11	DATA09	I/O	Data 09
D12	ADDR13	I/O	Address 13
D13	DATA14	I/O	Data 14
E1	$\overline{\text{CABLE}}$	O	Cacheable
E2	$\overline{\text{TCSA}}$	I	Translation chip select A
E3	+5V	—	Power
E4	$\overline{\text{CCYCLEI}}$	—	CPU cycle initiate
E11	+5V	—	Power
E12	ADDR12	I/O	Address 12
E13	ADDR14	I/O	Address 14
F1	$\overline{\text{AS}}$	I	Address strobe
F2	$\overline{\text{TCSB}}$	I	Translation chip select B
F3	GND	—	Ground
F11	GND	—	Ground
F12	ADDR15	I/O	Address 15
F13	DATA15	I/O	Data 15
G1	$\overline{\text{HIGHZ}}$	I	High impedance
G2	$\overline{\text{MDS}}$	O	MMU data strobe
G3	+5V	—	Power
G11	+5V	—	Power
G12	DATA16	I/O	Data 16
G13	ADDR17	I/O	Address 17
H1	$\overline{\text{ABORT}}$	I	Abort
H2	$\overline{\text{MDTACK}}$	I	MMU data transfer acknowledge
H3	GND	—	Ground
H11	GND	—	Ground
H12	DATA17	I/O	Data 17
H13	ADDR16	I/O	Address 16
J1	$\overline{\text{EPAS}}$	O	Early physical address strobe
J2	SPARE	—	Reserved
J3	+5V	—	Power
J11	+5V	—	Power
J12	DATA18	I/O	Data 18
J13	DATA19	I/O	Data 19
K1	$\overline{\text{PAS}}$	O	Physical address strobe
K2	$\overline{\text{MSRDY}}$	I/O	Synchronous ready
K3	MDSIZE0	O	MMU data size 0
K4	$\overline{\text{EMCYCLEI}}$	O	Early MMU cycle initiate
K10	SPARE	—	Reserved
K11	GND	—	Ground
K12	ADDR19	I/O	Address 19
K13	ADDR18	I/O	Address 18
L1	$\overline{\text{MCYCLEI}}$	O	MMU cycle initiate
L2	$\overline{\text{RMW}}$	O	R & M bits write
L3	MR/$\overline{\text{W}}$	I/O	MMU read/write
L4	$\overline{\text{DS}}$	I	Data strobe
L5	+5V	—	Power
L6	GND	—	Ground
L7	+5V	—	Power

Table 13. Pin Descriptions – 125-Pin Square PGA Package (Continued)

Pin	Name	Type	Description
L8	GND	—	Ground
L9	+5V	—	Power
L10	ADDR23	I/O	Address 23
L11	DATA23	I/O	Data 23
L12	DATA21	I/O	Data 21
L13	DATA20	I/O	Data 20
M1	$\overline{\text{MFAULT}}$	I/O	MMU fault
M2	$\overline{\text{MDRDY}}$	O	MMU data ready
M3	DATA31	I/O	Data 31
M4	ADDR31	I/O	Address 31
M5	ADDR28	I/O	Address 28
M6	DATA29	I/O	Data 29
M7	CLK23	I	Clock
M8	CLK34	I	Clock
M9	DATA26	I/O	Data 26
M10	ADDR25	I/O	Address 25
M11	ADDR24	I/O	Address 24
M12	DATA22	I/O	Data 22
M13	ADDR20	I/O	Address 20
N1	MDSIZE1	O	MMU data size 1
N2	$\overline{\text{DSHAD}}$	O	Data shadow
N3	DATA30	I/O	Data 30
N4	ADDR29	I/O	Address 29
N5	ADDR30	I/O	Address 30
N6	DATA28	I/O	Data 28
N7	ADDR27	I/O	Address 27
N8	ADDR26	I/O	Address 26
N9	DATA 27	I/O	Data 27
N10	DATA25	I/O	Data 25
N11	DATA24	I/O	Data 24
N12	ADDR22	I/O	Address 22
N13	ADDR21	I/O	Address 21

Functional Groups

Table 14. Address and Data

Name	Pin	Type	Function
ADDR00—ADDR31	A4, A5, B6, A6, B7, A11, B10, B11, A10, A12, B12, A13, E12, D12, E13, F12, H13, G13, K13, K12, M13, N13, N12, L10, M11, M10, N8, N7, M5, N4, N5, M4	I/O	**Address Bus.** These pins provide a bidirectional bus for virtual (input) and physical (output) addresses. When $\overline{\text{VAD}}$ is asserted, a virtual address is received from the CPU. When $\overline{\text{PAS}}$ is asserted, a physical address exists on the bus. ADDR00 is the least significant address bit.
DATA00—DATA31	A3, B4, B5, A7, A9, B8, A8, B9, C11, D11, D10, B13, C12, C13, D13, F13, G12, H12, J12, J13, L13, L12, M12, L11, N11, N10, M9, N9, N6, M6, N3, M3	I/O	**Data Bus.** These pins provide a bidirectional bus to convey descriptors to and from the MMU. DATA00 is the least significant data bit.

Table 15. Interface Control

Name	Pin	Type	Function
$\overline{AS}$	F1	I	**Address Strobe.** Indicates a valid address on the address bus.
$\overline{CCYCLEI}$	E4	—	**CPU Cycle Initiate.** This signal is ignored by the *WE* 32101 MMU. However, it can be connected in those systems which desire upward pin compatability from a square PGA package *WE* 32101 MMU to a square PGA package *WE* 32201 MMU.
$\overline{DS}$	L4	I	**Data Strobe.** Indicates the state of the CPU data strobe to the MMU. During physical address cycles the $\overline{MDS}$ signal is used as a surrogate for the CPU data strobe.
$\overline{DSHAD}$	N2	O	**Data Shadow.** Indicates that the MMU is master of the address and data buses during miss-processing and R & M bit updates.
$\overline{EPAS}$	J1	O	**Early Physical Address Strobe.** Reserved for future use.
$\overline{MCS}$	D1	I	**MMU Chip Select.** When asserted, indicates that the MMU is operating in the peripheral mode.
$\overline{EMCYCLEI}$	K4	O	**Early MMU Cycle Initiate.** This signal is asserted by the MMU at the beginning of read cycles (miss-processing) and read-modify-write cycles (R & M updates). It is also issued with the physical address during translation and in the first wait state of a physical mode access. Note that $\overline{EMCYCLEI}$ is issued one half cycle before $\overline{MCYCLEI}$.
$\overline{MCYCLEI}$	L1	O	**MMU Cycle Initiate.** This signal is asserted at the beginning of read cycles (miss-processing), read-modify-write cycles (R & M bit updates), and the physical address during translation.
$\overline{MDRDY}$	M2	O	**MMU Data Ready.** Indicates a successful (no fault) descriptor access during R & M bit updates on miss-processing. When $\overline{DSHAD}$ is inactive, $\overline{MDRDY}$ is 3-stated.
$\overline{MDS}$	G2	O	**MMU Data Strobe.** During a read cycle this signal indicates data may be placed on the data bus. On a write cycle it indicates valid data are on the data bus. During physical address cycles this signal is an extension of the CPU data strobe.
$\overline{MDTACK}$	H2	I	**MMU Data Transfer Acknowledge.** Indicates to the MMU if wait states are inserted in an MMU memory access. If not asserted when sampled during miss-processing, R & M bit update wait states are inserted.
$\overline{MSRDY}$	K2	I/O	**Synchronous Ready.** As an input, this signal is sampled during miss-processing and during R & M bit updating. If not asserted when sampled, wait states are inserted in an MMU memory access. $\overline{MSRDY}$ is an output when the MMU is in peripheral mode and during move translated word operations.

Table 15. Interface Control (Continued)

Name	Pin	Type	Function
$\overline{PAS}$	K1	O	**Physical Address Strobe.** Indicates that a valid physical address is on the address bus.
$\overline{TCSA}$, $\overline{TCSB}$	E2, F2	I	**Translation Chip Selects.** When both are asserted, these signals instruct the MMU to perform translation functions. If only one signal is asserted, the MMU ignores the address generated by the CPU. If both signals are not asserted, then all MMU outputs are 3-stated. These signals aid multiple MMU configurations.

Table 16. MMU Status

Name	Pin	Type	Function
$\overline{CABLE}$	E1	O	**Cacheable.** Indicates the value of the **$** bit of the descriptor used for translation. $\overline{CABLE}$ is valid with $\overline{PAS}$.
$\overline{CONTIG}$	B1	O	**Contiguous Segment.** Indicates a contiguous segment is being translated. During miss-processing or R & M bit updates, $\overline{CONTIG}$ is set to logic 0.
MDSIZE0, MDSIZE1	K3, N1	O	**MMU Data Size.** These signals indicate data size is a word (MDSIZE0=0 and MDSIZE1=0) when $\overline{DSHAD}$ is asserted.
MR/$\overline{W}$	L3	I/O	**MMU Read/Write.** As an output, indicates that a read or write operation is in progress. As an input, indicates that the CPU is performing a read or write.
$\overline{RMW}$	L2	O	**R & M Bits Write.** Indicates an R & M bit update when the MMU is master of address and data buses.
SAS0—SAS3	D4, D2, B2, C2	I	**Access Status.** These signals indicate the access status. (See Table 21).
$\overline{VAD}$	C1	I	**Virtual Address.** Indicates that address is virtual. When not asserted, address is physical.
XMD0, XMD1	B3, A2	I	**Execution Level.** These signals indicate the current execution level of the CPU.

Table 17. Exceptions

Name	Pin	Type	Description
$\overline{ABORT}$	H1	I	**Abort.** Terminates MMU operation when asserted. $\overline{ABORT}$ is used on aborted CPU cache fill operations.
$\overline{MFAULT}$	M1	I/O	**MMU Fault.** As an input $\overline{MFAULT}$ indicates miss-processing or R & M bit update memory faults. $\overline{MFAULT}$ is an input when the MMU has control of the buses. As an output, $\overline{MFAULT}$ indicates a fault during translation, a descriptor-related fault during miss-processing, or R & M bit updates only after $\overline{DSHAD}$ is disabled.
$\overline{MRESET}$	A1	I	**MMU Reset.** Resets the MMU to its last known state.

Table 18. Clocks

Name	Pin	Type	Function
CLK23	M7	I	**Clock.** Connected to the CLK23 input clock of the CPU.
CLK34	M8	I	**Clock.** Connected to the CLK34 input clock of the CPU.

Table 19. Test

Name	Pin	Type	Function
$\overline{\text{HIGHZ}}$	G1	I	**Test.** 3-states all MMU output signals.

Table 20. Access Status Code Assignments

Access Status Inputs SAS3—SAS0				Access Type	MMU Behavior	Permission
3	2	1	0			
0	0	0	0	Move translated word sequence	Perform MOVTRW	Read
0	0	0	1	Support processor data write	Normal translation	Write
0	0	1	0	Autovector interrupt acknowledge	Physical mode access	—
0	0	1	1	Support processor data fetch	Normal translation	Read
0	1	0	0	Stop acknowledge	Ignore	—
0	1	0	1	Support processor broadcase	Ignore	—
0	1	1	0	Support processor status fetch	Ignore	—
0	1	1	1	Read interlocked	Normal translation	Read/write
1	0	0	0	Address fetch	Normal translation	Read
1	0	0	1	Operand fetch	Normal translation	Read
1	0	1	0	Write	Normal translation	Write
1	0	1	1	Interrupt acknowledge	Physical mode access	—
1	1	0	0	Instruction fetch after PC discontinuity	Normal translation	Execute
1	1	0	1	Instruction prefetch	Normal translation	Execute
1	1	1	0	Instruction fetch	Normal translation	Execute
1	1	1	1	No operation	Ignore	—

MMU-CPU Interconnections

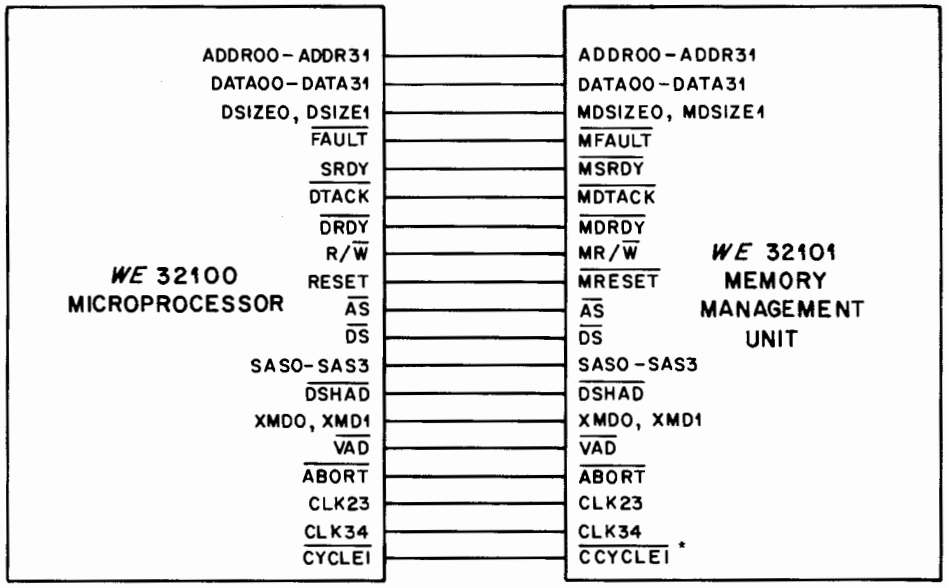
Single MMU Configuration

For a single MMU configuration, several MMU signal lines are interfaced directly to the CPU. Figure 8a shows how these signal groups are interconnected. Table 21 contains the pin-to-pin connections for the 125-pin packages.

Multiple MMU Configuration

Two select signals ($\overline{TCSA}$ and $\overline{TCSB}$) activate and deactivate the MMU. A dual MMU configuration can be achieved by decoding ADDR29 with an external inverter to control these select signals, as shown on Figure 8b. Note that only one MMU is active at a time in multiple MMU configurations. The active MMU is responsible for address translation and miss-processing. The address decoding must ensure that only one MMU is selected during the peripheral mode.

Two inverters are required to decode ADDR28 and ADDR29 for a configuration using four MMUs.



* This connection is optional.

Figure 8a. MMU-CPU Interconnections-Single MMU Configuration

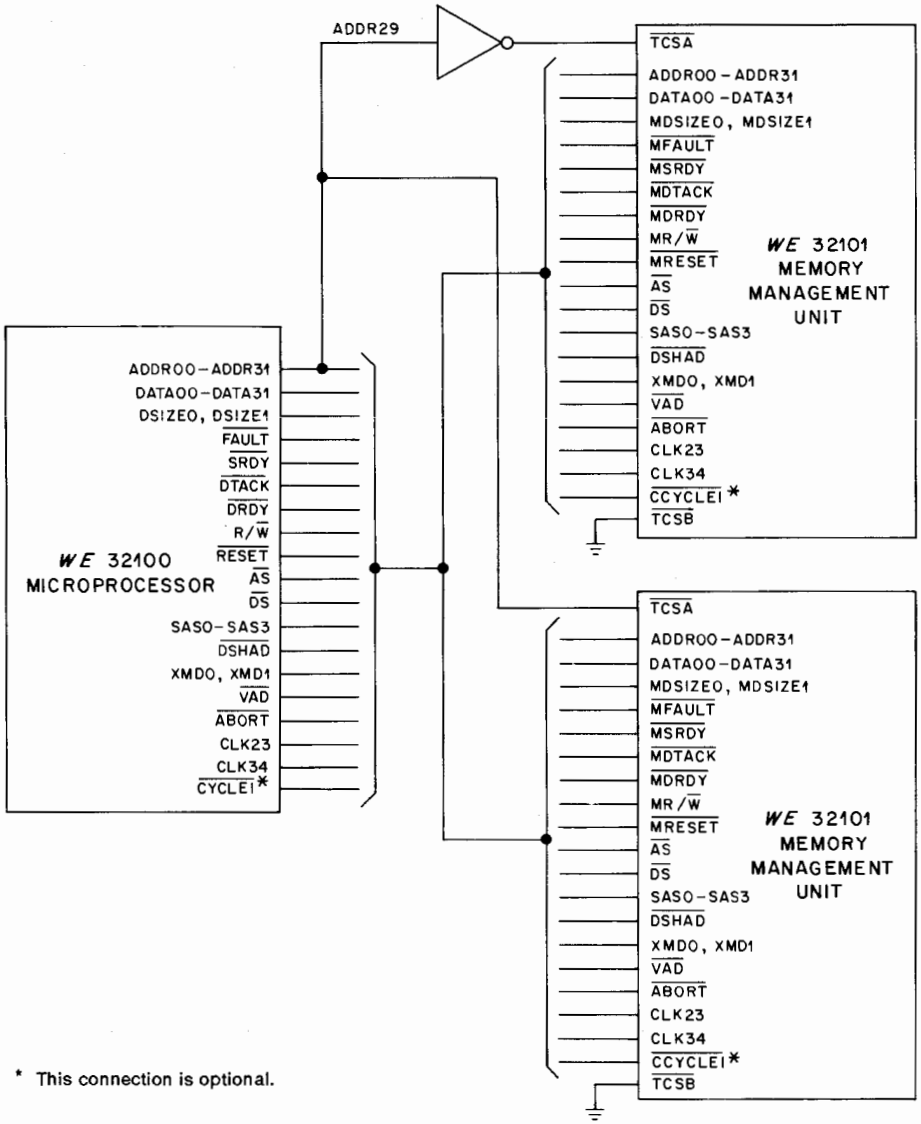


Figure 8b. MMU-CPU Interconnections-Multiple MMU Configuration

Table 21. MMU-CPU Interconnections

MMU Signal(s)	MMU Pin(s)	CPU Signal(s)	CPU Pin(s)
ADDR00—ADDR31	A4,A5,B6,A6,B7,A11, B10,B11,A10,A12,B12, A13,E12,D12,E13,F12, H13,G13,K13,K12,M13, N13,N12,L10,M11,M10, N8,N7,M5,N4,N5,M4	ADDR00—ADDR31	L10,N9,N7,M7,N10,M8, N5,M5,L6,N4,N3,L4, M1,K2,L1,H1,K1,G2, E1,C1,D2,F1,E2,D1, D3,C2,C4,A4,B5, A6,B6
DATA00—DATA31	A3,B4,B5,A7,A9,B8, A8,B9,C11,D11,D10,B13, C12,C13,D13,F13,G12, H12,J12,J13,L13,L12, M12,L11,N11,N10,M9,N9, N6,M6,N3,M3	DATA00—DATA31	K10,M11,M10,M9,N11, M6,N8,N6,M4,N2,N1, L3,K4,M3,M2,K3,L2, J2,H2,J1,G3,G1,B1, A1,A2,C3,A3,D4,B2, B3,C5,B4
MDSIZE0,MDSIZE1	K3,N1	DSIZE0,DSIZE1	B12, A12
$\overline{\text{MFAULT}}$	M1	$\overline{\text{FAULT}}$	C13
$\overline{\text{MSRDY}}$	K2	$\overline{\text{SRDY}}$	D10
$\overline{\text{MDTACK}}$	H2	$\overline{\text{DTACK}}$	E12
$\overline{\text{MDRDY}}$	M2	$\overline{\text{DRDY}}$	J13
MR/$\overline{\text{W}}$	L3	R/$\overline{\text{W}}$	B11
$\overline{\text{MRESET}}$	A1	$\overline{\text{RESET}}$	D13
$\overline{\text{AS}}$	F1	$\overline{\text{AS}}$	H12
$\overline{\text{DS}}$	L4	$\overline{\text{DS}}$	H13
SAS0—SAS3	D4,D2,B2,C2	SAS0—SAS3	B10,A13,A11,A9
$\overline{\text{DSHAD}}$	N2	$\overline{\text{DSHAD}}$	F12
XMD0,XMD1	B3,A2	XMD0,XMD1	A10,A7
$\overline{\text{VAD}}$	C1	$\overline{\text{VAD}}$	M13
$\overline{\text{ABORT}}$	H1	$\overline{\text{ABORT}}$	J12
CLK23	M7	CLK23	F11
CLK34	M8	CLK34	G11
$\overline{\text{CCYCLEI}}$*	E4	$\overline{\text{CYCLEI}}$*	G12

* This connection is optional.

Operating System Considerations

Some MMU features that support an operating system are:

- Demand paging and demand segmentation provided through the use of the P, R, and M bits contained in SDs and PDs.
- Indirect segment descriptors that allow segments to be given different access permissions, yet still be shared by different processes running at the same execution level.
- An object trap that provides a mechanism that generates a unique fault on any access to a given segment.
- Cacheable bit contained in SDs that mark segments as cacheable or not cacheable.
- Page-write fault that is used to minimize copying of pages upon system fork.
- Software bits contained in SDs and PDs that can be used by the operating system designer for any special purpose.

The operating system must initialize the MMU. Initialization involves:

- Resetting the MMU to a known state
- Defining physical memory with SDTs and PDTs
- Writing SDT addresses and length into MMU section RAMs.

Peripheral mode is used when internal MMU elements are initialized. The descriptor caches may be preloaded to avoid miss-processing; otherwise, the descriptor caches are filled during miss-processing when an SD or PD is needed. The R & M bits in the CR may be set, allowing the MMU to support operating system page- or segment-replacement algorithms.

The operating system should also set up the block-moves area of the process control block (PCB) for each process in the system. Block moves can be used to set the MMU registers when process switches occur. This causes the MMU to flush its caches.

In translation of contiguous segments, address wraparound may occur. Address wraparound occurs when the sum of the address in the SD and SOT field is greater than $(2^{32}-1)$. The operating system must eliminate this problem so that no misinterpretations can take place.

Operating system action is required when the MMU signals the CPU that a fault has occurred. The MMU detects several faults that relate to errors and places the corresponding code in the FLTCR and the virtual address in the FLTAR. Using the FLTCR and FLTAR registers, an extensive fault handler can be implemented by the operating system.

Characteristics

$V_{CC} = 5.0\ V \pm 5\%$, $V_{SS} = 0\ V$, $C_L = 130\ pF$, $T_A = 0$ to $70\ ^{\circ}C$

Timing Characteristics

All TTL timing specifications are referenced to and from 0.8 V for a low voltage and 2.0 V for a high voltage. CMOS clock references are to and from $V_{CC}/2$. All minimum and maximum values are in ns.

Table 22. Timing Characteristics

Num	Symbol	Description	Fig(s)	10 MHz		14 MHz		18 MHz	
				Min	Max	Min	Max	Min	Max
1	tSASVC34L	Access status code set-up time	14	13	—	5	—	4	—
2	tC34LSASX	Access status code hold time	14	51	—	30	—	23	—
3	tTCSLASL	Translation chip select set-up time	14	13	—	13	—	10	—
4	tASLTCSH	Translation chip select hold time	14	20	—	14	—	11	—
5	tVADHC34L	$\overline{VAD}$ set-up time	14	15	—	5	—	4	—
6	tC34LVADX	$\overline{VAD}$ hold time	14	51	—	30	—	23	—
7	tXMDVC34L	Execution mode set-up time	16	15	—	5	—	4	—
8	tC34LXMDX	Execution mode hold time	16	51	—	30	—	23	—
9	tDTALC34H	MMU data transfer acknowledge set-up time*	17	13	—	13	—	8	—
10	tMDSHDTAH	MMU data transfer acknowledge asserted	17	0	—	0	—	0	—
11	tMSRLC34L	MMU synchronous ready set-up time	17, 18	15	—	13	—	12	—

* This is an asynchronous signal. Set-up times are specified for testing and informational purposes. The set-up times guarantee recognition at the next clock edge.

Table 22. Timing Characteristics (Continued)

Num	Symbol	Description	Fig(s)	10 MHz		14 MHz		18 MHz	
				Min	Max	Min	Max	Min	Max
12	tC34LMSRH	MMU synchronous ready hold time	17,18	20	—	14	—	11	—
13	tMRWLC34L	MMU read/write set-up time	14	17	—	5	—	4	—
14	tC34LMRWX	MMU read/write hold time	14	20	—	18	—	15	—
15	tASLC34L	Address strobe set-up time	9, 14	13	—	13	—	10	—
16	tC34LASL	Address strobe hold time	14	20	—	14	—	11	—
17	tABTLC23H	MMU abort set-up time	10	13	—	12	—	9	—
18	tC23HABTH	MMU abort hold time	10	20	—	15	—	12	—
19	tMCSLC23H	MMU chip select set-up time*	14	13	—	13	—	10	—
20	tPASHMCSH	MMU chip select asserted	14	0	—	0	—	0	—
20A	tPASLMCSL	MMU chip select assertion time	—	0	—	0	—	0	—
21A	tMFTLC34H	MMU fault set-up time*	19	17	—	15	—	12	—
21B	tMFTLC34L	MMU fault set-up time**	18	15	—	16	—	12	—
22B	tC34LMFTH	MMU fault hold time**	18	25	—	18	—	14	—
25	tADDVC23L	Address set-up time; referenced from clock 23	14, 16	9	—	8	—	6	—
26	tC23LADDZ	Address hold time (input)	14, 16	25	—	23	—	19	—
27	tDATVC23L	Data set-up time – peripheral mode	14	24	—	17	—	13	—
28	tC23LDATX	Data hold time – peripheral mode	14	25	—	17	—	13	—
29	tDATVC34H	Data set-up time for MMU-initiated access	17	13	—	7	—	5	—
30	tC34HDATX	Data hold time for MMU-initiated access	17	25	—	20	—	16	—
31	tDSHC23H	Data strobe negation set-up time	13	13	—	13	—	10	—
32	tC23HDSH	Data strobe negation hold time	13	20	—	14	—	11	—
40	tC23HEPAL	Early address strobe assertion time	16, 17	—	46	—	37	—	34
41	tC23HEPAH	Early address strobe negation time	16, 17	—	50	—	42	—	34
42A	tC34HPASL	Physical address strobe assertion time	16, 17	—	36	—	33	—	29

* This is an asynchronous signal. Set-up times are specified for testing and informational purposes. The set-up times guarantee recognition at the next clock edge.

** This is a synchronous signal when asserted with $\overline{\text{DTACK}}$ or $\overline{\text{MSRDY}}$.

Table 22. Timing Characteristics (Continued)

Num	Symbol	Description	Fig(s)	10 MHz		14 MHz		18 MHz	
				Min	Max	Min	Max	Min	Max
42B	tADDVPASL	Address set-up time; referenced from physical address strobe	16, 17	88	—	66	—	45	—
42C	tADDVEPAL	Address set-up time; referenced from early physical address strobe	—	47	—	30	—	23	—
43A	tC23HPASH	Physical address strobe negation time (CPU initiated)	16	—	50	—	39	—	33
43B	tPASHADDZ	Address hold time (output)	16, 17	14	—	9	—	6	—
43C	tCLK34HPASH	Physical address strobe negation time (MMU initiated)	17	—	50	—	39	—	31
44A	tC34HMDSL	MMU data strobe assertion time (CPU read)	13, 16, 17	—	50	—	38	—	30
44B	tDATVMDSL	Data set-up time	17	25	—	15	—	11	—
44C	tCLK34LMDSL	MMU data strobe assertion time (CPU write)	14	—	50	—	38	—	30
45A	tC23HMDSH	MMU data strobe negation time (CPU initiated)	16	—	50	—	36	—	30
45B	tMDSHDATZ	Data hold time	17	14	—	14	—	11	—
45C	tMDSHPASH	Physical address strobe negation time	16, 17	–3	—	–3	—	–3	—
45D	tCLK34HMDSH	MMU data strobe negation time (MMU initiated)	17	—	50	—	36	—	31
46	tC34LCBLL	Cacheable assertion time	15, 16	—	60	—	43	—	33
47A	tC34LCBLH	Cacheable negation time	17	—	62	—	44	—	36
47B	tC34HCBLH	Cacheable negation time	16	—	62	—	44	—	34
48A	tC34LCNTL	Continuous segment assertion time	15, 16	—	50	—	40	—	35
48B	tC34HCNTL	Continuous segment negation time	16	—	62	—	44	—	36
50	tC34LMCYL	MMU cycle initiate assertion time	16, 17	—	51	—	40	—	30
50A	tC34HEMCV	Early MMU cycle initiate assertion time	16, 17	—	51	—	40	—	30

Table 22. Timing Characteristics (Continued)

Num	Symbol	Description	Fig(s)	10 MHz		14 MHz		18 MHz	
				Min	Max	Min	Max	Min	Max
51	tC34LMCYH	MMU cycle initiate negation time	16, 17	—	57	—	42	—	33
51A	tC34LEMCH	Early MMU cycle initiate negation time	16, 17	—	57	—	42	—	33
52	tC23HMSRL	MMU synchronous ready assertion time	14, 18, 19	—	46	—	36	—	30
53	tASHMSRH	Address strobe rising edge to MMU synchronous ready high impedance	14, 18, 19	—	46	—	32	—	28
54	tC23HMFTL	MMU fault assertion time	18, 19	—	50	—	39	—	29
55	tASHMFTZ	Address strobe rising edge to MMU fault high impedance	18, 19	—	45	—	33	—	28
56	tC34LRMWL	R & M bits write assertion time	17	—	48	—	39	—	30
57	tC34LRMWH	R & M bits write negation time	17	—	48	—	35	—	28
58	tC34LDSHL	Data shadow assertion time	15	—	50	—	39	—	30
59	tC34LDSHH	Data shadow negation time	17, 18, 19	—	50	—	39	—	30
60	tC34LMDZL	MMU data size assertion time	15	—	59	—	40	—	31
61	tC34LMDZH	Clock 34 low to MMU data size high impedance	17	—	62	—	44	—	37
62	tC34LMRWH	MMU read/write assertion time	15, 17	—	52	—	41	—	32
63	tC34LMRWL	MMU read/write negation time	17	—	52	—	40	—	31
64	tC34HADDV	Address assertion time	16, 17	—	42	—	37	—	33
65	tC34LADDZ	Clock 34 low to address high impedance	16, 17	—	50	—	35	—	30
66	tC34LDATV	Data assertion time	12, 13 17	—	57	—	40	—	38
67	tC34LDATZ	Clock 34 low to data high impedance	12, 13 17	—	50	—	38	—	30
68	tC34HMDRL	MMU data ready assertion time	17	—	53	—	40	—	31
69	tC34HMDRH	MMU data ready negation time	17	—	53	—	40	—	31
70	tHIGZLOUTZ	HIGHZ low to all outputs high impedance	—	—	100	—	72	—	56

Table 22. Timing Characteristics (Continued)

Num	Symbol	Description	Fig(s)	10 MHz		14 MHz		18 MHz	
				Min	Max	Min	Max	Min	Max
70A	tC23LMSOZ	Clock 23 low to strobe outputs high impedance	9, 10	—	50	—	36	—	30
70B	tHIGZOUTV	HIGHZ high to all outputs valid	—	—	100	—	70	—	54
71	tC23L1C23H2	Clock 23 rise time	11	—	4	—	4	—	4
72	tC23H2C23L1	Clock 23 fall time	11	—	4	—	4	—	4
73	tC23L1C23L1	Clock 23 period (T)	11	100	—	71.4	—	56	—
74	tC34L1C34H2	Clock 34 rise time	11	—	4	—	4	—	4
75	tC34H2C34L1	Clock 34 fall time	11	—	4	—	4	—	4
76	tC34L1C34L1	Clock 34 period (T)	11	100	—	71.4	—	56	—
77	tC23J	Clock 23 jitter	—	—	0.5	—	0.5	—	0.5
78	tC23E	Clock 23 duty cycle error	—	—	3	—	2	—	2
79	tC34J	Clock 34 jitter	—	—	0.5	—	0.5	—	0.5
80	tC34E	Clock 34 duty cycle error	—	—	3	—	2	—	2
81	tSKEW	Clock skew	—	—	3	—	2	—	2

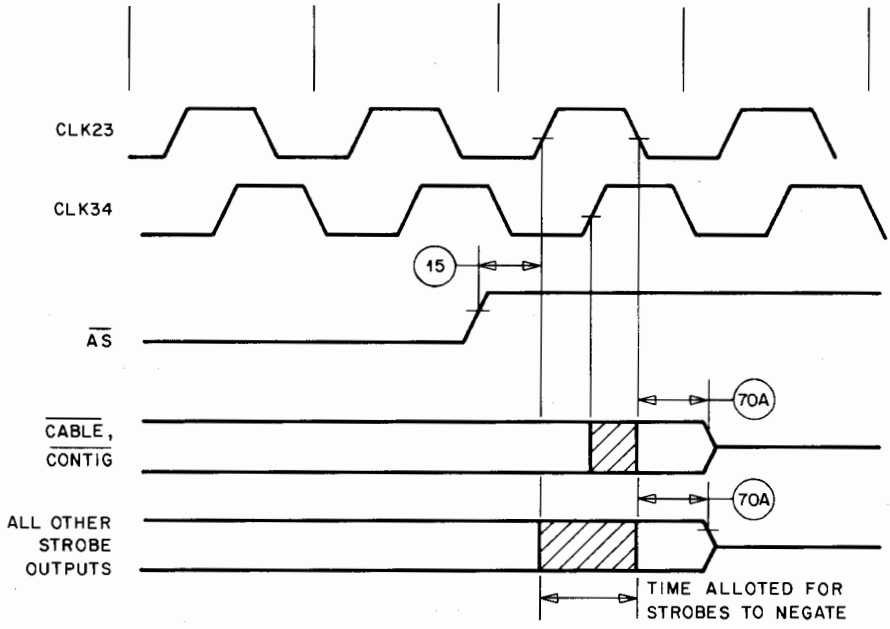


Figure 9. CPU Terminates Access

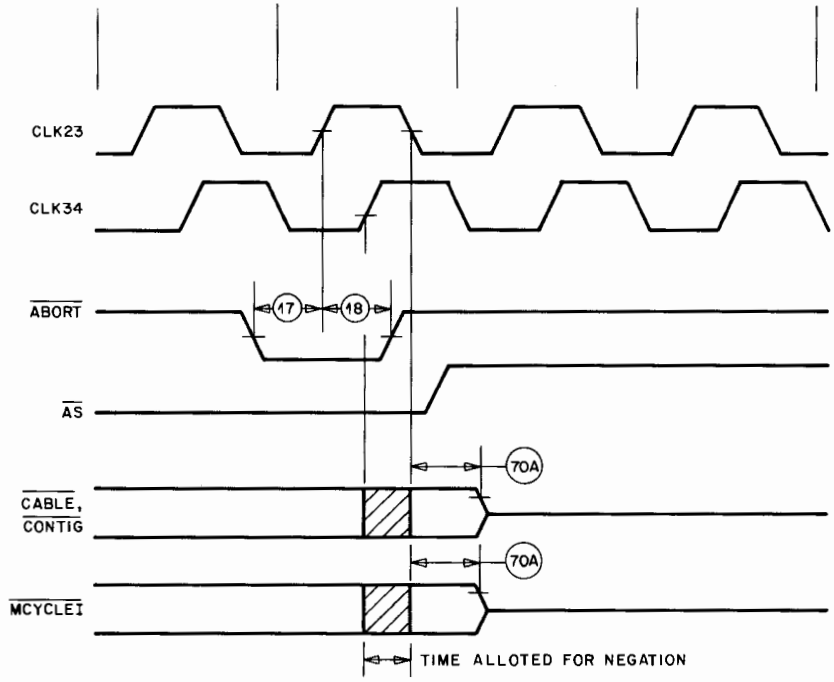
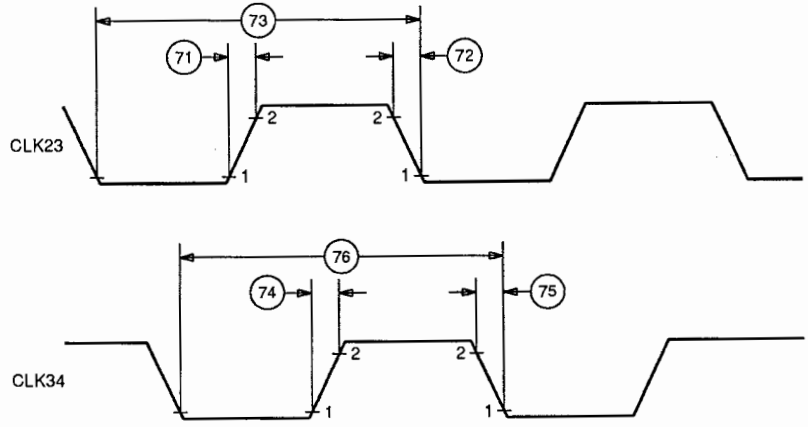


Figure 10. CPU Aborted Access



Notes:
Duty Cycle Error – The duty cycle of each clock input may deviate from 50% but should not exceed timing specification numbers 78 and 80 for CLK23 and CLK34, respectively.

Skew – For 10 MHz operation, CLK23 nominally leads CLK34 by 90° (1/4 clock period). This phase lead should not deviate more than timing specification number 81.

Jitter – The period of each clock input may deviate from its nominal value and should not exceed timing specification numbers 77 and 79 for CLK23 and CLK34, respectively.

Figure 11. Clock Inputs

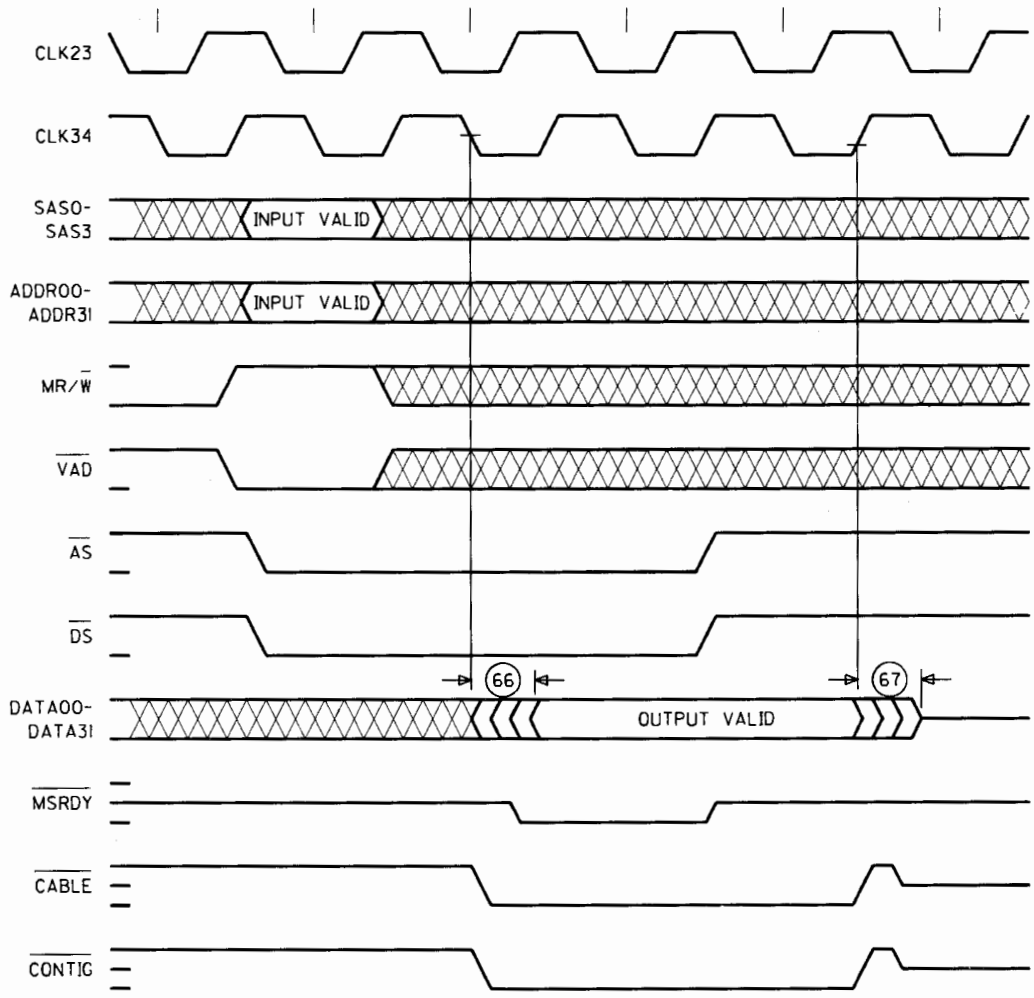


Figure 12. CPU Virtual Move Translated Word Access Cache Hit Translation

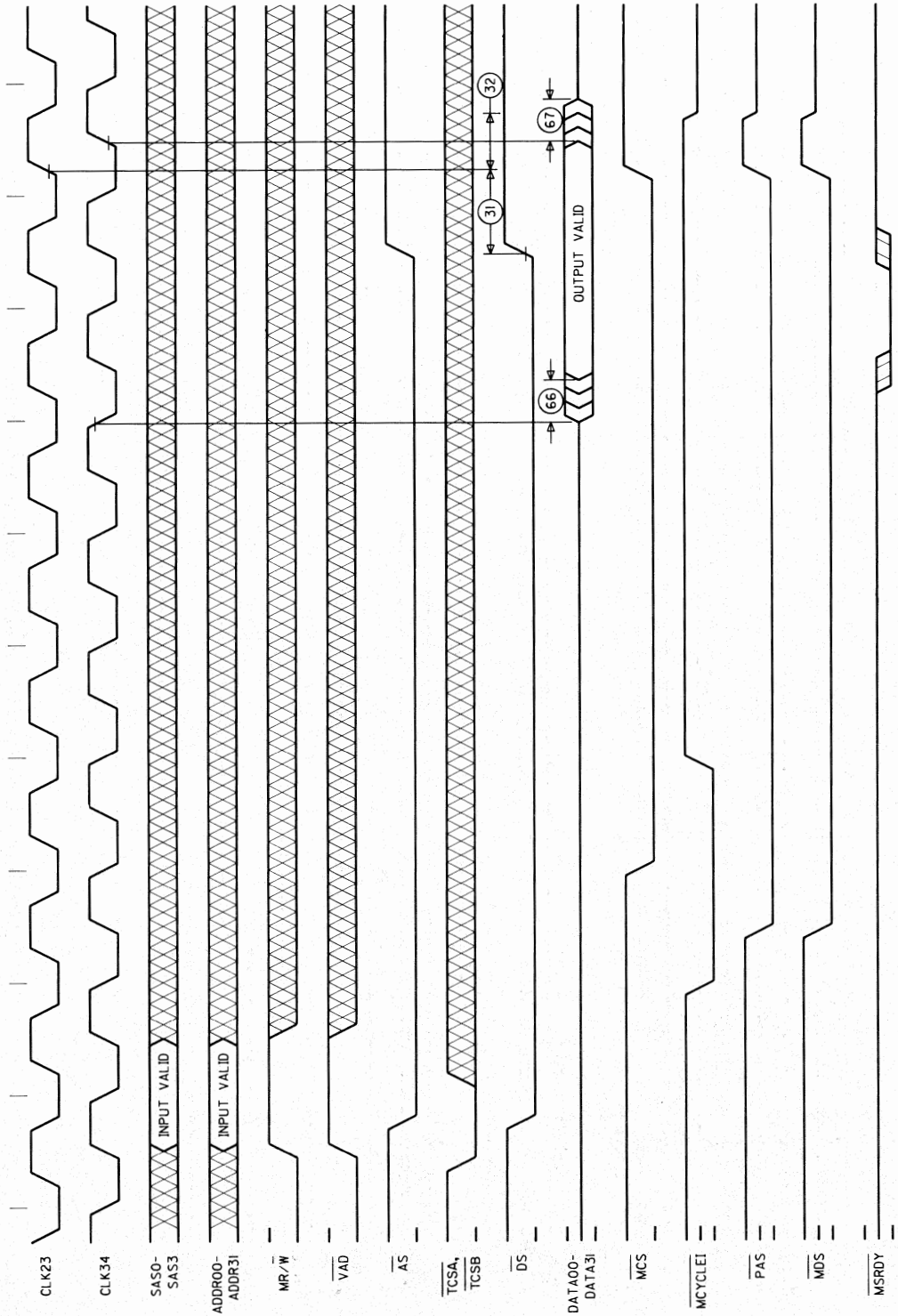


Figure 13. CPU Physical Access Peripheral Mode Read

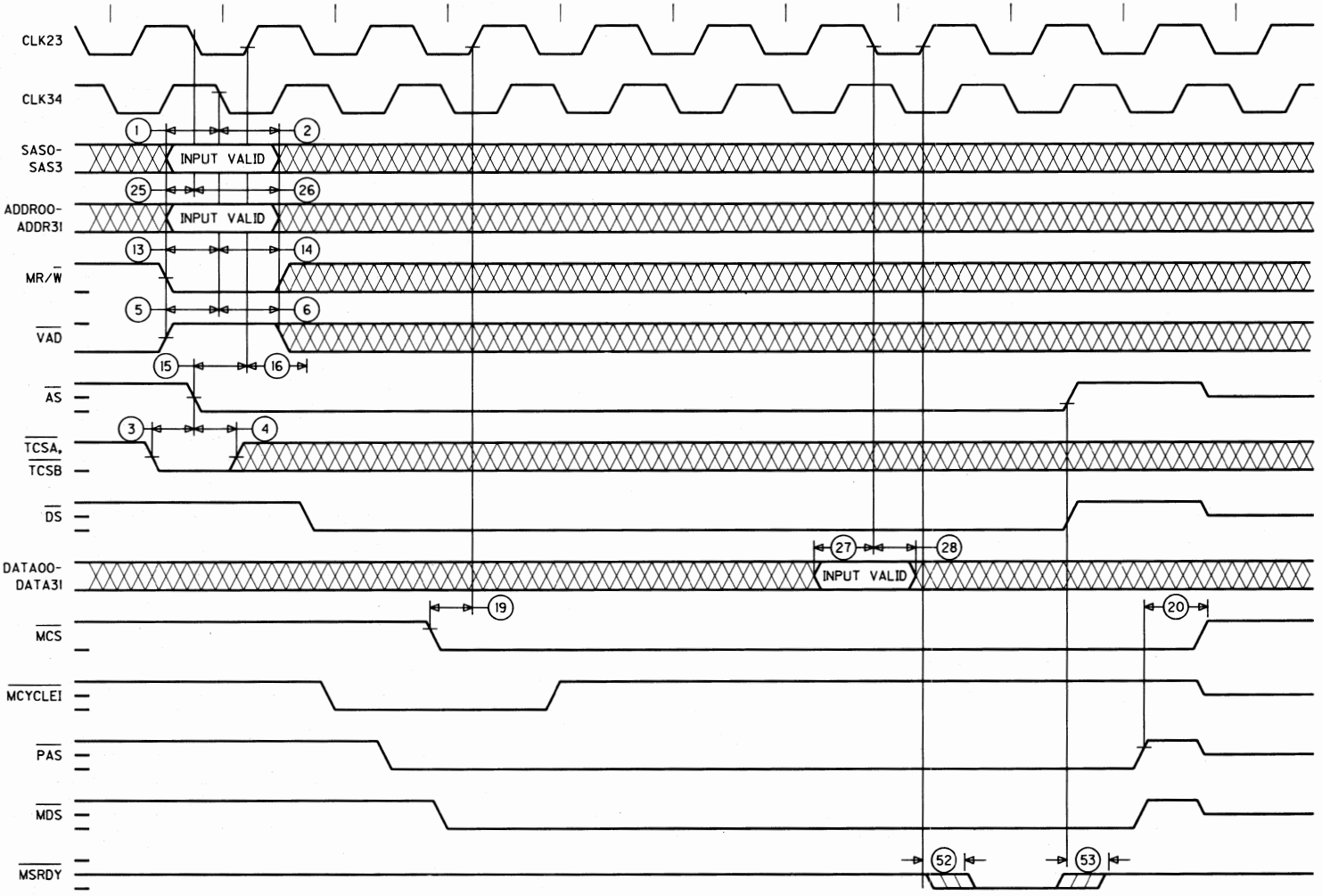


Figure 14. CPU Physical Access Peripheral Mode Write

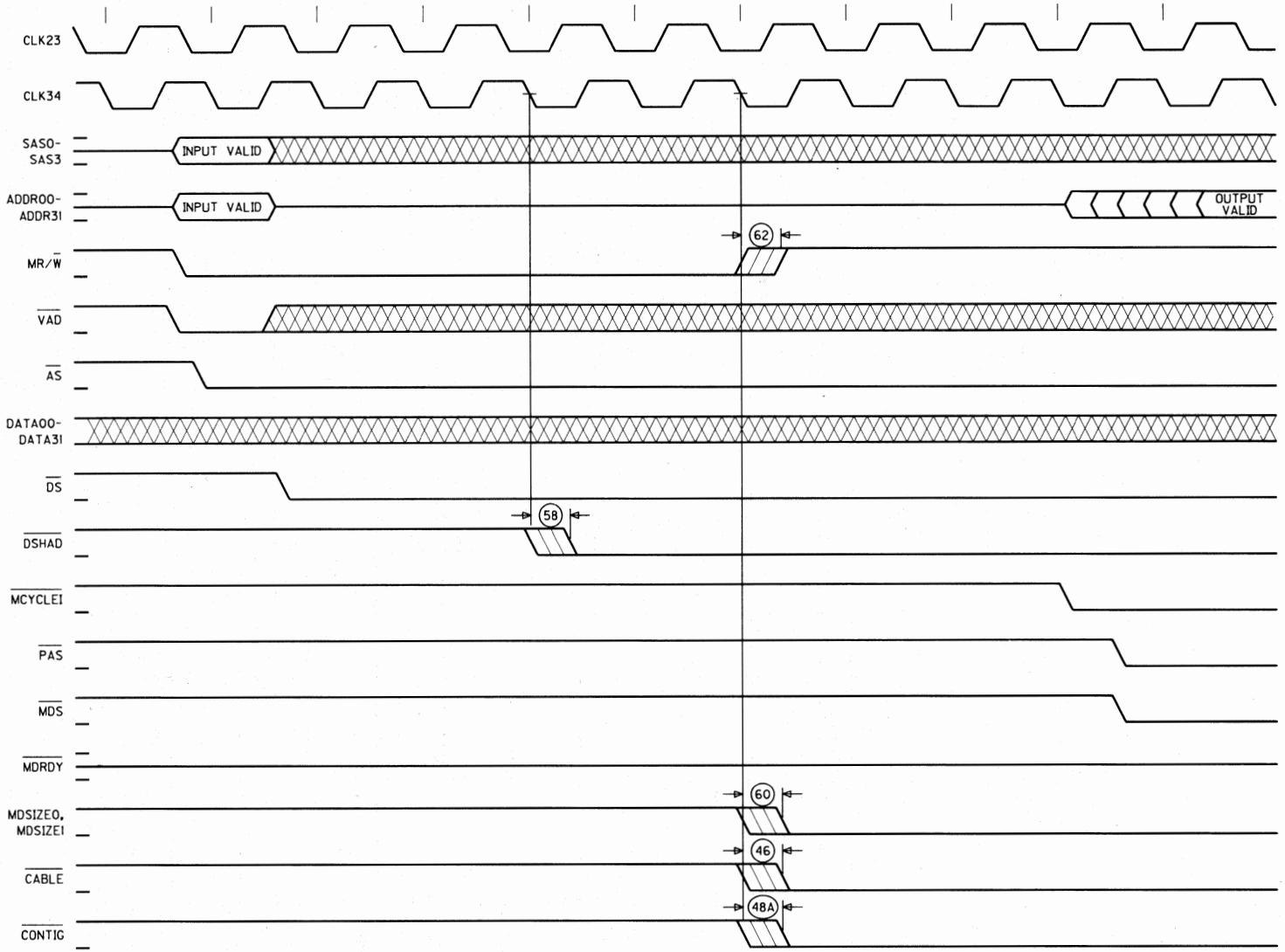


Figure 15. Write Access Miss-Process Start (Segment Miss)

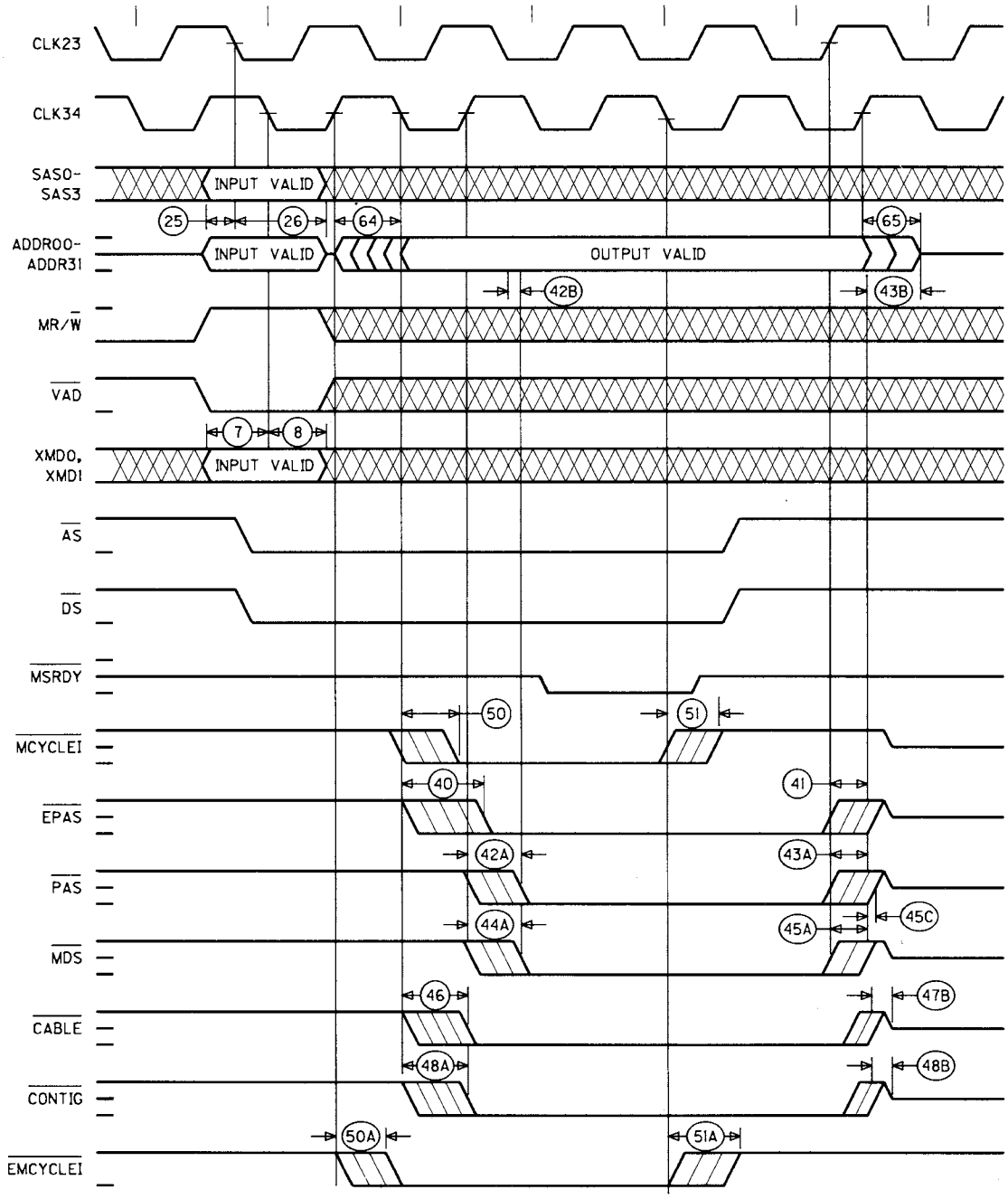


Figure 16. Read Access Cache Hit Translation

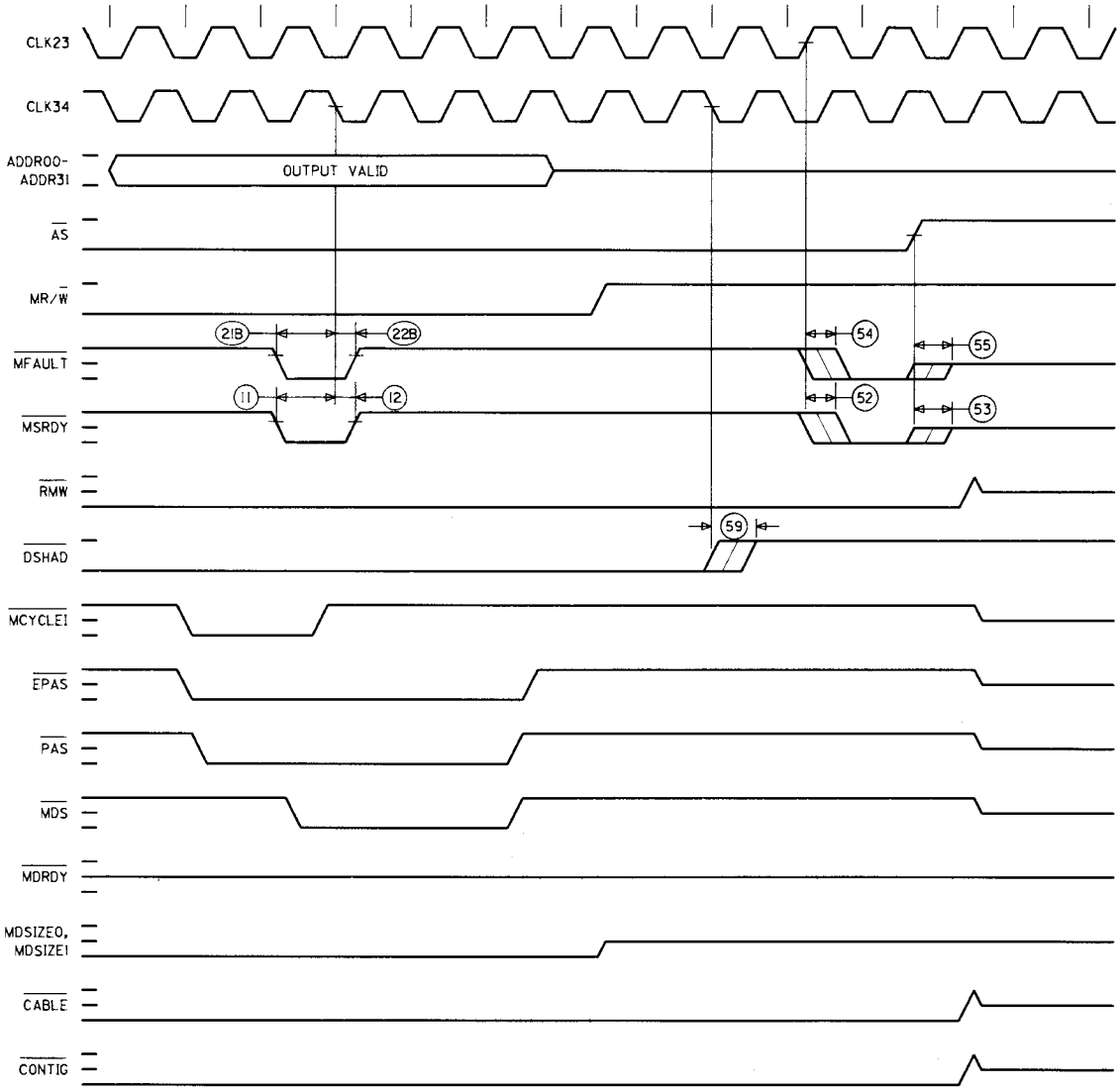


Figure 17. MMU Ready-Modify-Write Access

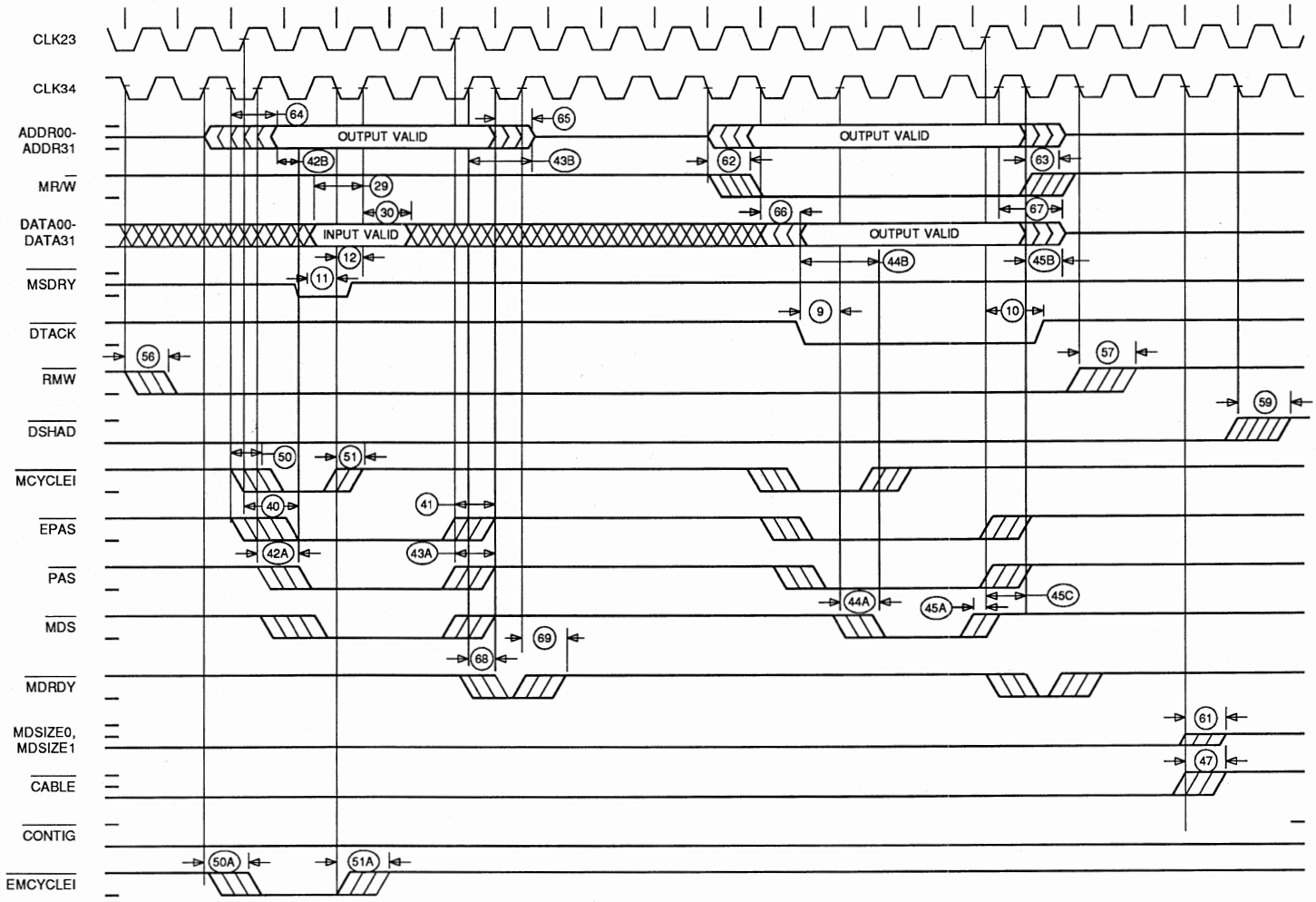


Figure 18. MMU Write Access With Synchronous Memory Faults

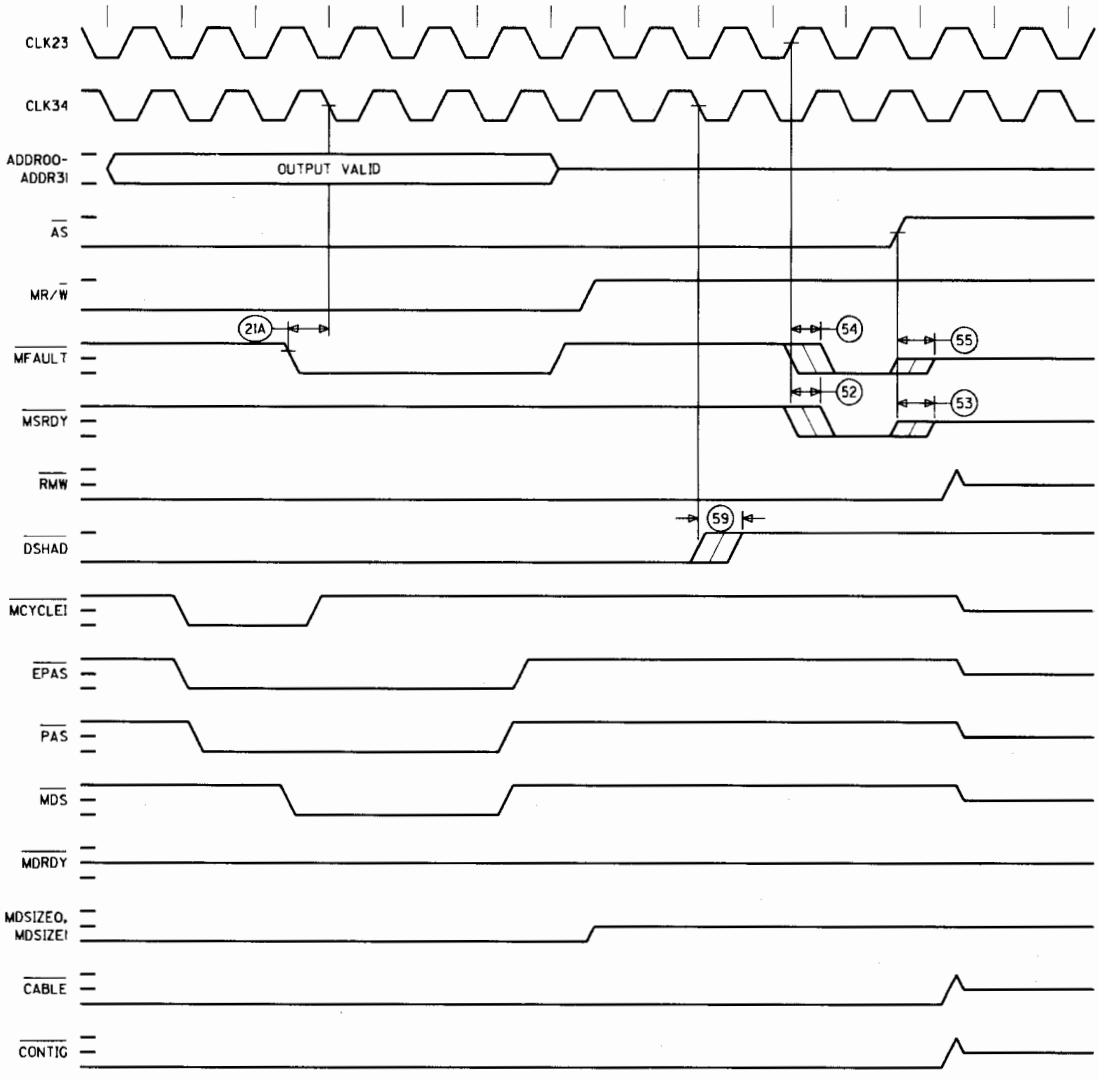


Figure 19. MMU Write Access With Asynchronous Memory Fault

Electrical Characteristics

Inputs

All inputs except the CMOS clock inputs are TTL-compatible.

Table 23. DC Input Parameters

Inputs		Min	Max	Unit
TTL input voltage	high-level	2.0	V_{CC} + 0.5	V
	low-level	–0.5	0.8	V
CMOS clocks input voltage	high-level	V_{CC} – 1.3	V_{CC} + 0.5	V
	low-level	0	0.8	V
TTL input loading current ($2.0\ V \leq V_{IH} \leq V_{CC}$)	high-level	0	0.01	mA
TTL input loading current ($0\ V \leq V_{IL} \leq 0.8\ V$)	low-level	–0.01	0	mA
CMOS clocks input loading current ($V_{CC} - 1.3\ V \leq V_{IH} \leq V_{CC}$)	high-level	0	0.01	mA
CMOS clocks input loading current ($0 \leq V_{IL} \leq 0.8$)	low-level	–0.01	0	mA

Outputs

Two classes of outputs are provided that support TTL input voltage levels. These classes are defined as:

Class 1: This class is capable of driving one TTL load or 8 PNP Schottky TTL loads and has current allowance for an external holding resistor employed in 3-state buffers. The minimum holding resistor value is 2.7 kΩ, with the exception that no holding resistors are needed for ADDR00—ADDR31.

Class 2: The signal in this class is an open drain device used for wired-logic operations, allowing more than one device to drive a node without conflict. An external resistor is required to pull the signal high. The minimum pull-up resistor value is 680 Ω.

The following is a list of the outputs assigned to Classes 1 and 2:

Class 1	Class 2
ADDR00—ADDR31	$\overline{\text{MFAULT}}$
$\overline{\text{CABLE}}$	$\overline{\text{MSRDY}}$
$\overline{\text{CONTIG}}$	
DATA00—DATA31	
$\overline{\text{DSHAD}}$	
$\overline{\text{EPAS}}$	
$\overline{\text{MCYCLEI}}$	
$\overline{\text{MDRDY}}$	
$\overline{\text{MDS}}$	
MDSIZE0, MDSIZE1	
$\overline{\text{MPAS}}$	
MR/$\overline{\text{W}}$	
$\overline{\text{RMW}}$	
$\overline{\text{EMCYCLEI}}$	

Table 24. DC Output Parameters

Outputs		Min	Max	Unit
Output sink current (I_{OL}) ($V_{OL} \leq 0.4$ V)	Class 1	—	5.5	mA
	Class 2	—	12.0	mA
Output source current (I_{OH}) ($V_{OH} \geq 2.4$ V)	Class 1	—	–2.0	mA
	Class 2	—	–0.01 open drain	mA
Output logic levels*	high-level	2.4	—	V
	low-level	—	0.4	V

* Referenced to system ground (GND).

Operating Conditions

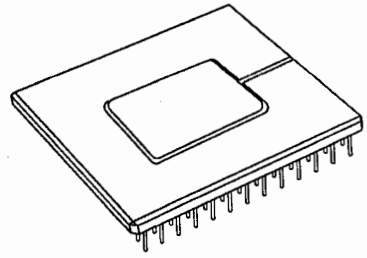
Table 25. DC Operating Conditions

Parameter		Symbol	Min	Nom	Max	Unit
Supply voltage		V_{CC}	4.75	5.00	5.25	V
Input load capacitance	TTL inputs	C_{IL}	—	—	12	pF
	CMOS clocks		—	—	7	pF
Total output load capacitance	Class 1	C_L	—	—	130	pF
	Class 2		—	—	130	pF
Ambient temperature at the microprocessor pins		T_A	0	—	70	°C
Humidity range		—	5%	—	95%	—
Power dissipation	at 10 MHz	PD	—	—	0.20	W
	at 14 MHz		—	—	0.28	W
	at 18 MHz		—	—	0.30	W
Operating frequency		F	—	—	18	MHz
Capacitive derating factor ($25\ pF \leq C_L \leq 225\ pF$)	at 10 MHz	dt/dc	—	0.1	—	ns/pF
	at 14 MHz		—	0.1	—	ns/pF
	at 18 MHz		—	.08	—	ns/pF

WE® 32201 Memory Management Unit

Description

The *WE* 32201 Memory Management Unit (MMU) is a 32-bit bus-structured device that provides logical-to-physical address translation, memory organization, control, and access protection for the *WE* 32200 Microprocessor. It is implemented in 1-micron, CMOS technology. The MMU performs address translation by mapping virtual memory addresses to physical memory addresses, which allows 2^{32} bytes of virtual memory and up to 2^{32} bytes of physical memory per process. It supports both demand paged and demand segmented virtual memory systems. The MMU also allows the use of shared segments for intertask communication. Access privileges for each segment provide system protection. The *WE* 32201 Memory Management Unit comes with an on-chip 4 Kbyte, 2-way, set-associative instruction/data cache that returns data with zero wait states on *WE* 32200 CPU virtual and physical memory accesses. Transparent data cache miss-processing and zero wait state hits significantly reduce overall system access wait states. The physical cache stores data of multiple processes and performs bus monitoring to maintain cache integrity. The *WE* 32201 Memory Management Unit is available in 24 MHz and higher frequency versions; requires a single 5 V supply; and is available in a 133-pin square, ceramic pin grid array (PGA) package.



Features

- Manages mapping of up to 4 Gbytes of virtual address space and up to 4 Gbytes of physical address space
- Paged and contiguous segmentation support
- On-chip, fully associative CAM-based, 64-entry descriptor cache
- On-chip cache miss-processing
- On-chip 4 Kbyte instruction/data cache
- Four types of access protection at four execution levels
- Transparent multiple-context support
- Flexible translation probe
- 1-1/2 cycle virtual to physical address translation
- Variable page sizes
- Hardware support for *UNIX* System demand paging with automatic R/M bit update
- Shared segments managed by indirect segment descriptors
- Extensive fault detection and resolution capability
- Low-power, 1-micron, CMOS technology

Description 183
Features 183
User Information 184
System Design 184
Memory Segmentation 184
Virtual Address Fields 184
MMU-CPU Interconnections 185
Single MMU Configuration 185
Programming 186
Descriptors 186
Segment Descriptors 186
Access Permission Field (Acc) 187
Page Descriptors 188
MMU Internal Elements 189
Section RAMs 189
Caches 190
MMU Registers 193
Data Cache Features 196
Data Cache Organization 196
Data Cache Accesses 196
Peripheral Mode Access 197
Memory-Mapped Peripheral Mode 197
Faults 198
Translation Probe 198
Pin Descriptions 202
Numerical Order 202
Functional Groups 204
Characteristics 208
Timing Characteristics 208
Timing Diagrams 212
Electrical Characteristics 221
Inputs 221
Outputs 221
Operating Conditions 222

User Information

System Design

The *WE* 32201 Memory Management Unit (MMU) is required when virtual memory storage is used for a system containing the *WE* 32200 Microprocessor.

Memory Segmentation

The MMU divides the virtual address space into four sections which in turn may be subdivided into as many as 8K segments per section. The segments are *paged* and are mapped into the physical address space by the MMU. A paged segment can consist of up to sixty-four 2 Kbyte pages. The following access permissions are associated with segments (a separate value may be assigned for each execution level):

- Execute only
- Read/execute
- Read/write/execute
- No access

Virtual Address Fields

The MMU requires the division of virtual addresses into the following four fields: a section identifier (SID) field which specifies the section of virtual address space; a segment select (SSL) field which specifies the segment within the section; a page select (PSL) field which specifies the page in the segment; and a page offset (POT) field which specifies the byte in the page. The formats of a virtual address in a paged segment are shown on Figures 1 through 3.

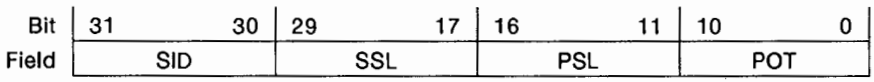


Figure 1. Virtual Address Fields for 2K Page Size

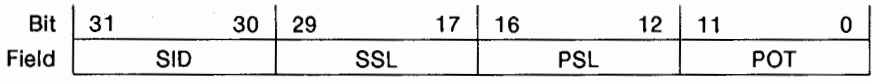


Figure 2. Virtual Address Fields for 4K Page Size

Bit	31 — 30	29 — 17	16 — 13	12 — 0
Field	SID	SSL	PSL	POT

Figure 3. Virtual Address Fields for 8K Page Size

MMU-CPU Interconnections

Single MMU Configuration. For a single MMU configuration, several MMU signal lines are interfaced directly to the CPU. Figure 4 shows how these signal groups are interconnected. Table 1 contains the pin-to-pin connections for the 133-pin PGA package.

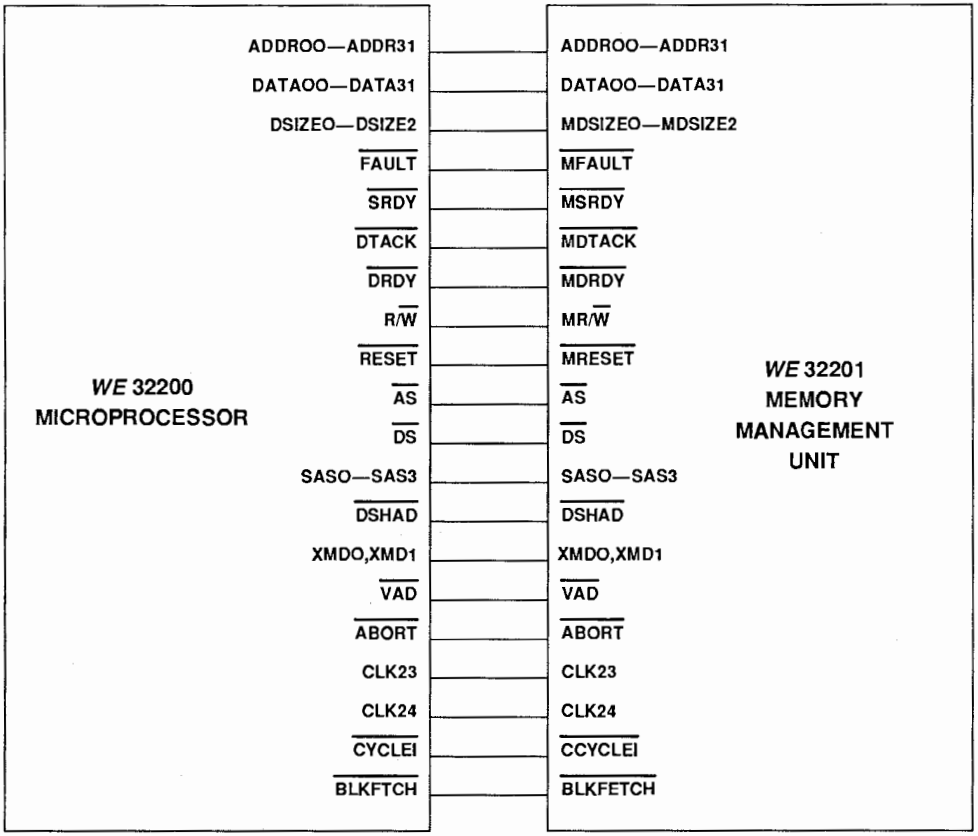


Figure 4. Single MMU-CPU Interconnections

Table 1. MMU-CPU Interconnections

MMU Signal(s)	MMU Pin(s)	CPU Signal(s)	CPU Pin(s)
ADDR00—ADDR31	A4,A5,B6,A6,B7,A11, B10,B11,A10,A12,B12, A13,E12,D12,E13,F12, H13,G13,K13,K12,M13, N13,N12,L10,M11,M10, N8,N7,M5,N4,N5,M4	ADDR00—ADDR31	L10,N9,N7,M7,N10, M8,N5,L6,N4,N3, L4,M1,K2,L1,H1,K1, G2,E1,C1,D2,F1,E2, D1,D3,C2,C4,A4,B5, A6,B6
DATA00—DATA31	A3,B4,B5,A7,A9,B8, A8,B9,C11,D11,D10, B13,C12,C13,D13,F13, G12,H12,J12,J13,L13, L12,M12,L11,N11,N10, M9,N9,N6,M6,N3,M3	DATA00—DATA31	K10,M11,M10,M9, N11,M6,N8,N6,M4, N2,N1,L3,K4,M3,M2, K3,L2,J2,H2,J1,G3, G1,B1,A1,A2,C3,A3, D4,B2,B3,C5,B4
MDSIZE0—MDSIZE2	K3,N1,J2	DSIZE0—DSIZE2	B12,A12,D9
$\overline{\text{MFAULT}}$	M1	$\overline{\text{FAULT}}$	C13
$\overline{\text{MSRDY}}$	K2	$\overline{\text{SRDY}}$	D10
$\overline{\text{MDTACK}}$	H2	$\overline{\text{DTACK}}$	E12
$\overline{\text{MDRDY}}$	M2	$\overline{\text{DRDY}}$	J13
MR/$\overline{\text{W}}$	L3	R/$\overline{\text{W}}$	B11
$\overline{\text{MRESET}}$	A1	$\overline{\text{RESET}}$	D13
$\overline{\text{AS}}$	F1	$\overline{\text{AS}}$	H12
$\overline{\text{DS}}$	L4	$\overline{\text{DS}}$	H13
SAS0—SAS3	D4,D2,B2,C2	SAS0—SAS3	B10,A13,A11,A9
$\overline{\text{DSHAD}}$	N2	$\overline{\text{DSHAD}}$	F12
XMD0,XMD1	B3,A2	XMD0,XMD1	A10,A7
$\overline{\text{VAD}}$	C1	$\overline{\text{VAD}}$	M13
$\overline{\text{ABORT}}$	H1	$\overline{\text{ABORT}}$	J12
CLK23	M7	CLK23	F11
CLK34	M8	CLK34	G11
$\overline{\text{CCYCLEI}}$	E4	$\overline{\text{CYCLEI}}$	G12
$\overline{\text{BLKFETCH}}$	K10	$\overline{\text{BLKFTCH}}$	E13

Programming

Descriptors

The virtual memory space is divided into four 1 Gbyte areas called *sections*. Each section may consist of up to 8K *segments*, where each segment is a maximum of 128K bytes long. Segments consist of *pages* which may each be 2K, 4K, or 8K bytes long. The MMU can support one page size at a time, the value being set in the configuration register. Since segments are a multiple of pages, they always start on a page boundary.

Using descriptors that contain the information necessary for segment and page mapping, the MMU performs address translation. The MMU has two types of descriptors — segment descriptors (SDs) for mapping paged segments, and page descriptors (PDs) for mapping pages.

Segment Descriptors. A segment descriptor contains an address that points to the base address of the page descriptor table. The SDs

for each of the four sections of virtual memory are located in physical memory in segment descriptor tables (SDTs). There is one SDT associated with each section. The SD entry in the SDT is 8 bytes in length and occupies two successive words in physical memory. The SD format and a description of each field are given in Tables 3 and 4.

Access Permission Field (Acc). The access permission field contains a 2-bit code defining the type of access permission associated with each execution level (kernel, executive, supervisor, and user). These codes are defined in Table 2. The MMU uses this field during address translations. Its format is shown on Figure 5.

The MMU checks the type of access requested by the CPU and determines which permissions are needed to allow the access. If the permissions do not allow the access, an access fault occurs.

Table 2. Access Permission Codes

Value	Permission	Description
00	NA	No access
01	EO	Execute only
10	RE	Read/execute
11	RWE	Read/write/execute

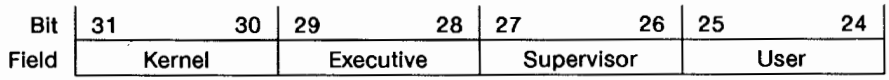


Figure 5. Access Permission Field Format

Table 3. First Word of a Segment Descriptor Field

Bit	31 24	23 18	17 8	7	6	5	4	3	2	1	0
Field	Acc	MaxOff	Res	I	V	R	$	C	M	P	—

Bit	Field	Contents	Description
0	P	Present	Specifies if a segment is present in physical memory. If P=1, segment is present; if P=0, segment is not present.
1	M	Modified	When M=1, the segment is modified via a write access.
2	C	Contiguous	Specifies if the segment is paged (C=0), or contiguous (C=1).
3	$	Cacheable	Determines state of $\overline{\text{CABLE}}$ during translation. (If $=0, then $\overline{\text{CABLE}}$=1; if $=1, then $\overline{\text{CABLE}}$=0).
4	Res	Reserved	This field is reserved for future use. Must be cleared (0).
5	R	Referenced	When R=1, the segment is referenced via a read access.
6	V	Valid	If V=1, SD is valid. If V=0, SD is not valid.
7	I	Indirection	If this bit is set, the other fields in this descriptor are ignored except as noted below. The access rights are used as the access rights for the indirect SD. The second word of this SD is a pointer to a new SD which is fetched and used as the new SD. If V=0 for the SD, then an invalid SD fault is signaled. The V bit is checked before the I bit. At most, there can be 2 levels of indirection (i.e., 3 SDs in a row with their I bits set will cause a fault).

Table 3. First Word of a Segment Descriptor Field (Continued)

Bit	Field	Contents	Description
8—17	Res	Reserved	Reserved for future use. Must always be cleared.
18—23	Max Off	Maximum offset	Used to calculate the maximum offset from the start of the segment that a virtual address may specify. Used for checking segment length.
24—31	Acc	Access permissions	Specifies the access type and permissions for each execution level (see Access Permission Field).

Table 4. Second Word of a Segment Descriptor Field

Bit	31 — 3	2 — 0
Field	Address	Soft

Bit	Field	Contents	Description
0—2	Soft	Software	Reserved for software use. The MMU does not change the value of this field at any time.
3—31	Address	Address	Pointer to page descriptor table for paged segments. Pointer to physical address of another SD when indirection feature is used.

Page Descriptors. A page descriptor contains a page base address that is concatenated with a page offset (POT) from the virtual address (VA) to form the physical address.

The PDs for each segment are located in physical memory in page descriptor tables (PDTs), and there is one PDT associated with each paged segment. The PD entry in the PDT is 4 bytes in length. The PD format and a description of each field are shown in Table 5.

The MMU minimizes address translation times by storing a number of page descriptors in its on-chip page descriptor cache. It fetches these descriptors from physical memory, as needed, without CPU intervention. Also, the MMU contains other registers and data structures that support the operating system.

Table 5. Page Descriptor Field

Bit	31 — 11	10 — 8	7 — 6	5	4	3 — 2	1	0
Field	Page Address	Soft	Res	R	W	Res	M	P

Bit	Field	Contents	Description
0	P	Present	Specifies if the page is present in primary memory. If P=1, page is present; if P=0, page is absent.
1	M	Modified	When M=1, the page associated with the descriptor has been modified.
2—3	Res	Reserved	Reserved for future use. Must always be cleared.
4	W	Fault on write	If W=1, a page-write fault occurs. If W=0, translation continues normally. This bit is not examined during miss-processing.

Table 5. Page Descriptor Field (Continued)

Bit	Field	Contents	Description
5	R	Referenced	If R=1, the page associated with the descriptor has been referenced.
6—7	Res	Reserved	Reserved for future use. Must always be cleared.
8—10	Soft	Software	Reserved for software use. The MMU does not change the value of this field at any time.
11—31	Page Address	Page address	Physical address of page. For 4 Kbyte pages, bit 11 will not be used for MMU operations. For 8 Kbyte pages, bits 11 and 12 will not be used for MMU operations. These bits, however, are considered part of the physical address and will be stored in the PDC.

MMU Internal Elements

This section describes the internal objects of the MMU, i.e., the structures that are not directly visible to the MMU user.

Section RAMs

The MMU contains two RAM areas called Section RAM A (SRAMA) and Section RAM B (SRAMB). Both of these areas contain four 32-bit words that describe the base address of the SDT for each section (contained in SRAMA) and the length (i.e., number of entries) of the SDT (contained in SRAMB) for each section.

SRAMA entries are used during miss-processing (a segment descriptor needed for address translation is not present in the MMU descriptor cache) to access descriptor tables in memory to fetch the necessary descriptors into the descriptor caches. SRAMB entries contain one less than the number of segments in each section and are used for checking the SDT bounds during miss-processing.

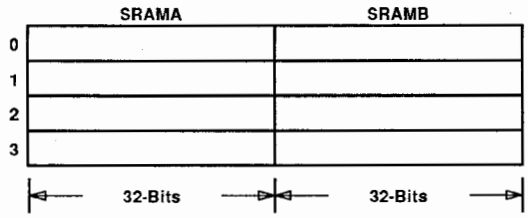


Figure 6. SRAM Organization

Table 6. SRAMA Entry Format

Bit	31 — 3	2 — 0
Field	SDT Addr	Res

Bit	Field	Contents	Description
0—2	Res	Reserved	This field is reserved for future use. It must contain zeros.
3—31	SDT Addr	SDT address	Address of the beginning of the segment descriptor table for the section.

Table 7. SRAMB Entry Format

Bit	31 — 23	22 — 10	9 — 0
Field	Res	SDT Len	Res

Bit	Field	Contents	Description
0—9	Res	Reserved	This field is reserved for future use. It must contain zeros.
10—22	SDT Len	SDT length	This field describes how many segments compose the section. The value specified in the field is: number of segments in section – 1.
23—31	Res	Reserved	This field is reserved for future use. It must contain zeros.

Caches

ID Number Cache (IDNC). The ID number cache is used by the MMU to transparently assign ID numbers to each section of the MMU. The cache is organized in a 16-entry, fully associative configuration. The tag portion of the cache consists of the upper 29 bits of the address which points to the segment table for the section. A V bit is also associated with the tag of each IDNC entry. When set, this bit indicates that the entry is valid. Upon reset of the MMU, all V bits are cleared. The data portion of the cache is a fixed value corresponding to the location of the entry within the cache. This value is used as the ID Number (IDN) for the SDT address cached at that location. Figure 7 shows the structure of the IDNC.

When read in peripheral mode, an entry in the IDNC has the format shown in Table 8. The IDNC is not writeable in peripheral mode. The IDNC is used in multiple context mode only.

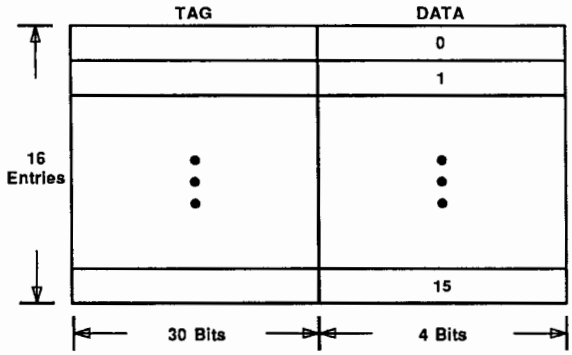


Figure 7. IDNC Organization

Table 8. IDNC Format

Bit	31 — 3	2 — 1	0
Field	TAG	Res	U

Bit	Field	Contents	Description
0	U	Useable	This bit indicates that the entry is useable or valid.
1—2	Res	Reserved	These bits are reserved and contain zeroes when read.
3—31	TAG	Tag	These are the upper 29 bits of the address which point to the segment table for the section.

Current ID Number Registers (CIDNR). The CIDNRs contain the four current IDNs (4 bits each). Whenever an SRAMA entry is updated, the corresponding CIDNR is also updated. The CIDNR is used during translation to select the IDN to be used in the page descriptor cache tag lookup. They are indexed using the SID field of the VA. When read in peripheral mode, an entry in the CIDNR has the format shown in Table 9.

The CIDNR is not writeable in peripheral mode. The CIDNR is used in multiple context mode only.

Table 9. CIDNR Format

Bit	31 — 4	3 — 0
Field	Res	IDN

Bit	Field	Contents	Description
0—3	IDN	ID number	This field contains the current ID number for the section.
4—31	Res	Reserved	These bits are reserved for future use. Their value is undefined.

Segment Descriptor Cache (SDC). The Segment Descriptor Cache contains segment descriptors to reduce miss-processing overhead. The cache is organized in an 8-entry, directly mapped configuration. Bits 19, 18, and 17 of the VA are used as the index into the cache. When an SRAMA entry is written, all entries of the corresponding section are flushed from the SDC. Figure 8 shows the structure of the SDC. The TAG field consists of the VADDR and the G bit.

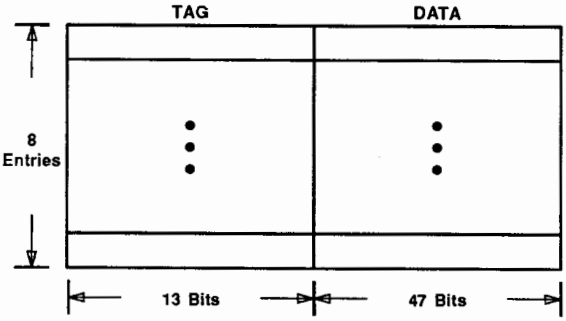


Figure 8. SDC Organization

Figures 9 and 10 show the format of the entries in the SDC when accessed in peripheral mode.

Bit	63 35	34	33	32
Field	Addr	$	C	G

Figure 9. SDC Entry Format (Word 0)

Bit	31 24	23	22	21	20 15	14 12	11 0
Field	Acc	R	M	Res	MaxOff	Res	VAddr

Figure 10. SDC Entry Format (Word 1)

Page Descriptor Cache (PDC). The page descriptor cache contains page descriptors for fast translation from virtual to physical addresses. The cache is organized in a 64-entry, fully associative configuration. Figure 11 shows the structure of the PDC.

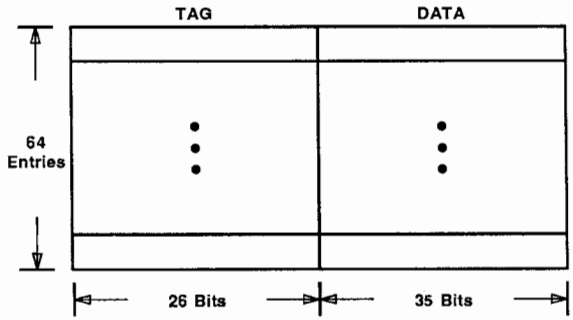


Figure 11. PDC Organization

In multiple context mode, for all operations except single-entry flushing, the TAG field consists of the G and IDN fields, and bits 37—55 of the VAddr field (VA11—VA29). For single-entry flushing, the TAG field consists of the G, IDN, and VAddr fields (VA11—VA31). For 4 Kbyte pages, bit 37 of the VAddr is not used as part of the TAG; for 8 Kbyte pages, bits 37 and 38 are not used as part of the TAG.

In single context mode, for all operations, the TAG field consists of the G and VAddr fields (VA11—VA31). In all cases, for 4 Kbyte pages, VAddr37 (VA11) is not used as part of the TAG. For 8 Kbyte pages, VAddr37 and VAddr38 (VA11 and VA12) are not used as part of the TAG.

Figures 12 and 13 show the format of the entries in the PDC when accessed in peripheral mode.

The PAddr, R, M, and W bits are from the PD and the $ bit and the Acc field are from the SD. The G bit, when set, indicates a valid entry in the PDC. The IDN cached is the IDN that was currently selected from the CIDNRs. The U (used) bit describes if the entry was used

recently. All G bits and U bits are cleared upon reset. An entry's G and U bits are also cleared if the entry was flushed from the PDC. The VAddr consists of bits 11—31 of the VA.

MMU Registers

The MMU contains five 32-bit registers that provide information pertaining to the present state of the MMU.

- **Fault Code Register (FLTCR).** The FLTCR is loaded whenever a fault occurs during an MMU operation. Writing the FLTCR in peripheral mode causes the FLTCR to be set to default state shown below. These are the only times that the FLTCR can change. Reading the FLTCR does not change its contents. The format and default values of the FLTCR are shown in Table 10.
- **Fault Address Register (FLTAR).** The FLTAR is loaded from the VAR whenever the FLTCR is loaded because of a fault detected by the MMU. Also, the FLTAR is always updated on a translation probe.
- **Virtual Address Register (VAR).** For CPU virtual mode transactions, the 32-bit virtual address is loaded into this register upon every translation. For CPU physical mode transactions, the 32-bit physical address is loaded into this register. Writing to the VAR causes the corresponding PD to be flushed from the PDC and the corresponding SD to be flushed from the SDC. If the SD is contiguous, then all PDs from that segment are flushed.
- **Configuration Register (CR).** The configuration register is used to enable or disable certain options of the MMU. Upon reset, all bits are cleared. If the CR is written to, all G bits in the PDC are cleared, all U bits in the IDNC are cleared, and all G bits in the SDC are cleared. The data cache remains unchanged. Table 11 shows the CR format.
- **Flush ID Number Register (FIDNR).** Writing the address of a SDT to the FIDNR causes all page descriptor cache entries associated with the flushed ID to be flushed from the PDC and all entries to be flushed from the SDC. The FIDNR is not readable. Table 12 shows the FIDNR format. The FIDNR is used in multiple context mode only.

Bit	63	62	61 58	57 37	36 35	34	33	32
Field	U	G	IDN	VAddr	Res	$	C	Res

Figure 12. PDC Entry Format (Word 0)

Bit	31 24	23	22	21	20 0
Field	Acc	R	M	W	PAddr

Figure 13. PDC Entry Format (Word 1)

Table 10. Fault Code Register Format and Default Values

Bit	31 — 11	10 — 7	6 — 5	4 — 0
Field	Res	AccReq	AccXlevel	FT

Bit	31 — 11	10 — 7	6 — 5	4 — 0
Field	Res	Acc Req	Acc X Level	FT
Default	0x0	0xA	Xmd Value at Write	0x0

Bit	Field	Contents	Description
0—4	FT	Fault type	Contains value of fault type that occurred (see **Faults**).
5—6	AccXlevel	Access execution level	Records execution level of access that was requested when fault occurred. Bit 5 is the least significant bit. **Access Xlevel** — **Execution Level** 00 — Kernel 01 — Executive 10 — Supervisor 11 — User
7—10	AccReq	Access requested	Access type requested by CPU when a fault occurred. Bit 7 is the least significant bit. **Access Requested Value** — **Access Type** 0000 — Move translated (MT) 0001 — Support processor data write 0011 — Support processor data fetch 0111 — Read interlocked 1000 — Address fetch 1001 — Operand fetch 1010 — Write 1100 — Instruction fetch after PC discontinuity 1101 — Instruction prefetch 1110 — Instruction fetch Any value not given is reserved.
11—31	Res	Reserved	Reserved for future use. If read, zeros are returned.

Table 11. Configuration Register Format

Bit	31 7	6	5	4 3	2	1	0
Field	Res	DCE	MCE	PS	$	R	M

Bit	Field	Contents	Description
0	M	Modified	This bit determines whether the M bit in the segment descriptors will be updated. This will be performed if the bit is set. This bit should not be modified once a translation has been completed. Such a modification could cause the M bit in the segment descriptors to be incorrect.
1	R	Referenced	This bit determines whether the R bit in the segment descriptors will be updated. This will be performed if the bit is set. This bit should not be modified once a translation has been completed. Such a modification could cause the R bit in the segment descriptors to be incorrect.
2	$	Cacheable	This bit will be reflected by the $\overline{\text{CABLE}}$ signal during miss-processing and R/M bit updating. If set, $\overline{\text{CABLE}}$ will be asserted.
3—4	PS	Page size	This field determines the page size that the MMU will use for its operations. **Bit 4 / Bit 3 / Page Size / Bit 4 / Bit 3 / Page Size** 0 / 0 / 2 Kbytes / 1 / 0 / 8 Kbytes 0 / 1 / 4 Kbytes / 1 / 1 / Reserved
5	MCE	Multiple context enable	This bit must be set to enable the multiple context feature of the MMU. If cleared, the MMU will operate in single context mode.
6	DCE	Data cache enable	This bit must be set in order to enable the data cache. For parts without the data cache, this bit is considered reserved and must contain a zero.
7—31	Res	Reserved	This field is reserved for future use. It must contain zeroes.

Table 12. Flush ID Number Register Format

Bit	31 3	2 0
Field	SDT Addr	Res

Bit	Field	Contents	Description
0—2	Res	Reserved	This field is reserved for future use. It must contain zeroes.
3—31	SDT Addr	SDT address	Address of the beginning of the segment descriptor table.

Data Cache Features

The *WE* 32201 Memory Management Unit has an on-chip physical cache that stores instructions and data. Its purpose is to increase the system performance by reducing the average system access time.

The following features are supported by the data cache:

- Zero wait state access with a cache hit.
- Automatic miss-processing with no additional wait states. This guarantees that the data cache will never adversely affect performance, even for pathological address patterns.
- Physical cache supports both physical and virtual mode accesses.
- Physical cache allows multiple processes to be cached.
- Selective caching based on page descriptor $-bit, $\overline{\text{CABLEIN}}$ signal.
- Bus monitoring maintains cache integrity in systems with multiple bus masters, MMUs.
- Expandable with multiple MMU's. In a multiple MMU system, each MMU may only cache data for which it receives a translation chip select. Since MMU address spaces do not overlap in such a system, their caches are also independent, effectively increasing the data cache size.
- All CPU data sizes, including single bytes, halfwords, three bytes, words, and double words (blockfetches) are fully supported in systems with 32-bit memory. However, partial word accesses to narrower memories will not be cached.

Data Cache Organization

The *WE* 32201 Memory Management Unit fully supports the physical data cache with an on-chip controller, tags, and 4 Kbytes of memory. The data cache is organized as follows:

- Physical cache – based on physical addresses
- 4 Kbytes – instructions and data
- 2-way set associative
- 512 entries – 256 sets of two entries
- Double-word block size – 8 bytes/entry
- Single-word sector – 4 bytes/sector.

The structure of the data cache is shown on Figure 14.

There are two parts, each with 256 entries. The entire physical address space can be mapped into either of them. Each entry has a tag field and a double word of data. The double word is, in turn, divided into single words. While both words lie in the same block, this division allows one of them to be valid while the other is not valid. The tag field is described in Table 13.

Data Cache Accesses

The cache performs one of several functions during an access, depending on whether hits or misses are detected in the data and tag, and whether the access is a read or a write. However, all cache activities are completely transparent to the CPU: miss-processing is overlapped with normal memory accesses, and hits simply result in a high speed access.

Partial word access sometimes require special handling in each of the following cases. Partial word access are defined as byte, halfword, and three-byte accesses.

- Read Tag-Hit Data-Hit: Returns data in 2 cycles providing a zero wait state transaction for the CPU.
- Read-Tag-Hit Data-Miss: The data cache performs a partial miss-processing, and then allows the CPU to complete access to memory.
- Read Tag-Miss: Indicates there was no tag hit, the data cache then performs miss-processing.
- Write Tag-Hit: The data cache is updated with new data
- Write Tag-Miss: The data cache is unaffected.

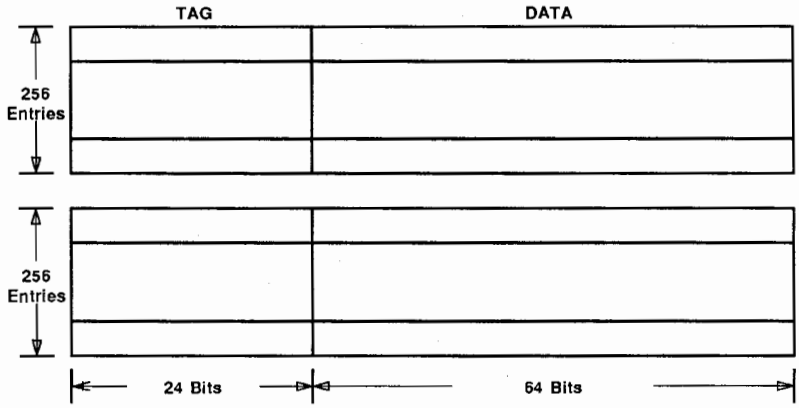


Figure 14. Data Cache Structure

Table 13. Data Cache Tag Format

Bit	23	22 21	20 0
Field	U	G	ADDRESS

Bit	Field	Contents	Description
23	U	Used	Indicates which entry of a set has been used most recently. If U=0, the left hand entry was accessed most recently; if U=1, the right hand entry was accessed most recently. Only the U bit in the left hand tag behaves this way - the right hand U bit is zero.
22—21	G	Good	There is one G bit for each of the two data words. Each G bit is set (1) if the corresponding data word and the tag address are both valid. If both bits are cleared (0), the entry is considered invalid.
20—0	Address	Address	Physical address of the double word of data.

Peripheral Mode Access

If the MMU's $\overline{MCS}$ signal is asserted, the MMU will either read or write to the requested MMU object. Peripheral mode accesses are treated independently from the translation part of the access. The translating MMU may itself be chip selected for a peripheral mode access. In a multiple MMU configuration, an MMU which has not been selected for translation may be selected for a peripheral mode operation.

$\overline{MSRDY}$ is asserted to acknowledge the transaction. This signal is negated after $\overline{DS}$ and/or $\overline{MCS}$ negate.

Memory-Mapped Peripheral Mode. This mode of operation allows many of the elements of the MMU to be addressed as memory locations; i.e., the MMU is treated as a memory-mapped peripheral. All of the objects, internal or user visible, can be accessed and all occupy some of the physical address space. A peripheral mode access occurs when the MMU chip select is asserted ($\overline{MCS}$=0). The MMU then interprets physical addresses as shown in Table 14 and on Figure 15.

Table 14. Peripheral Mode Address Mapping

Bit	31 12	11 8	6 2	1 0
Field	Res	Entity	Index	Res

Bit	Field	Contents	Description
0—1	Res	Reserved	These bits are ignored by the MMU and are treated as zeros.
2—7	Index	Index address	This field is used to index each addressable entity. Bits 2 are used when caches are accessed; bits 2 and 3 are used when section RAMs are accessed; and the field is ignored (treated as zeros) when registers are accessed.
8—11	Entity	Entity	This field selects which entity is to be accessed as per entity value specified. Bit 8 is least significant bit.

Value	Entity Addressed	Operation
0000	SDC bits 0—31	W/R
0001	SDC bits 32—63	W/R
0010	PDC bits 0—31	W/R
0011	PDC bits 32—63	W/R
0100	FDCR	W only
0101	Reserved	
0110	Section RAM A	W/R
0111	Section RAM B	W/R
1000	Fault code register (FCR)	W/R
1001	Fault address register (FAR)	W/R
1010	Configuration register (CR)	W/R
1011	Virtual address register (VAR)	W/R
1100	IDNC	R only
1101	CINDR	R only
1110	FIDNR	W only
1111	VR	R only

12—31	Res	Reserved	This field is ignored by the MMU and treated as zeros.

Faults

A fault occurs when the address translation process cannot be completed. Specifically, it is an error condition associated with read and write operations. The MMU handles faults with the FLTCR and FLTAR registers. The fault type is indicated in the FLTCR fault-type field and the virtual address of the fault is in the FLTAR. The FLTCR fault-type field is described in Table 15.

Translation Probe

The *WE* 32201 MMU provides a translation probe, which allows the operating system to quickly simulate a translation, within approximately 10 cycles above the actual translation time. The probe checks for access rights and all other faults which may occur during a normal translation, including miss-processing and R- and M-bit update faults. The translation probe does not normally assert a fault if one is detected, although the user may request this. Also, the probe access may return the address of the page or segment descriptor that was used to perform the translation, or the fault code register (FLTCR).

A translation probe is performed using the move translated word instruction and protocol. The virtual address, concatenated with a control byte, is passed to the MMU on the address bus, and the requested (e.g., FLTCR) is returned to the CPU on the data bus, in one small operation. The instruction appears as:

MOVTRW {uhalf}*src,dst

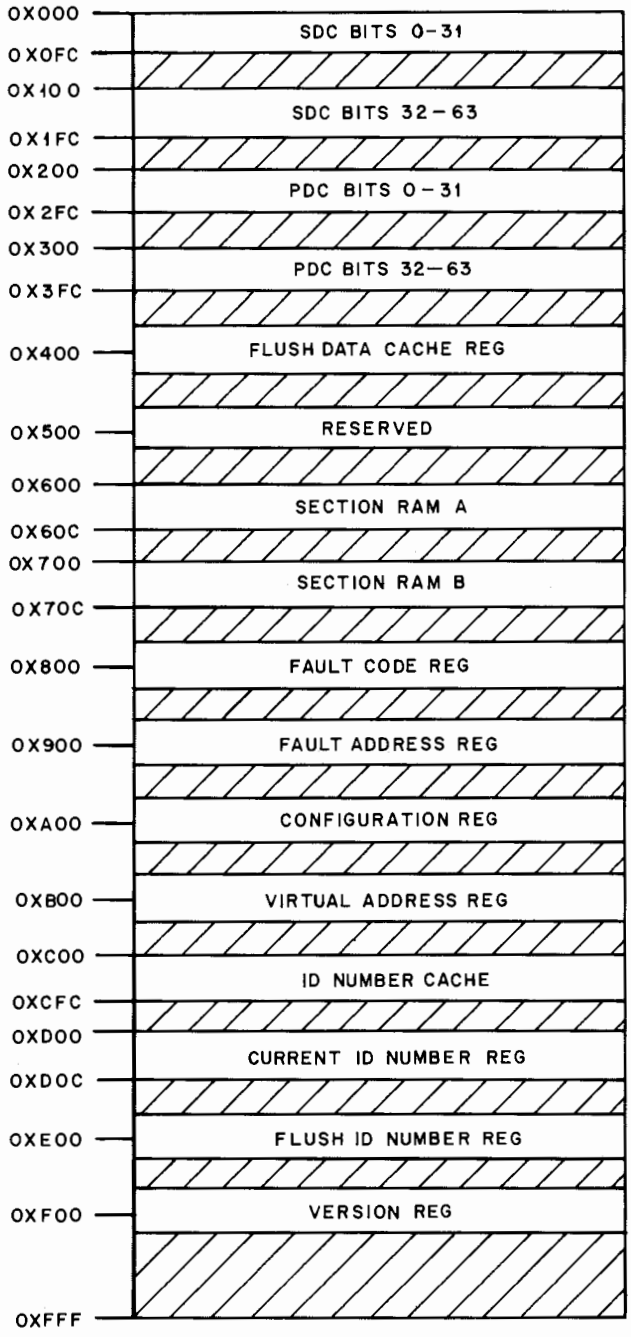


Figure 15. Peripheral Mode Memory Map

Table 15. FLTCR Fault-Type Field Description

Fault Type	Fault Name	Description
00000	No fault	No fault occurred.
00001	Miss-processing memory	Occurs when a memory fault is signaled to the MMU during a descriptor read for miss-processing purposes.
00010	R&M update memory	Occurs when a memory fault is signaled to the MMU during a descriptor read or write for R- and M-bit update purposes.
00011	SDT length	Occurs when the SSL of the virtual address exceeds the SDT length field of SRAMB.
00100	Page write	If translation for a write or interlocked read access uses a PD, the W-bit of the PD is checked. If W=1, this fault occurs.
00101	PDT length	Occurs when the lower bits of a virtual address specify a reference beyond the address range in the maximum offset field of the corresponding cached SD. Occurs only when fetching a PD or a paged segment.
00110	Invalid SD	Occurs if SD is marked invalid (V=0).
00111	Segment not present	Occurs if P-bit in SD is 0 and V=1, C=1, and I=0.
01001	PDT not present	Occurs if P-bit in SD is 0, V=1, C=0, and I=0. In this case, the PDT for the segment is not present in physical memory.
01010	Page not present	Occurs when P-bit in PD is 0.
01011	Too many indirections	Occurs if three consecutive SDs with I=1 are encountered during miss-processing.
01101	Access rights	Occurs if access requested is not permitted by access permissions field of the SD or PD used.
01110	Access offset	Occurs during miss-processing when the PSL is greater than the MaxOff field specified in the SD entry and the C bit is set.

Note: All fault type values not listed are reserved.

The *src* operand is any address or register, and points to the concatenated virtual address and control byte. The format of the *src* (probe source) is shown in Table 16. The expanded mode, {uhalf} changes the DSIZE signals to "halfword," differentiating the probe from a normal MOVTRW.

When a probe access is completed, the MMU returns a full word of data on the data bus. If the FLTCR is requested, it will be returned intact and the success of the probe can be determined from its fault type field.

Table 16. Probe Source Format

Bit	31 — 11	10 — 8	7 — 6	5 — 4	3 — 2	1 — 0
Field	Addr	xx	Acc	Xmd	T	BY

Bit	Field	Contents	Description
0—1	BY	Byte select	Bit 0 must be cleared to avoid a CPU alignment fault. Bit 1 determines whether the CPU will write the upper or lower halfword to the destination. These bits are ignored by the MMU, which always returns a full word.
2—3	T	Probe type	These bits determine what type of result will be returned. In the last case, the fault pin is asserted if a fault condition is detected during the probe. **T — Probe Request** 00 Page descriptor address 01 Segment descriptor address 10 FLTCR, no fault on fault 11 FLTCR, fault on fault
4—5	Xmd	Execution mode	These bits select the execution level for the probe access: **Xmd — Execution Level** 00 Kernel 01 Executive 10 Supervisor 11 User
6—7	Acc	Access type	These bits determine which access type will be checked during the probe, and have the same encodings as segment descriptor access rights (see Table 2). The access rights probe will pass if the permissions required are allowed in the descriptor, although an exact match is not necessary, as shown below. **SD Access Rights** Acc — RWE — RE — EO — NA RWE — Pass* — Fail — Fail — Fail RE — Pass — Pass — Fail — Fail EO — Pass — Pass — Pass — Fail NA — Pass — Pass — Pass — Pass
8—10	xx	Don't care	These are the lower address bits which are ignored, since they are not part of the page select field. For a 2 Kbyte page size, bits 8 and 9 are ignored and for a 4 Kbyte page size, bits 8—10 are ignored. Bits 8—11 are ignored for an 8 Kbyte page size.
11—31	Addr	Address	This field is the virtual address of the page to be probed. Note that only the upper 19—21 address bits are required to select the page, depending on page size.

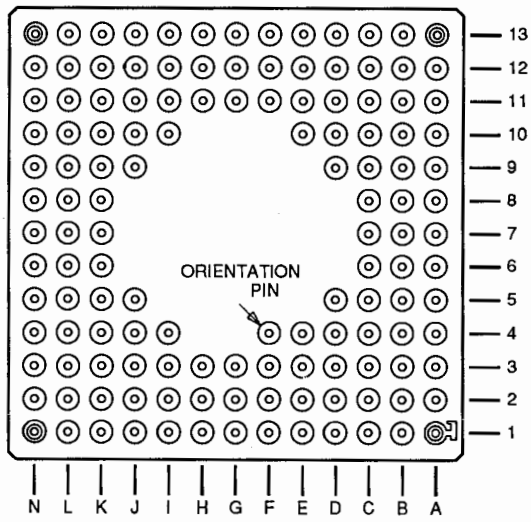
* For paged segments, the page descriptor's W bit must be set for the probe to pass.

Pin Descriptions

The *WE* 32201 Memory Management Unit is available in a 133-pin square, ceramic PGA. Figure 16 and Table 17—23 describe the pin assignments.

The term *asserted* in the pin function descriptions means that the signal is driven to its active state either by the microprocessor (outputs) or an external device (inputs). The term *negated* means that the signal is driven to its inactive state. A bar over a signal name (e.g., $\overline{AS}$) indicates that the signal is active low, logic 0. The 0 bit is the least significant bit for signals which occupy two or more pins (e.g., DSIZE0—DSIZE1). In the signal type column, I indicates an input, O output, and I/O a bidirectional signal.

Table 17 lists signals in numerical order by pin number for the 133-pin square PGA packages; Tables 18—23 describe the signals by functional group.



Bottom View

Figure 16. WE® 32201 Memory Management Unit 133-Pin Square, Ceramic PGA Package

Numerical Order

Table 17. WE® 32201 Memory Management Unit Pin Descriptions

Pin	Name	Type	Description
A1	$\overline{MRESET}$	I	MMU reset
A2	XMD1	I	Execution mode 1
A3	DATA00	I/O	Data 00
A4	ADDR00	I/O	Address 00
A5	ADDR01	I/O	Address 01
A6	ADDR03	I/O	Address 03
A7	DATA03	I/O	Data 03
A8	DATA06	I/O	Data 06
A9	DATA04	I/O	Data 04
A10	ADDR08	I/O	Address 08
A11	ADDR05	I/O	Address 05
A12	ADDR09	I/O	Address 09
A13	ADDR11	I/O	Address 11
B1	$\overline{CONTIG}$	O	Contiguous segment
B2	SAS2	I	Access status code 2
B3	XMD0	I	Execution mode 0
B4	DATA01	I/O	Data 01
B5	DATA02	I/O	Data 02
B6	ADDR02	I/O	Address 02
B7	ADDR04	I/O	Address 04
B8	DATA05	I/O	Data 05
B9	DATA07	I/O	Data 07
B10	ADDR06	I/O	Address 06
B11	ADDR07	I/O	Address 07
B12	ADDR10	I/O	Address 10
B13	DATA11	I/O	Data 11

Table 17. WE® 32201 Memory Management Unit Pin Descriptions (Continued)

Pin	Name	Type	Description
C1	$\overline{\text{VAD}}$	I	Virtual address
C2	SAS3	I	Access status code 3
C3	$\overline{\text{CABLEIN}}$	I	Cacheable in
C4	GND	—	Ground
C5	+5V	—	Power
C6	GND	—	Ground
C7	+5V	—	Power
C8	GND	—	Ground
C9	+5V	—	Power
C10	GND	—	Ground
C11	DATA08	I/O	Data 08
C12	DATA12	I/O	Data 12
C13	DATA13	I/O	Data 13
D1	$\overline{\text{MCS}}$	I	MMU chip select
D2	SAS1	I	Access status code 1
D3	GND	—	Ground
D4	SAS0	I	Access status code 0
D5	GND	—	Ground
D9	GND	—	Ground
D10	DATA10	I/O	Data 10
D11	DATA09	I/O	Data 09
D12	ADDR13	I/O	Address 13
D13	DATA14	I/O	Data 14
E1	$\overline{\text{CABLE}}$	O	Cacheable
E2	$\overline{\text{TCSA}}$	I	Translation chip select A
E3	+5V	—	Power
E4	$\overline{\text{CCYCLEI}}$	I	CPU cycle initiate
E10	GND	—	Ground
E11	+5V	—	Power
E12	ADDR12	I/O	Address 12
E13	ADDR14	I/O	Address 14
F1	$\overline{\text{AS}}$	I	Address strobe
F2	$\overline{\text{TCSB}}$	I	Translation chip select B
F3	GND	—	Ground
F4	GND	—	Ground
F11	GND	—	Ground
F12	ADDR15	I/O	Address 15
F13	DATA15	I/O	Data 15
G1	$\overline{\text{HIGHZ}}$	I	High impedance
G2	$\overline{\text{MDS}}$	O	MMU data strobe
G3	+5V	—	Power
G11	+5V	—	Power
G12	DATA16	I/O	Data 16
G13	ADDR17	I/O	Address 17
H1	$\overline{\text{MABORT}}$	I	MMU abort
H2	$\overline{\text{MDTACK}}$	I	MMU data transfer acknowledge
H3	GND	—	Ground
H11	GND	—	Ground
H12	DATA17	I/O	Data 17
H13	ADDR16	I/O	Address 16
J1	$\overline{\text{EPAS}}$	I/O	Early physical address strobe
J2	MDSIZE2	I/O	MMU data size 2
J3	+5V	—	Power
J4	GND	—	Ground
J10	RES	—	Reserved
J11	+5V	—	Power
J12	DATA18	I/O	Data 18
J13	DATA19	I/O	Data 19
K1	$\overline{\text{PAS}}$	I/O	Physical address strobe
K2	$\overline{\text{MSRDY}}$	I/O	MMU synchronous ready
K3	MDSIZE0	I/O	MMU data size 0
K4	$\overline{\text{EMCYCLEI}}$	O	Early MMU cycle initiate
K5	GND	—	Ground
K9	RES	—	Reserved
K10	$\overline{\text{BLKFETCH}}$	O	Block fetch
K11	GND	—	Ground
K12	ADDR19	I/O	Address 19
K13	ADDR18	I/O	Address 18
L1	$\overline{\text{MCYCLEI}}$	O	MMU cycle initiate
L2	$\overline{\text{RMW}}$	O	R and M bits write
L3	MR/$\overline{\text{W}}$	I/O	MMU read/write
L4	$\overline{\text{DS}}$	I	Data strobe
L5	+5V	—	Power
L6	GND	—	Ground
L7	+5V	—	Power
L8	GND	—	Ground
L9	+5V	—	Power
L10	ADDR23	I/O	Address 23
L11	DATA23	I/O	Data 23
L12	DATA21	I/O	Data 21
L13	DATA20	I/O	Data 20

Table 17. WE® 32201 Memory Management Unit Pin Descriptions (Continued)

Pin	Name	Type	Description
M1	$\overline{\text{MFAULT}}$	I/O	MMU fault
M2	$\overline{\text{MDRDY}}$	I/O	MMU data ready
M3	DATA31	I/O	Data 31
M4	ADDR31	I/O	Address 31
M5	ADDR28	I/O	Address 28
M6	DATA29	I/O	Data 29
M7	CLK23	I	Input clock 23
M8	CLK34	I	Input clock 34
M9	DATA26	I/O	Data 26
M10	ADDR25	I/O	Address 25
M11	ADDR24	I/O	Address 24
M12	DATA22	I/O	Data 22
M13	ADDR20	I/O	Address 20
N1	MDSIZE1	I/O	MMU data size 1
N2	$\overline{\text{DSHAD}}$	O	Data shadow
N3	DATA30	I/O	Data 30
N4	ADDR29	I/O	Address 29
N5	ADDR30	I/O	Address 30
N6	DATA28	I/O	Data 28
N7	ADDR27	I/O	Address 27
N8	ADDR26	I/O	Address 26
N9	DATA 27	I/O	Data 27
N10	DATA25	I/O	Data 25
N11	DATA24	I/O	Data 24
N12	ADDR22	I/O	Address 22
N13	ADDR21	I/O	Address 21

Functional Groups

Table 18. Address and Data

Name	Pin(s)	Type	Function
ADDR00—ADDR31	A4, A5, B6, A6, B7, A11, B10, B11, A10, A12, B12, A13, E12, D12, E13, F12, H13, G13, K13, K12, M13, N13, N12, L10, M11, M10, N8, N7, M5, N4, N5, M4	I/O	**Address Bus.** These pins provide a bi-directional bus for virtual (input) and physical (output) addresses. When $\overline{\text{VAD}}$ is asserted, a virtual address is received from the CPU. When $\overline{\text{PAS}}$ is asserted, a physical address exists on the bus. ADDR00 is the least significant address bit.
DATA00—DATA31	A3, B4, B5, A7, A9, B8, A8, B9, C11, D11, D10, B13, C12, C13, D13, F13, G12, H12, J12, J13, L13, L12, M12, L11, N11, N10, M9, N9, N6, M6, N3, M3	I/O	**Data Bus.** These pins provide a bi-directional bus to convey descriptors to and from the MMU. DATA00 is the least significant data bit.

Table 19. Interface Control

Name	Pin(s)	Type	Function
$\overline{\text{AS}}$	F1	I	**Address Strobe.** Indicates a valid address on the address bus.
$\overline{\text{CCYCLEI}}$	E4	I	**CPU Cycle Initiate.** This signal is asserted by the CPU whenever it initiates a bus transaction.
$\overline{\text{DS}}$	L4	I	**Data Strobe.** Indicates the state of the CPU data strobe to the MMU. During physical address cycles the $\overline{\text{MDS}}$ signal is used as a surrogate for the CPU data strobe.
$\overline{\text{DSHAD}}$	N2	O	**Data Shadow.** Indicates that the MMU is master of the address and data buses during miss-processing and R-and M-bit updates.

Table 19. Interface Control (Continued)

Name	Pin(s)	Type	Function
$\overline{EPAS}$	J1	I/O	**Early Physical Address Strobe.** This signal, when asserted, indicates that a valid physical address is on the address bus. It is asserted two phases earlier than $\overline{PAS}$. Either $\overline{EPAS}$ or $\overline{PAS}$ may be used to indicate a physical bus transaction to the data cache.
$\overline{MCS}$	D1	I	**MMU Chip Select.** When asserted, indicates that the MMU is operating in the peripheral mode.
$\overline{EMCYCLEI}$	K4	O	**Early MMU Cycle Initiate.** This signal is asserted by the MMU at the beginning of read cycles (miss-processing) and read-modify-write cycles (R & M updates). It is also issued with the physical address during translation and in the first wait state of a physical mode access. Note that $\overline{EMCYCLEI}$ is issued one half cycle before $\overline{MCYCLEI}$.
$\overline{MCYCLEI}$	L1	O	**MMU Cycle Initiate.** This signal is asserted at the beginning of read cycles (miss-processing), read-modify-write cycles (R & M updates), and the physical address during translation.
$\overline{MDRDY}$	M2	I/O	**MMU Data Ready.** Indicates a successful (no fault) descriptor access during R- and M-bit updates on miss-processing. When $\overline{DSHAD}$ is inactive, $\overline{MDRDY}$ is 3-stated. This signal is sampled when the cache must latch data from memory (during read misses and write hits).
$\overline{MDS}$	G2	O	**MMU Data Strobe.** During a read cycle this signal indicates data may be placed on the data bus. On a write cycle it indicates valid data are on the data bus. During physical address cycles this signal is an extension of the CPU data strobe.
$\overline{MDTACK}$	H2	I	**MMU Data Transfer Acknowledge.** Indicates to the MMU if wait states are inserted in an MMU memory access. If not asserted when sampled during miss-processing, R- and M-bit update wait states are inserted.
$\overline{MSRDY}$	K2	I/O	**Synchronous Ready.** As an input, this signal is sampled during miss-processing and during R- and M-bit updating. If not asserted when sampled, wait states are inserted in an MMU memory access. $\overline{MSRDY}$ is an output when the MMU is in peripheral mode.
$\overline{PAS}$	K1	I/O	**Physical Address Strobe.** When asserted, this signal indicates that a valid physical address is on the address bus. This signal is an input when the MMU is in the bus monitoring mode.
$\overline{TCSA}$,$\overline{TCSB}$	E2,F2	I	**Translation Chip Selects.** When both are asserted, these signals instruct the MMU to perform translation functions. If only one signal is asserted, the MMU ignores the address generated by the CPU. If both signals are not asserted, then all MMU outputs are 3-stated. These signals aid multiple MMU configurations.

Table 20. MMU Status

Name	Pin(s)	Type	Function
$\overline{CABLE}$	E1	O	**Cacheable.** Indicates the value of the $ bit of the descriptor used for translation. $\overline{CABLE}$ is valid with $\overline{PAS}$.
$\overline{CABLEIN}$	C3	I	**Cacheable In.** If this signal is negated, then the data cache will not cache data returned from memory.

Table 20. MMU Status (Continued)

Name	Pin(s)	Type	Function
$\overline{\text{CONTIG}}$	B1	O	**Contiguous Segment.** Indicates a contiguous segment is being translated. During miss- processing or R- and M-bit updates, $\overline{\text{CONTIG}}$ is asserted.
MDSIZE0—MDSIZE2	K3, N1, J2	I/O	**MMU Data Size.** These signals indicate the size of the current CPU bus transaction. They are driven by the MMU whenever it is performing miss-processing or R/M bit updating. In the latter case, they will always indicate a word transaction. They are interpreted as follows: **MDSIZE 2 1 0** — **Transaction Size** 0 0 0 — 3 bytes 0 0 1 — 0 bytes 0 1 0 — reserved (triple word) 0 1 1 — quad word 1 0 0 — word 1 0 1 — double word 1 1 0 — half word 1 1 1 — byte
MR/$\overline{\text{W}}$	L3	I/O	**MMU Read/Write.** As an output, indicates that a read or write operation is in progress. As an input, indicates that the CPU is performing a read or write.
$\overline{\text{RMW}}$	L2	O	**R and M Bits Write.** Indicates an R- and M-bit update when the MMU is master of address and data buses.
SAS0—SAS3	D4, D2, B2, C2	I	**Access Status Code.** These signals indicate the access status. (See Table 24. Access Status Code Assignments.)
$\overline{\text{VAD}}$	C1	I	**Virtual Address.** Indicates that address is virtual. When not asserted, address is physical.
XMD0, XMD1	B3, A2	I	**Execution Mode.** These signals indicate the current execution mode of the CPU.
$\overline{\text{BLKFETCH}}$	K10	O	**Block Fetch.** This signal is asserted if a data cache hit occurs on a CPU block fetch transaction.

Table 21. Exceptions

Name	Pin(s)	Type	Function
$\overline{\text{MABORT}}$	H1	I	**MMU Abort.** Terminates MMU operation when asserted. $\overline{\text{MABORT}}$ is used on aborted CPU cache fill operations.
$\overline{\text{MFAULT}}$	M1	I/O	**MMU Fault.** As an input, $\overline{\text{MFAULT}}$ indicates miss-processing or R- and M-bit update memory faults. $\overline{\text{MFAULT}}$ is an input when the MMU has control of the buses. As an output, $\overline{\text{MFAULT}}$ indicates a fault during translation, a descriptor-related fault during miss-processing, or R- and M-bit updates only after $\overline{\text{DSHAD}}$ is disabled.
$\overline{\text{MRESET}}$	A1	I	**MMU Reset.** Resets the MMU to its last known state.

Table 22. Clocks

Name	Pin(s)	Type	Function
CLK23	M7	I	**Clock.** Connected to the CLK23 input clock of the CPU.
CLK34	M8	I	**Clock.** Connected to the CLK34 input clock of the CPU.

Table 23. Test

Name	Pin(s)	Type	Function
$\overline{\text{HIGHZ}}$	G1	I	**Test.** 3-states all MMU output signals.

Table 24. Access Status Code Assignments

Access Status Inputs SAS0—SAS3				Access Type	MMU Behavior	Permission
3	2	1	0			
0	0	0	0	Move translated word	Perform MOVTRW sequence	Read
0	0	0	1	Support process data write	Normal translation	Write
0	0	1	0	Autovector interrupt acknowledge	Physical mode access	—
0	0	1	1	Support processor data fetch	Normal translation	Read
0	1	0	0	Stop acknowledge	Ignore	—
0	1	0	1	Support processor broadcast	Ignore	—
0	1	1	0	Support processor status fetch	Ignore	—
0	1	1	1	Read interlocked	Normal translation	Read/write
1	0	0	0	Address fetch	Normal translation	Read
1	0	0	1	Operand fetch	Normal translation	Read
1	0	1	0	Write	Normal translation	Write
1	0	1	1	Interrupt acknowledge	Physical mode access	—
1	1	0	0	Instruction fetch after PC discontinuity	Normal translation	Execute
1	1	0	1	Instruction prefetch	Normal translation	Execute
1	1	1	0	Instruction fetch	Normal translation	Execute
1	1	1	1	No operation	Ignore	—

Characteristics

V_{CC} = 5.0 V ± 5%, V_{SS} = 0 V, C_L = 130 pF, T_A = 0 to 70 °C

Timing Characteristics

All TTL timing specifications are referenced to and from 0.8 V for a low voltage and 2.0 V for a high voltage. CMOS clock references are to and from $V_{CC}/2$. All min and max values are in ns.

Table 25. Timing Characteristics

Num	Symbol	Description	Fig(s)	24 MHz	
				Min	Max
1	tC34LSASV	Access status code set-up time	22	3	—
2	tC34LSASX	Access status code hold time	22	3	—
3	tASLTCSV	Translation chip select set-up time	22	3	—
4	tASLTCSX	Translation chip select hold time	22	3	—
5	tC34LVADV	$\overline{VAD}$ set-up time	22	3	—
6	tC34LVADX	$\overline{VAD}$ hold time	22	3	—
7	tC34LXMDV	Execution mode set-up time	24	3	—
8	tC34LXMDX	Execution mode hold time	24	3	—
9	tC34HDTAV	MMU data transfer* acknowledge set-up time	25	3	—
10	tMDSHDTAH	MMU data transfer acknowledge asserted	25	0	—
11	tC34LMSRV	MMU synchronous ready set-up time	25, 26	3	—
12	tC34LMSRX	MMU synchronous ready hold time	25, 26	3	—
13	tC34HMRWV	MMU read/write set-up time	22	3	—
14	tC34LMRWX	MMU read/write hold time	22	3	—
15	tC23HASV	Address strobe set-up time	17, 22	3	—
16	tC23HASX	Address strobe hold time	22	3	—
17	tC23HABTV	MMU abort set-up time	18	3	—
18	tC23HABTX	MMU abort hold time	18	3	—
19	tC23HMCSV	MMU chip select set-up time	22	3	—
20	tMPASHMCSH	MMU chip select asserted	22	0	—
20A	tMPASLMCSL	MMU chip select assertion time	—	0	—
21A	tC34HMFTV	MMU fault set-up time*	27	3	—

* This is an asynchronous signal. Set-up times are specified for testing and informational purposes. The set-up times guarantee recognition at the next clock edge.

Table 25. Timing Characteristics (Continued)

Num	Symbol	Description	Fig(s)	24 MHz	
				Min	Max
21B	tC34LMFTV	MMU fault set-up time*	26	3	—
22B	tC34LMFTX	MMU fault hold time*	26	3	—
25	tC23LADDV	Address set-up time; referenced from clock 23	22, 24	3	—
26	tC23LADDZ	Address hold time (input)	22, 24	3	—
27	tC23LDATV	Data set-up time – peripheral mode	22	3	—
28	tC23LDATX	Data hold time – peripheral mode	22	3	—
29	tC34HDATV	Data set-up time for MMU-initiated access	25	3	—
30	tC34HDATX	Data hold time for MMU-initiated access	25	3	—
31	tC23HDSV	Data strobe negation set-up time	21	3	—
32	tC23HDSH	Data strobe negation hold time	21	3	—
33	tC23LMRTV	$\overline{\text{MRESET}}$ set-up time	—	3	—
34	tMRTLMRTX	$\overline{\text{MRESET}}$ hold time	—	3Tc	—
35A	tC23LCCYV	$\overline{\text{CCYCLEI}}$ set-up time	—	3	—
35B	tC23LCCYX	$\overline{\text{CCYCLEI}}$ hold time	—	3	—
36A	tC23LMDZV	MDSIZE set-up time	—	3	—
36B	tC34LMDZX	MDSIZE hold time	—	3	—
37A	tC34LCBNV	$\overline{\text{CABLEIN}}$ set-up time	—	3	—
37B	tC34LCBNX	$\overline{\text{CABLEIN}}$ hold time	—	3	—
40	tC34LEPAL	Early address strobe assertion time	24, 25	—	23
41	tC23HEPAH	Early address strobe negation time	24, 25	—	23
42A	tC34LPASL	Physical address strobe assertion time	24, 25	—	23
42B	tPASLADDV	Address set-up time; referenced from physical address strobe	24, 25	38	—
42C	tEPALADDV	Address set-up time; referenced from early physical address strobe	—	17	—
43A	tC23HPASH	Physical address strobe negation time (CPU initiated)	24	—	23
43B	tPASHADDZ	Address hold time (output)	24, 25	5	—

* This is a synchronous signal when asserted with $\overline{\text{DTACK}}$ or $\overline{\text{MSRDY}}$.

Table 25. Timing Characteristics (Continued)

Num	Symbol	Description	Fig(s)	24 MHz	
				Min	Max
43C	tC34HPASH	Physical address strobe negation time (MMU initiated)	25	—	23
44A	tC34HMDSL	MMU data strobe assertion time (CPU read)	21, 24 25	—	23
44B	tMDSLDATV	Data set-up time	25	17	—
44C	tC34LMDSL	MMU data strobe assertion time (CPU write)	22	—	23
45A	tC23HMDSH	MMU data strobe negation time (CPU initiated)	24	—	23
45B	tMDSHDATZ	Data hold time	25	5	—
45C	tMDSHPASH	Physical address strobe negation time	24, 25	–3	—
45D	tC34HMDSH	MMU data strobe negation time (MMU initiated)	25	—	23
46	tC34LCBLL	Cacheable assertion time	23, 24	—	23
47A	tC34LCBLH	Cacheable negation time	25	—	23
47B	tC34HCBLH	Cacheable negation time	24	—	23
48A	tC34LCNTL	Continuous segment assertion time	23, 24	—	23
48B	tC34HCNTH	Continuous segment negation time	24	—	23
49	tC23HBLFL	BLKFTCH assertion time	—	—	23
49A	tASHBLFZ	Address strobe rising edge to BLKFTCH high impedance	—	—	23
50	tC34LMCYL	MMU cycle initiate assertion time	24, 25	—	23
50A	tC34HEMCL	Early MMU cycle initiate assertion time	24, 25	—	23
51	tC34LMCYH	MMU cycle initiate negation time	24, 25	—	23
51A	tC34LEMCH	Early MMU cycle initiate negation time	24, 25	—	23
52	tC23HMSRL	MMU synchronous ready assertion time	22, 26 28	—	23
53	tASHMSRZ	Address strobe rising edge to MMU synchronous ready high impedance	22, 26 27	—	23
54	tC23HMFTL	MMU fault assertion time	26, 27	—	23
55	tASHMFTZ	Address strobe rising edge to MMU fault high impedance	26, 27	—	23

Table 25. Timing Characteristics (Continued)

Num	Symbol	Description	Fig(s)	24 MHz	
				Min	Max
56	tC34LRMWL	R- and M-bits write assertion time	25	—	23
57	tC34LRMWH	R- and M-bits write negation time	25	—	23
58	tC34LDSHL	Data shadow assertion time	23	—	23
59	tC34LDSHH	Data shadow negation time	25, 26, 27	—	23
60	tC34LMDZL	MMU data size assertion time	23	—	23
61	tC34LMDZZ	Clock 34 low to MMU data size high impedance time	25	—	23
62	tC34LMRWH	MMU read/write assertion time	23, 25	—	23
63	tC34LMRWL	MMU read/write negation	25	—	23
64	tC34HADDV	Address assertion time	24, 25	—	23
65	tC34LADDZ	Clock 34 low to address high impedance	24, 25	—	23
66	tC34HDATV	Data assertion time	20, 21, 25	—	23
67	tC34LDATZ	Clock 34 low to data high impedance	20, 21, 25	—	23
68	tC34HMDRL	MMU data ready assertion time	25	—	23
69	tC34HMDRH	MMU data ready negation time	25	—	23
70	tHIZLOUTZ	HIGHZ low to all outputs high impedance	—	—	23
70A	tC23LMSOZ	Clock 23 low to strobe outputs high impedance	17, 18	—	23
71	tHIZHOUTV	HIGHZ high to all outputs valid	19	—	23
72	tC23J	Clock 23 jitter	19	—	0.5
73	tC23E	Clock 23 duty cycle error	19	—	2
74	tC23L1C23H2	Clock 23 rise time	19	—	4
75	tC23H2C23L1	Clock 23 fall time	19	—	4
76	tC23L1C23L1	Clock 23 period (T)	19	41	—
77	tC34J	Clock 34 jitter	—	—	0.5
78	tC34E	Clock 34 duty cycle error	—	—	2
79	tC34L1C34H2	Clock 34 rise time	—	—	4
80	tC34H2C34L1	Clock 34 fall time	—	—	4
81	tC34L1C34L1	Clock 34 period (T)	—	41	—
82	tSKEW	Clock skew	—	—	2

Timing Diagrams

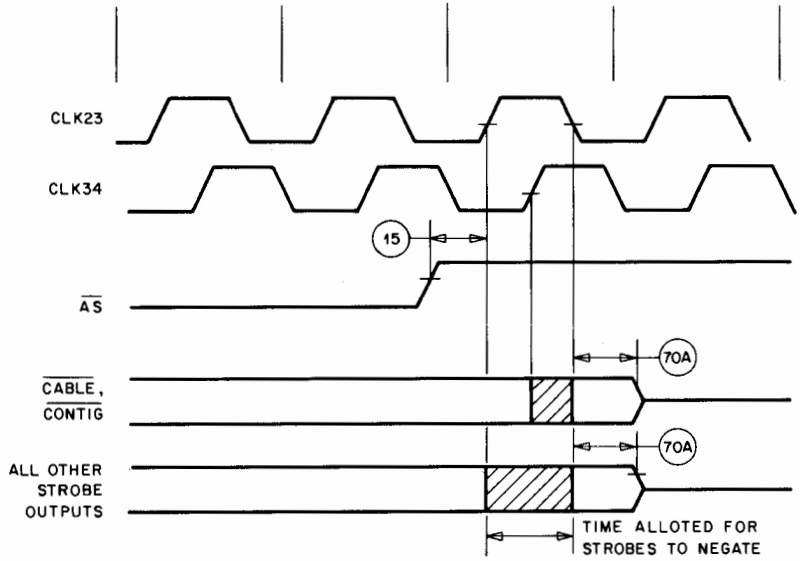


Figure 17. CPU Terminates Access

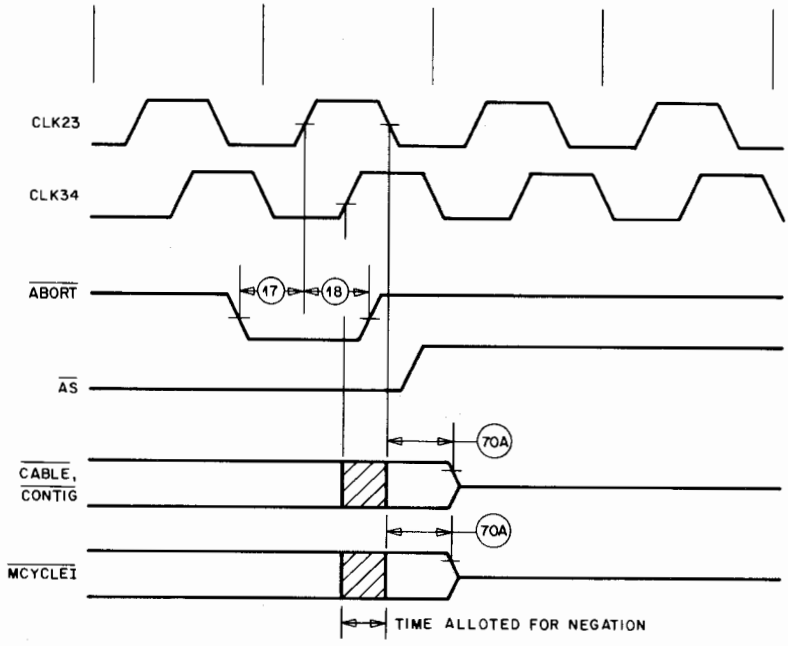
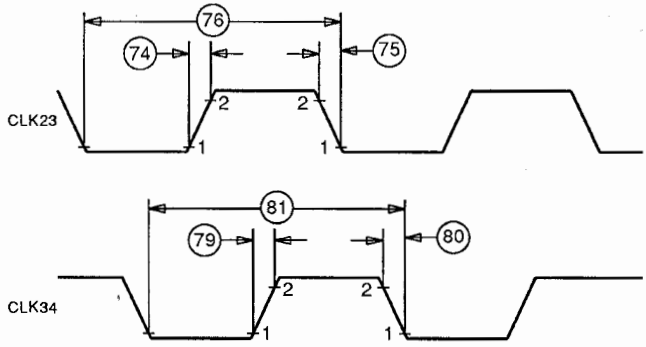


Figure 18. CPU Aborted Access



Duty Cycle Error – The duty cycle of each clock input may deviate for 50% but should not exceed timing specification numbers 73 and 78 for CLK23 and CLK34, respectively.

Skew – For 10 MHz operation, CLK23 nominally leads CLK34 by 90° (1/4 clock period). This phase lead should not deviate more than timing specification number 82.

Jitter – The period of each clock input may deviate from its nominal value and should not exceed timing specification numbers 72 and 77 for CLK23 and CLK34, respectively.

Figure 19. Clock Inputs

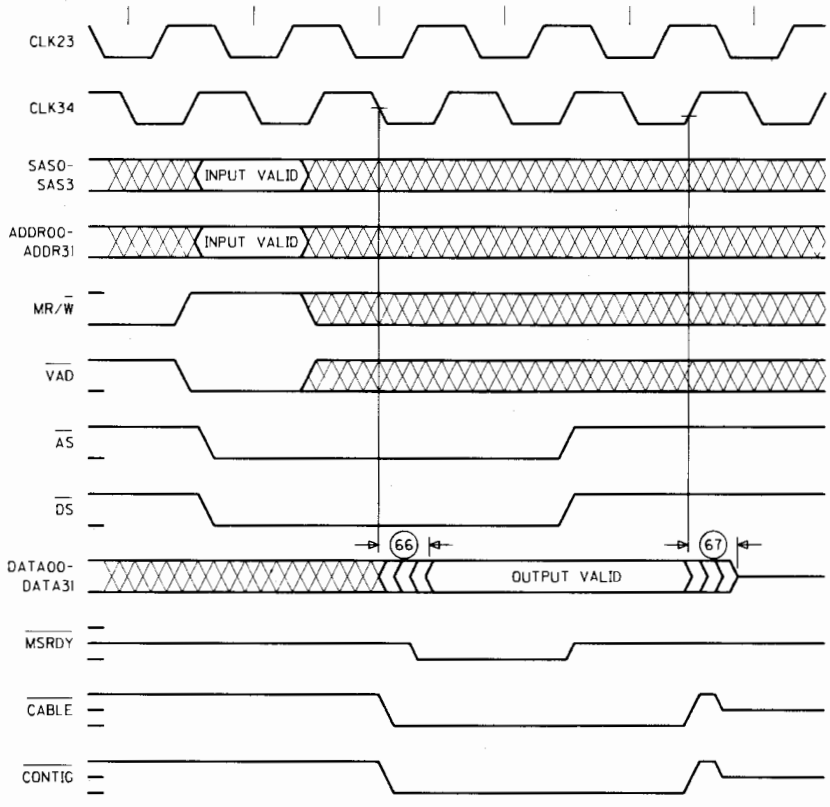


Figure 20. CPU Virtual Move Translated Word Access Cache Hit Translation

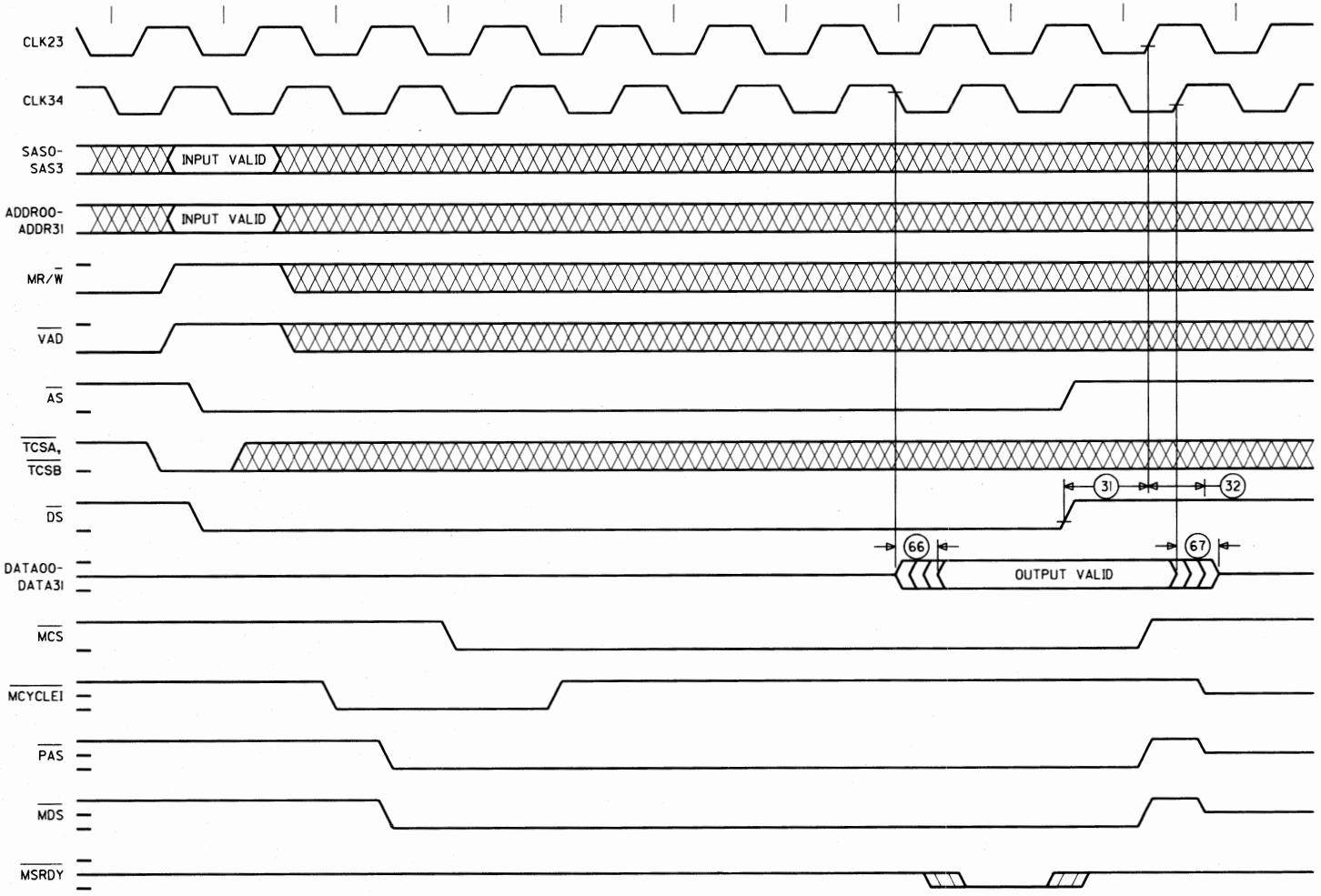


Figure 21. CPU Physical Access Peripheral Mode Read

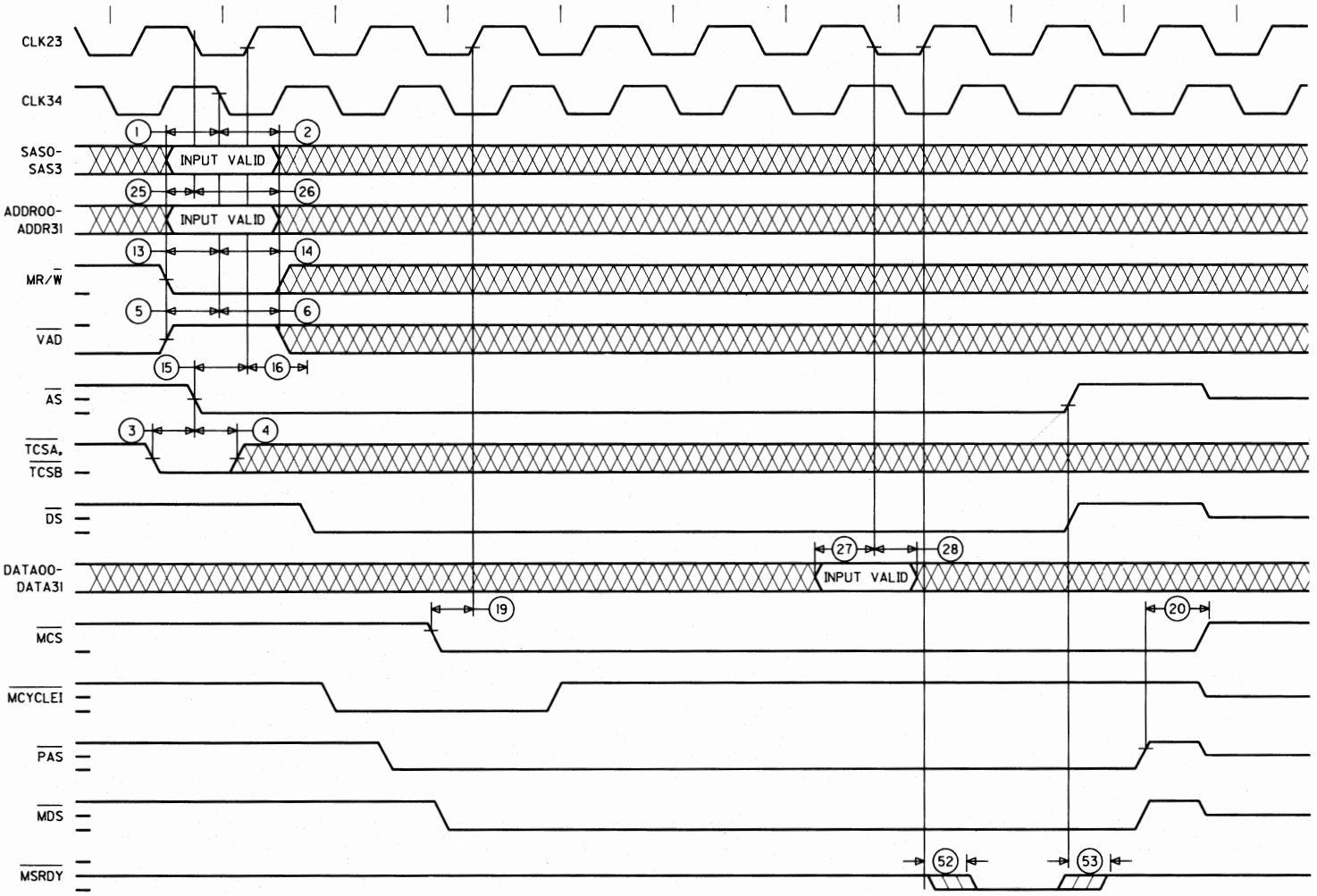


Figure 22. CPU Physical Access Peripheral Mode Write

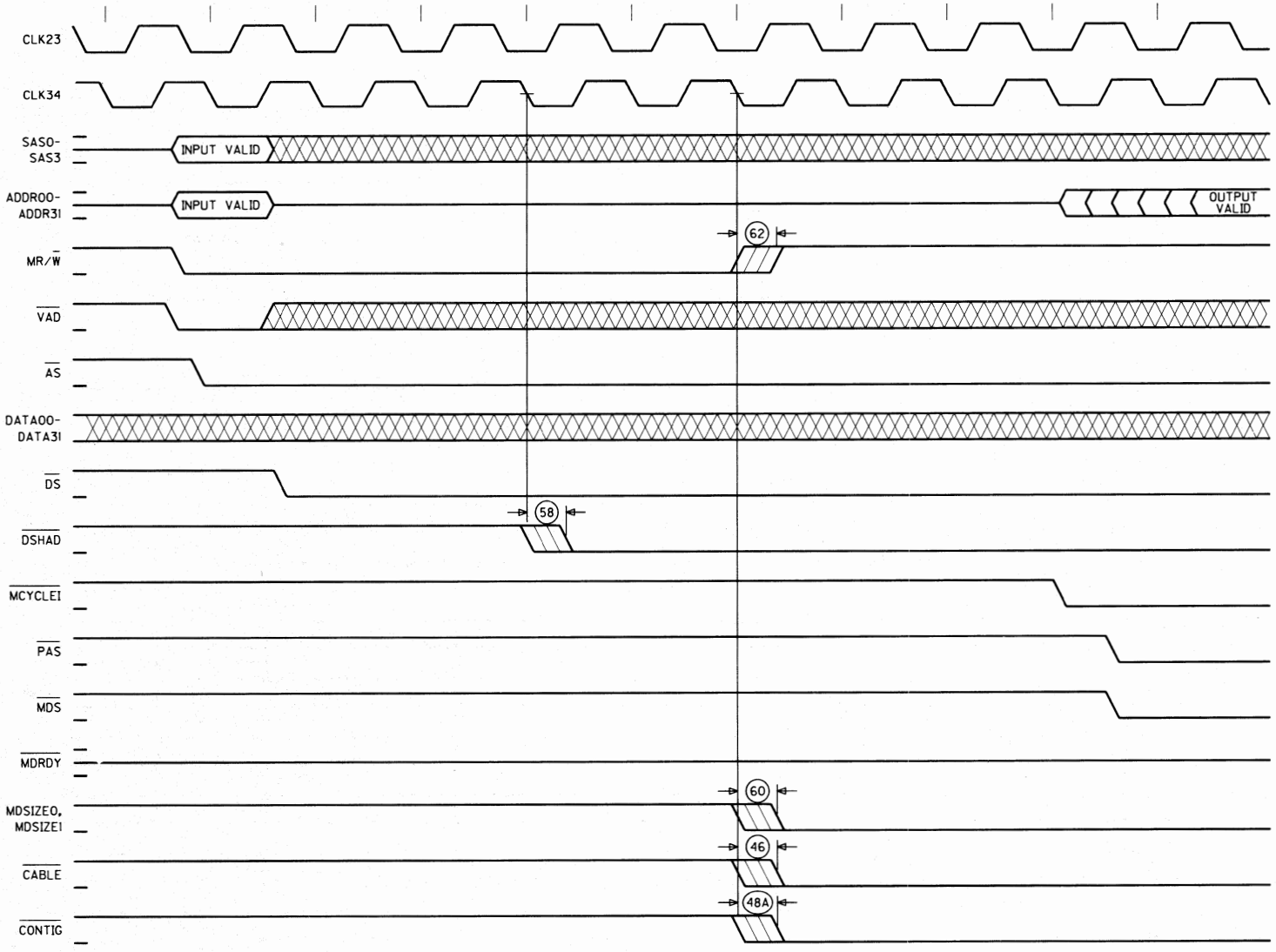


Figure 23. Write Access Miss-Process Start (Segment Miss)

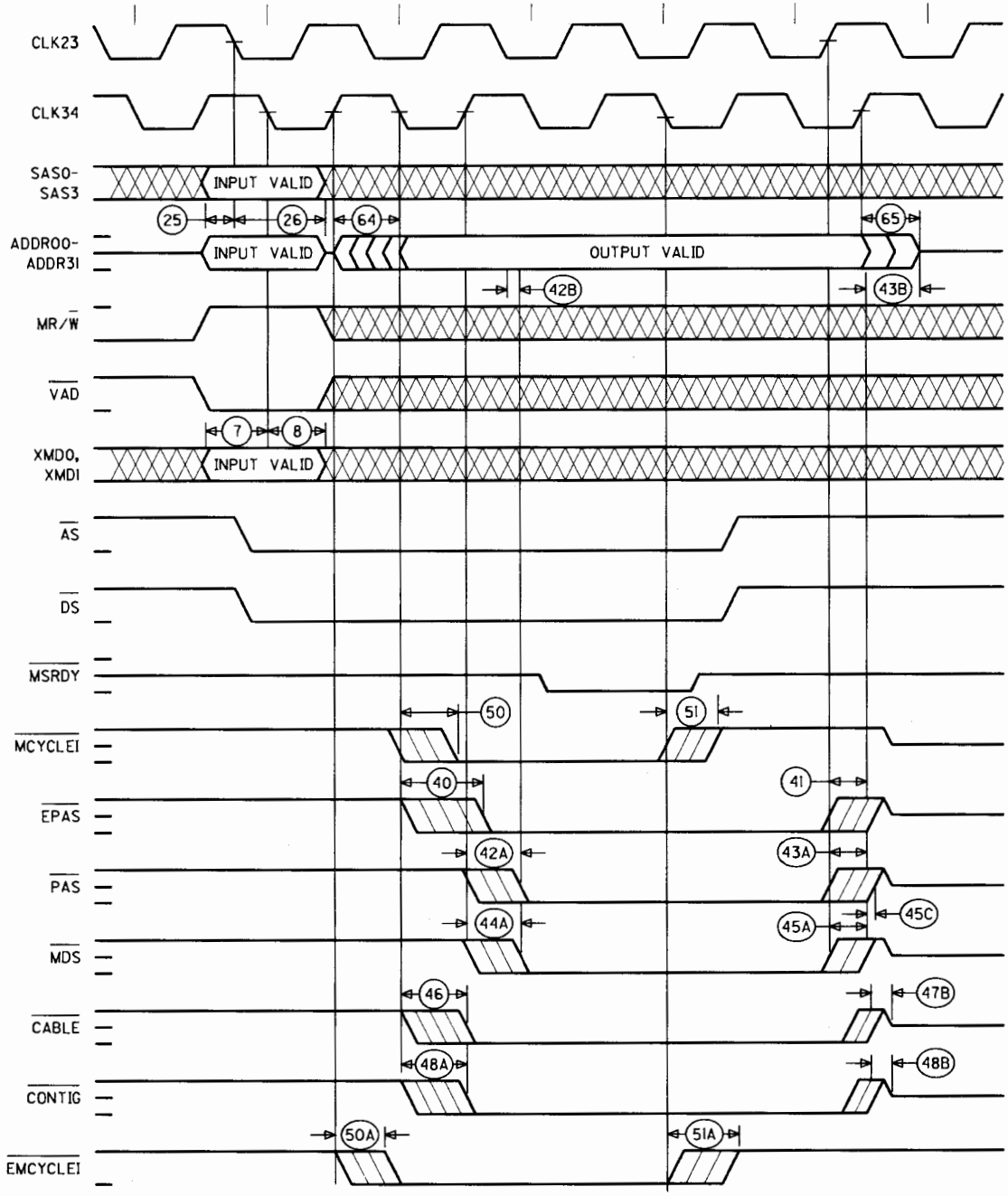


Figure 24. Read Access Cache Hit Translation

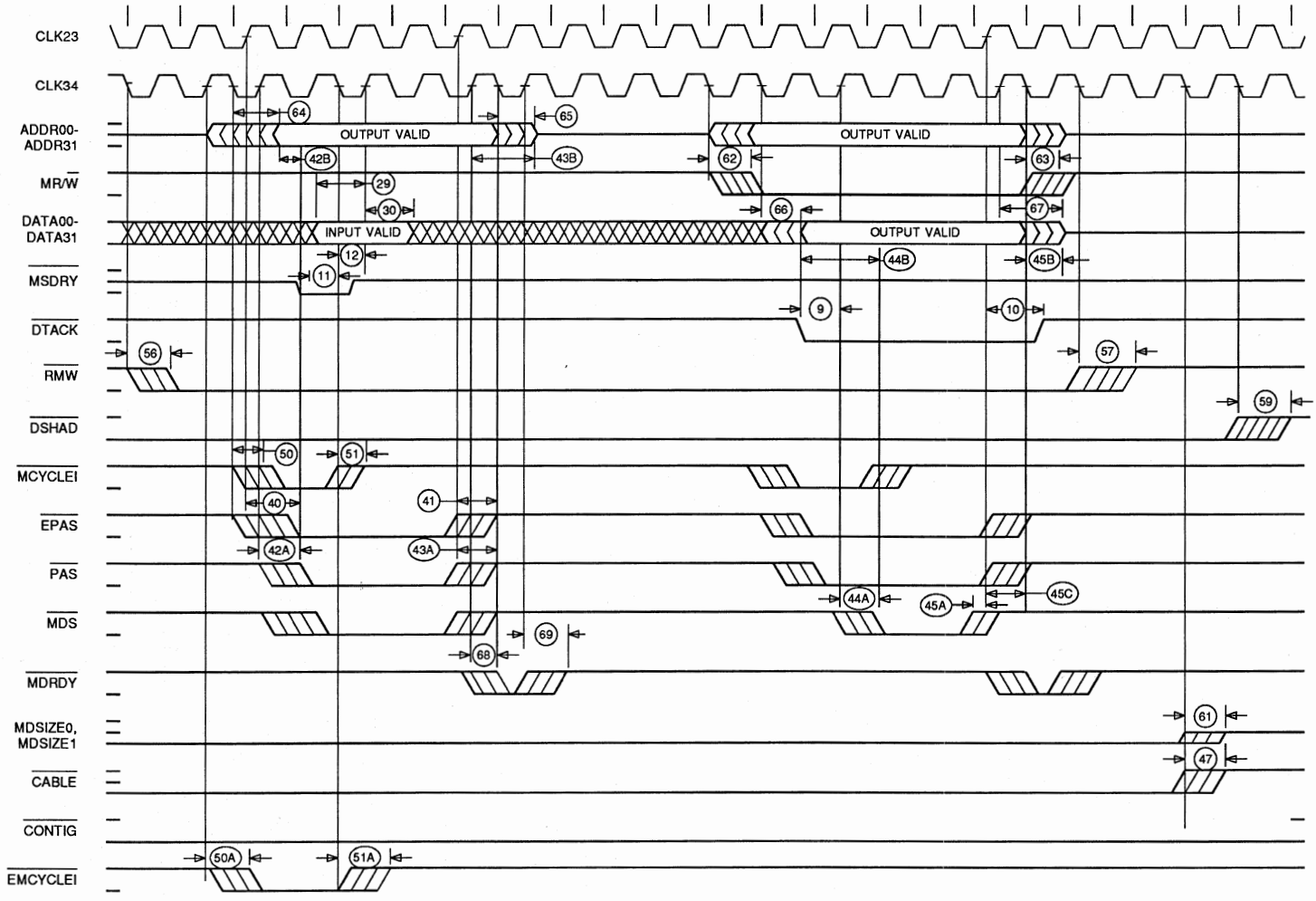


Figure 25. MMU Read-Modify-Write Access

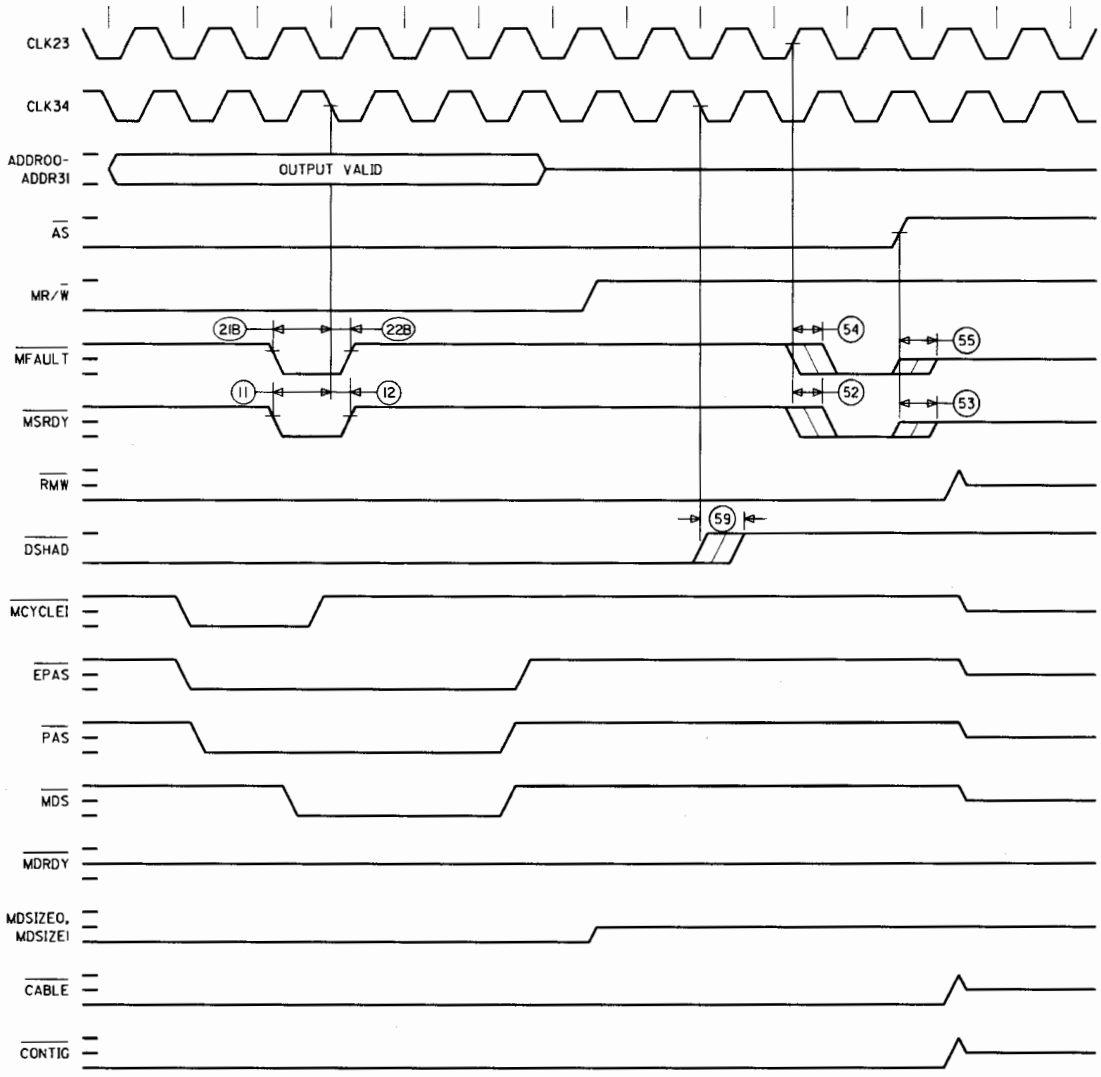


Figure 26. MMU Write Access With Synchronous Memory Faults

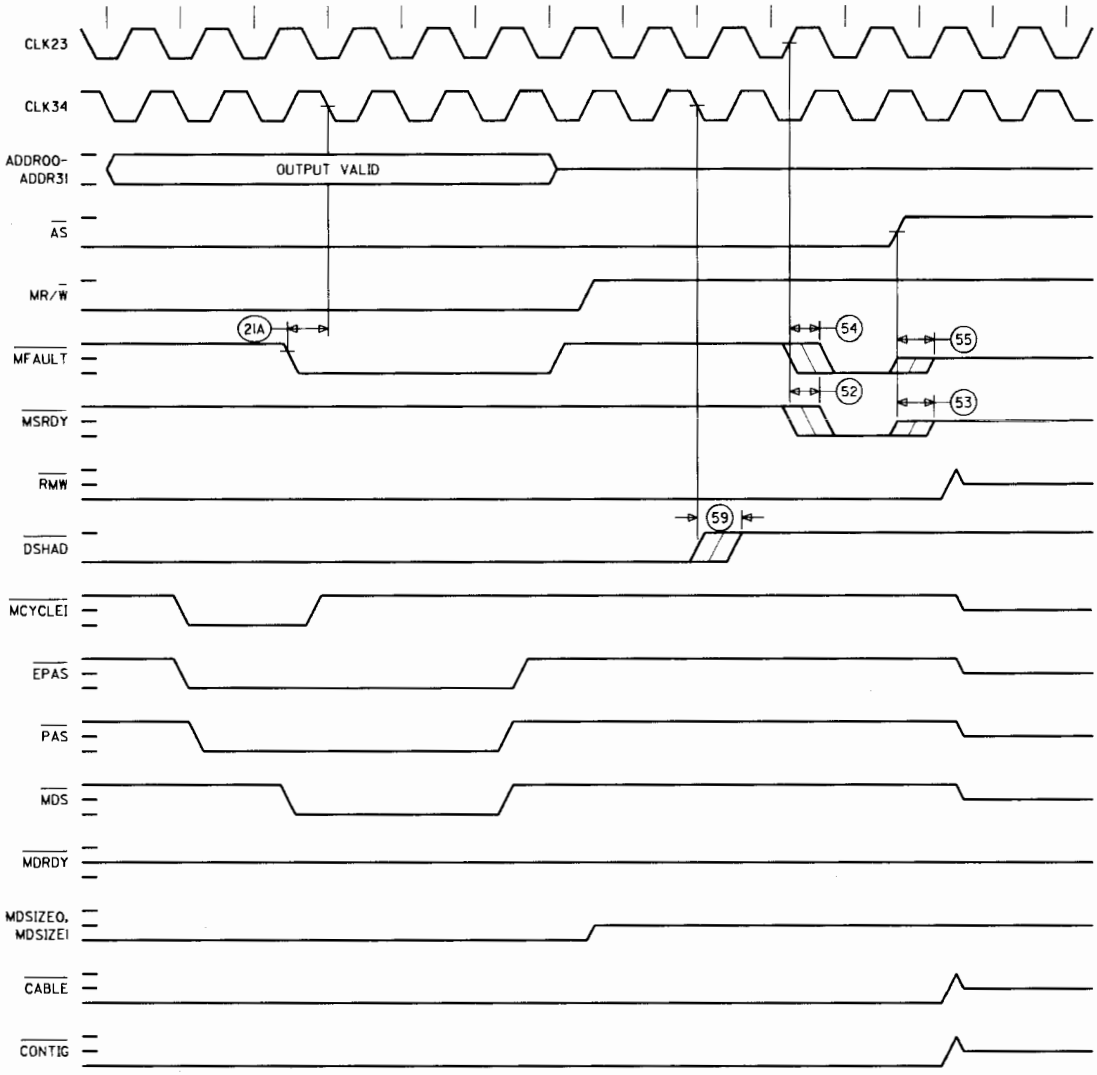


Figure 27. MMU Write Access With Asynchronous Memory Fault

Electrical Characteristics

Unless otherwise stated, all voltage level specifications are referenced to either V_{CC} (power supply input to the MMU chip package) or GND (ground input to the MMU chip package).

Inputs

All inputs, except the CMOS clock, are TTL compatible.

Table 26. DC Input Parameters

Inputs		Min	Nom	Max	Unit
TTL input voltage	high-level	2.0	—	V_{CC}+0.5	V
	low-level	−0.5	—	0.8	V
CMOS clocks input voltage	high-level	V_{CC}−1.3	—	V_{CC}+0.5	V
	low-level	0	—	0.8	V
TTL input loading current For V_{IH} ($2.0\ V \leq V_{IH} \leq V_{CC}$)	high-level	0	—	0.01	mA
For V_{IL} ($0V \leq V_{IL} \leq 0.8\ V$)	low-level	−0.01	—	0	mA
CMOS clocks input loading current For V_{IH} ($V_{CC}-1.3\ V \leq V_{IH} \leq V_{CC}$)	high-level	0	—	0.01	mA
For V_{IL} ($0\ V \leq V_{IL} \leq 0.8\ V$)	low-level	−0.01	—	0	mA

Outputs

Two classes of outputs are provided that support TTL input voltage levels. These classes are defined as:

Class 1: This class is capable of driving one TTL load or 8 PNP Schottky TTL loads. This class has current allowance for an external pull-up resistor employed in 3-state buffers that have the feature of returning to logic one before 3-stating. The minimum pull-up resistor value is 2.7 kΩ, with the exception that no pull-up resistors are needed for ADDR00—ADDR31.

Class 2: The signal in this class is an open drain device used for wired-logic operations, allowing more than one device to drive a node without conflict. An external resistor is required to pull the signal high. The minimum pull-up resistor value is 510 Ω.

The following lists the outputs assigned to each class:

Class 1	Class 2
ADDR00—ADDR31	$\overline{\text{MFAULT}}$
DATA00—DATA31	$\overline{\text{MSRDY}}$
$\overline{\text{MCYCLE}}$	
$\overline{\text{MDS}}$	
$\overline{\text{PAS}}$	
$\overline{\text{CABLE}}$	
$\overline{\text{DSHAD}}$	
$\overline{\text{MDRDY}}$	
MDSIZE0—MDSIZE 2	
MR/$\overline{\text{W}}$	

Table 27. DC Output Parameters

Outputs		Min	Nom	Max	Unit
Output sink current (I_{OL}) ($V_{OL} \leq 0.4$ V)	Class 1	—	—	5.5	mA
	Class 2	—	—	12.0	mA
Output source current (I_{OH}) ($V_{OH} \geq 2.4$ V)	Class 1	—	—	2.0	mA
	Class 2	—	—	0.01 Open drain	mA
Output logic levels	high-level	2.4	—	—	V
	low-level	—	—	0.4	V

Operating Conditions

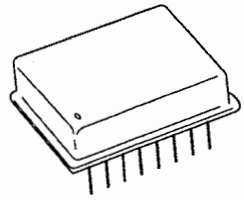
Table 28. DC Operating Conditions

Parameter		Symbol	Min	Nom	Max	Unit
Supply voltage		V_{CC}	4.75	5.00	5.25	V
Input load capacitance	TTL inputs	C_{IL}	—	—	12	pF
	CMOS clocks		—	—	7	pF
Total output load capacitance	Class 1	C_L	—	—	130	pF
	Class 2		—	—	130	pF
Ambient temperature at the microprocessor pins		T_A	0	—	70	°C
Humidity range		—	5%	—	95%	—
Power dissipation		PD	—	—	2.4	W
Operating frequency		F	24	—	—	MHz

WE® 32102 Clock

Description

The *WE* 32102 Clock supplies the two-phase, CMOS-level frequency source required by the *WE* 32-Bit Microprocessors and Peripherals. The *WE* 32102 Clock generates three outputs from a crystal controlled encoder. Two of these outputs are of the same frequency but 90° out of phase. A third output is twice the frequency (2X) of the other clock outputs. Other features include an external frequency source input, which provides control of the clock's output. The *WE* 32102 Clock is available at operating frequencies of 10-, 14-, 18-, and 24-MHz. The 2X output is available with the 10- and 14-MHz devices. The clock is housed in a hermetically sealed metal can (double-width, dual-in-line package) and requires a single 5 V supply for operation.



Features

- Two outputs 90° apart
- One double frequency output (2X) for 10- and 14-MHz
- Input for external clock source
- Input and output at CMOS levels
- Each output drives up to 130 pF of capacitive loading

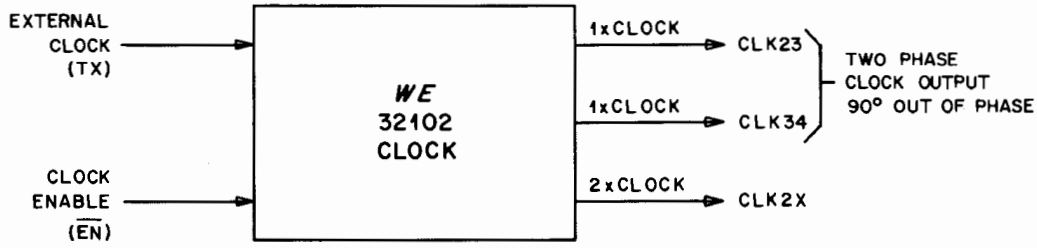


Figure 1. WE® 32102 Clock Functional Block Diagram

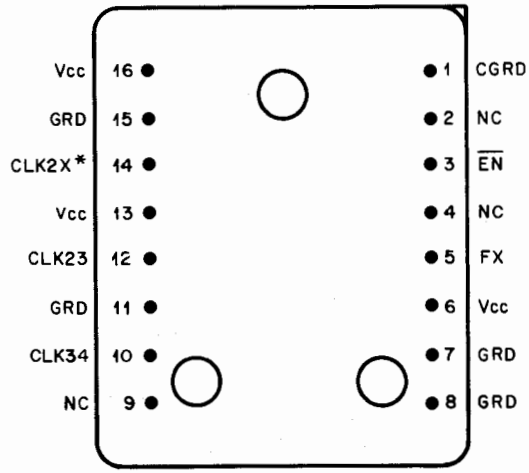
Description 223
Features 223
User Information 224
Pin Descriptions 225
Characteristics 226
Timing Characteristics 226
Electrical Characteristics 230

User Information

The *WE* 32102 Clock has three CMOS-level outputs: CLK23, CLK34, and CLK2X. Each of these outputs is capable of driving up to 130 pF of capacitive loading. For the skew specifications, the maximum capacitance load mismatch of the two clock outputs (CLK23 and CLK34) is 5%. Two CMOS inputs are provided to control the outputs of the *WE* 32102 Clock: FX, an external frequency source input; and $\overline{EN}$, an external frequency enable input.

The external source must be twice the frequency of the *WE* 32102 Clock (i.e., 20-MHz for a 10-MHz clock). The external source connects to the FX input and controls the outputs when the $\overline{EN}$ input is driven low (GND). If an external clock is not present, the $\overline{EN}$ input is internally pulled high (Vcc). The timing information given in Tables 2 and 3 is valid only when an external clock is not in control. When an external clock is used, the values specified in Tables 2 and 3 may be increased by any error in the external frequency source. The values for the parameters defined below are given in Table 2.

- **Duty Cycle Error.** The duty cycle of each of the clock outputs may deviate from the ideal case (50%).
- **Skew.** The time between clock edges of the two different clock outputs may deviate from their nominal values. For a 10-MHz clock, CLK23 leads CLK34 by 90° (25 ns). This phase lead should not deviate more than ± 3.0 ns. For a 14-MHz clock, CLK23 leads CLK34 by 90° (17.8 ns). This phase lead should not deviate more than ± 2 ns. For an 18-MHz clock, CLK23 leads CLK34 by 90° (13.9 ns). This phase lead should not deviate more than ± 2 ns.
- **Jitter.** The period of each clock output may deviate from its ideal value.



Bottom View

* This pin can be used only with the 10- and 14-MHz device.

Figure 2. WE® 32102 Clock Pin Diagram

Pin Descriptions

Table 1. WE® 32102 Clock Pin Descriptions

Pin	Symbol	Type	Description
1	CGND	—	**Case Ground.** Common with circuit ground.
2	NC	—	**No Connection.**
3	$\overline{EN}$	Input	**External Frequency Enable.** Provides a multiplexing feature that allows an external oscillator to control the clock.* $\overline{EN}$ **Clock Output (CLK23 and CLK34)** High crystal frequency** Low external source frequency Where: 3.5 V $\leq$ high $\leq$ 5.25 V 1.0 V $\leq$ low $\leq$ −0.5 V
4	NC	—	**No Connection.**
5	FX	Input	**External Frequency Source.** Provides an external CMOS input for a separate 2X clock source.
6	VCC	—	**Supply Voltage.** 4.75 V to 5.25 V.
7	GND	—	**Ground.**
8	GND	—	**Ground.**
9	NC	—	**No Connection.**
10	CLK34	Output	**1X Frequency Output.** 90° phase lag with respect to CLK23.
11	GND	—	**Ground.**
12	CLK23	Output	**1X Frequency Output.** Leads CLK34 by 90°.
13	VCC	—	**Supply Voltage.** 4.75 V to 5.25 V.
14	CLK2X	Output	**2X Frequency Output.** Provides frequency double that of either CLK23 or CLK34. This pin is used only with the 10- and 14-MHz devices.
15	GND	—	**Ground.**
16	VCC	—	**Supply Voltage.** 4.75 V to 5.25 V.

* A system reset must be done either during or after any transition of the $\overline{EN}$ input to insure proper operation of chips driven by the clock.

** $\overline{EN}$ pulled high internally.

Characteristics

Timing Characteristics

Table 2. 10-, 14-, 18-, and 24-MHz Timing Specifications*

Symbol	Description		10-MHz	14-MHz	18-MHz	24-MHz	Unit
tSK1	1X clock skew between CLK23 and CLK34	max	3	2	2	2	ns
		nom	—	—	—	—	
		min	—	—	—	—	
	CLK23, CLK34 duty cycle	max	53.0	52.0	53.0	53.0	%
		nom	50.0	50.0	50.0	50.0	
		min	47.0	48.0	47.0	47.0	
tDC1	1X clock duty cycle error	max	3	1.4	1.4	1.7	ns
		nom	—	—	—	—	
		min	—	—	—	—	
tJR1	1X clock jitter	max	0.5	0.5	0.3	0.3	ns
		nom	—	—	—	—	
		min	—	—	—	—	
tSK2	2X clock skew (between leading edge of CLK34 and leading edge of CLK2X)	max	8	6	—	—	ns
		nom	—	—	—	—	
		min	—	—	—	—	
	2X clock duty cycle	max	55.0	55.0	—	—	%
		nom	50.0	50.0	—	—	
		min	45.0	45.0	—	—	
tDC2	2X clock duty cycle error	max	2.5	1.8	—	—	ns
		nom	—	—	—	—	
		min	—	—	—	—	
tJR2	2X clock jitter	max	4	3	—	—	ns
		nom	—	—	—	—	
		min	—	—	—	—	

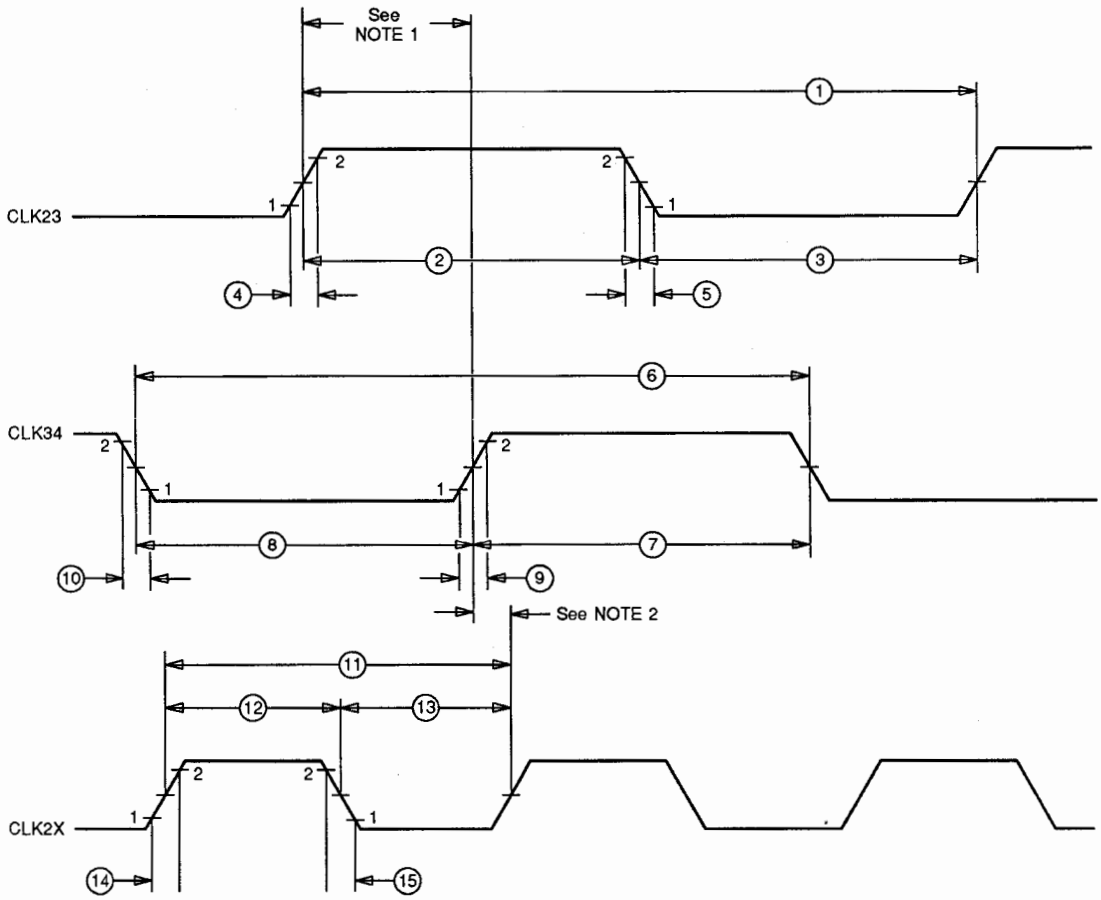
* Rise and fall times are measured between Vcc – 1.3 V and GND + 0.8 V regions and for capacitive loading 130 pF. Skew and duty cycles are measured at Vcc/2.0.

Table 3. 10-, 14-, 18-, and 24-MHz Timing Diagram Characteristics

Num	Symbol	Description		10-MHz	14-MHz	18-MHz	24-MHz	Unit
1	tC23HC23H	System clock 23 period	max	—	—	—	—	ns
			nom	100	71.4	55.6	41.7	
			min	—	—	—	—	
2	tC23HC23L	System clock 23 high width	max	53.0	37.1	29.5	22.1	ns
			nom	50.0	35.7	27.8	20.8	
			min	47.0	34.3	26.1	19.6	

Table 3. 10-, 14-, 18-, and 24-MHz Timing Diagram Characteristics (Continued)

Num	Symbol	Description		10-MHz	14-MHz	18-MHz	24-MHz	Unit
3	tC23LC23H	System clock 23 low width	max	53.0	37.1	29.5	22.1	ns
			nom	50.0	37.1	27.8	20.8	
			min	47.0	34.3	26.1	19.6	
4	tC23H1C23H2	System clock 23 rise time	max	4	4	4	4	ns
			nom	—	—	—	—	
			min	—	—	—	—	
5	tC23L2C23L1	System clock 23 fall time	max	4	4	4	4	ns
			nom	—	—	—	—	
			min	—	—	—	—	
6	tC34HC34L	System clock 34 period	max	—	—	—	—	ns
			nom	100	71.4	55.6	41.7	
			min	—	—	—	—	
7	tC34HC34L	System clock 34 high width	max	53.0	37.1	29.5	22.1	ns
			nom	50.0	35.7	27.8	20.8	
			min	47.0	34.3	26.1	19.6	
8	tC34HC34L	System clock 34 low width	max	53.0	37.1	29.5	22.1	ns
			nom	50.0	35.7	27.8	20.8	
			min	47.0	34.3	26.1	19.6	
9	tC34H1C34H2	System clock 34 rise time	max	4	4	4	4	ns
			nom	—	—	—	—	
			min	—	—	—	—	
10	tC34L2C34L1	System clock 34 fall time	max	4	4	4	4	ns
			nom	—	—	—	—	
			min	—	—	—	—	
11	tC2XHC2XH	2X clock period	min	—	—	—	—	ns
			nom	50.0	35.7	—	—	
			min	—	—	—	—	
12	tC2XHC2XL	2X clock high width	max	26.5	19.6	—	—	ns
			nom	25.0	17.8	—	—	
			min	23.5	16.0	—	—	
13	tC2XLC2XH	2X clock low width	max	26.5	19.6	—	—	ns
			nom	25.0	17.8	—	—	
			min	23.5	16.0	—	—	
14	tC2XH1C2HX2	2X clock rise time	max	4	4	—	—	ns
			nom	—	—	—	—	
			min	—	—	—	—	
15	tC2XL1C2XL2	2X clock fall time	max	4	4	—	—	ns
			nom	—	—	—	—	
			min	—	—	—	—	



[1] At 10-MHz, CLK23 leads CLK34 by 90° (25 ns). This phase lead should not deviate more than ± 3 ns.
At 14-MHz, CLK23 leads CLK34 by 90° (17.8 ns). This phase lead should not deviate more than ± 2 ns.
At 18-MHz, CLK23 leads CLK34 by 90° (13.9 ns). This phase lead should not deviate more than ± 2 ns.
At 24-MHz, CLK23 leads CLK34 by 90° (10.4 ns). This phase lead should not deviate more than ± 2 ns.

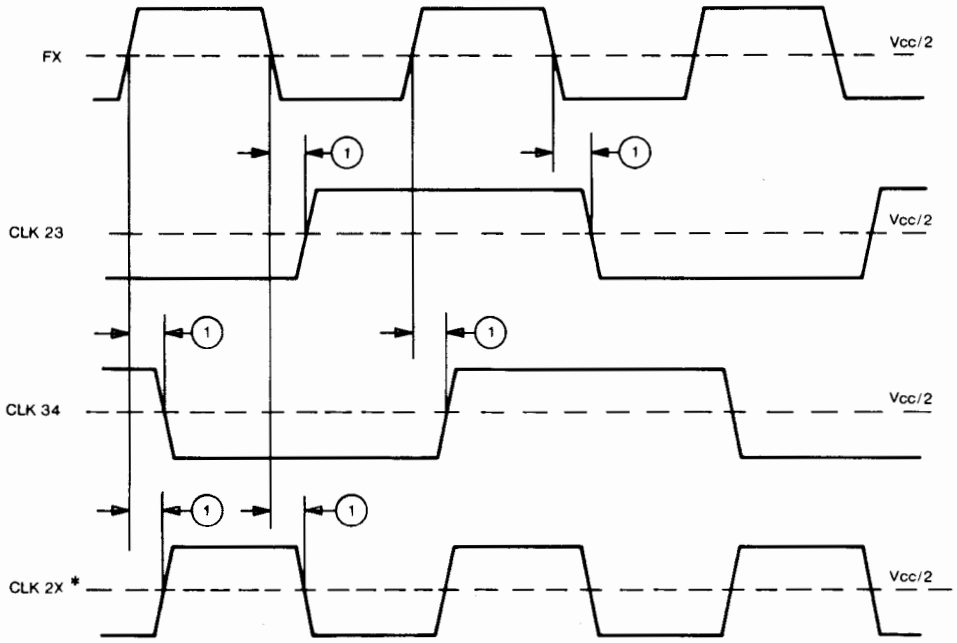
[2] At 10-MHz, CLK34 should not lead CLK 2X by more than 8 ns.
At 14-MHz, CLK34 should not lead CLK 2X by more than 6 ns.
At 18- and 24-MHz, there is no CLK2X output.

Figure 3. Timing Diagram

Table 4. 10-, 14-, 18-, and 24-MHz External Frequency Timing*

Num	Symbol	Description		10 MHz	14 MHz	18 MHz	24 MHz	Unit
1	tFXD	External frequency delay	max	90.0	65.0	65.0	25.0	ns
			nom	—	—	—	—	
			min	0	0	0	0	

* When $\overline{EN}$ is low, the clock outputs respond to input FX.



* Reference for 10- and 14-MHz devices only.

Figure 4. External Frequency Timing Diagram

Electrical Characteristics

Table 5. Operating Conditions

Parameter	Symbol	Min	Nom	Max	Unit
Supply voltage	V_{CC}	4.75	—	5.25	V
Output load capacitance	C_L	—	—	130	pF
Package case temperature	T_C	0	—	80	°C
Nominal operating frequency CLK23 and CLK34	F	10	—	24	MHz
Power dissipation	P	800	—	1.5	W
Long term stability (all outputs)*	—	–200	—	+200	ppm

* The long term stability includes calibration tolerance at 25 °C, operating temperature range, V_{CC} change, load change and aging.

Table 6. Output Electrical Characteristics

Case Temperature	Parameter	Min	Max	Unit
25 °C	Output sink current (I_{OL}) $V_{OL} \geq$ GND + .26 V	—	24	mA
	Output source current (I_{OH}) $V_{OH} \leq V_{CC}$ – .52 V	—	24	mA
80 °C	Output sink current (I_{OL}) $V_{OL} \geq$ GND + .33 V	—	24	mA
	Output source current (I_{OH}) $V_{OH} \leq V_{CC}$ – .66 V	—	24	mA

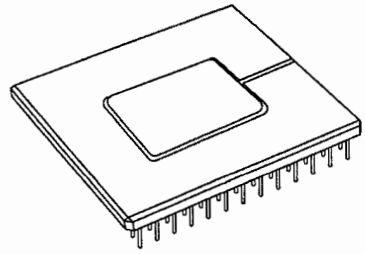
Table 7. Input Electrical Characteristics

Parameter		Min	Max	Unit
Loading current $V_{IH} = V_{CC}$	high-level	–1	1	mA
V_{IL} = GND	low-level	–1	1	mA
Input voltage for V_{CC} = 5 V at 25 to 80° C	high-level	3.5	5	V
	low-level	GND	1	V

WE® 32103 DRAM Controller

Description

The *WE* 32103 DRAM Controller provides address multiplexing, access and cycle time management, and refresh control for dynamic random access memory (DRAM). In a single chip, it provides the interface between high-performance dynamic memory subsystems and 32-bit microprocessor systems. The general-purpose memory interface is programmable to optimize the system performance for a wide range of memory configurations. The DRAM controller is capable of addressing up to 16 Mbytes of DRAM using 1 Mbit chips. The DRAM controller is available in 10-, 14-, and 18-MHz frequency versions; requires a single 5 V supply; and is in a 125-pin square, hermetic, ceramic pin grid array (PGA) package.



Features

- Overlaps memory accesses with MMU address translations
- Supports double- and quad-word memory transfers
- Full support for error detection and correction devices
- Supports dual-ported memory configurations using two DRAM controllers
- Drives up to 88 DRAM devices without external buffers
- Controls a variety of DRAM configurations of up to 1 Mbit devices
- Programmable support of page and nibble mode operations
- Programmable DRAM access times
- Internal refresh timer and refresh address counter
- General-purpose asynchronous operation and synchronous operation with the *UNIX* Microsystem

Description 231
Features 231
User Information 232
Microprocessor Interface 232
Early Access Initiation 232
Pretranslation 235
Addressing Assignment and Effects 235
Synchronous Operation 235
Asynchronous Operation 237
Multiword Data Transfers 237
Partial-Word Data Transfers 238
Dual-Port Configuration 238
Arbitration Scheme 239
Bus Exceptions 239
Faults 239
DRAM Interface 239
DRAM Initialization 239
Error Detection and Correction Interface .. 239
DRAM Timing 240
Data Acknowledge 240
Refresh 240
Reset 243
High-Impedance State 244
Clock Rate Independent Refresh Mechanism 244
DRAM Controller Registers 245
System Configuration Register (R0) 246
Refresh Configuration Register (R1) 248
RAS Timing Register (R2) 249
CAS Timing Register (R3) 250
Acknowledge Timing Register (R4) 251
Data Timing Register (R5) 252
Fault Register (R6) 253
Start Register (R7) 253
Fault Address Registers (R8—R10) 254
Refresh Interval Register (R11) 254
Pin Descriptions 254
Numerical Order 255
Functional Groups 258
Protocol Diagrams 263
Characteristics 281
Timing Characteristics 281
Timing Diagrams 286
Electrical Characteristics 308
Inputs 308
Outputs 308
Operating Conditions 309
Derating Curves 310

User Information

The *WE* 32103 DRAM Controller provides, in a single chip, all of the basic control necessary to interface DRAM devices in a 32-bit microprocessor system. This includes address multiplexing, row and column strobe timing, refresh timing, and acknowledge timing. In addition, the DRAM controller includes special features to provide a high performance memory interface for a wide variety of 32-bit systems. Figure 1 shows a functional block diagram of the DRAM controller.

Microprocessor Interface

Figures 2 and 3 show the DRAM controller system interface with and without an MMU. Figure 4 shows how the system address bus is typically partitioned by the DRAM controller.

Early Access Initiation. In a typical microprocessor system, the vast majority of all bus operations are memory accesses. Knowing this, the DRAM controller begins an access to memory as soon as it detects the beginning of a bus operation. The assertion of four inputs triggers the DRAM controller to begin a memory access before it is actually selected (chip select, $\overline{CS}$, asserted): address strobe ($\overline{AS}$), early cycle initiate ($\overline{EMCI}$), cycle initiate ($\overline{CYCLEI}$), and physical address strobe ($\overline{PAS}$). If $\overline{CS}$ does not arrive when expected after one of these signals is asserted, the DRAM controller terminates the access as quickly as possible without violating any DRAM timing specifications. This act-first, think-later mode of operation will result in a false start ($\overline{RASX}$ with or without $\overline{CASX}$ assertion) when an access is not to the DRAM controller address space (e.g., no $\overline{CS}$ generation). Accesses to the DRAM controller address space will result in a $\overline{CS}$ being generated (to the DRAM controller) allowing the pretranslated access to complete. In a typical system, most accesses will result in a $\overline{CS}$ generation to the DRAM controller. The DRAM controller will, however, wait for the assertion of data strobe ($\overline{DS}$) before asserting the column address strobes ($\overline{CAS0}$—$\overline{CAS3}$) on write operations. Data output drivers will not be enabled and an acknowledge will not be returned to the CPU until (and unless) the DRAM controller receives a $\overline{CS}$.

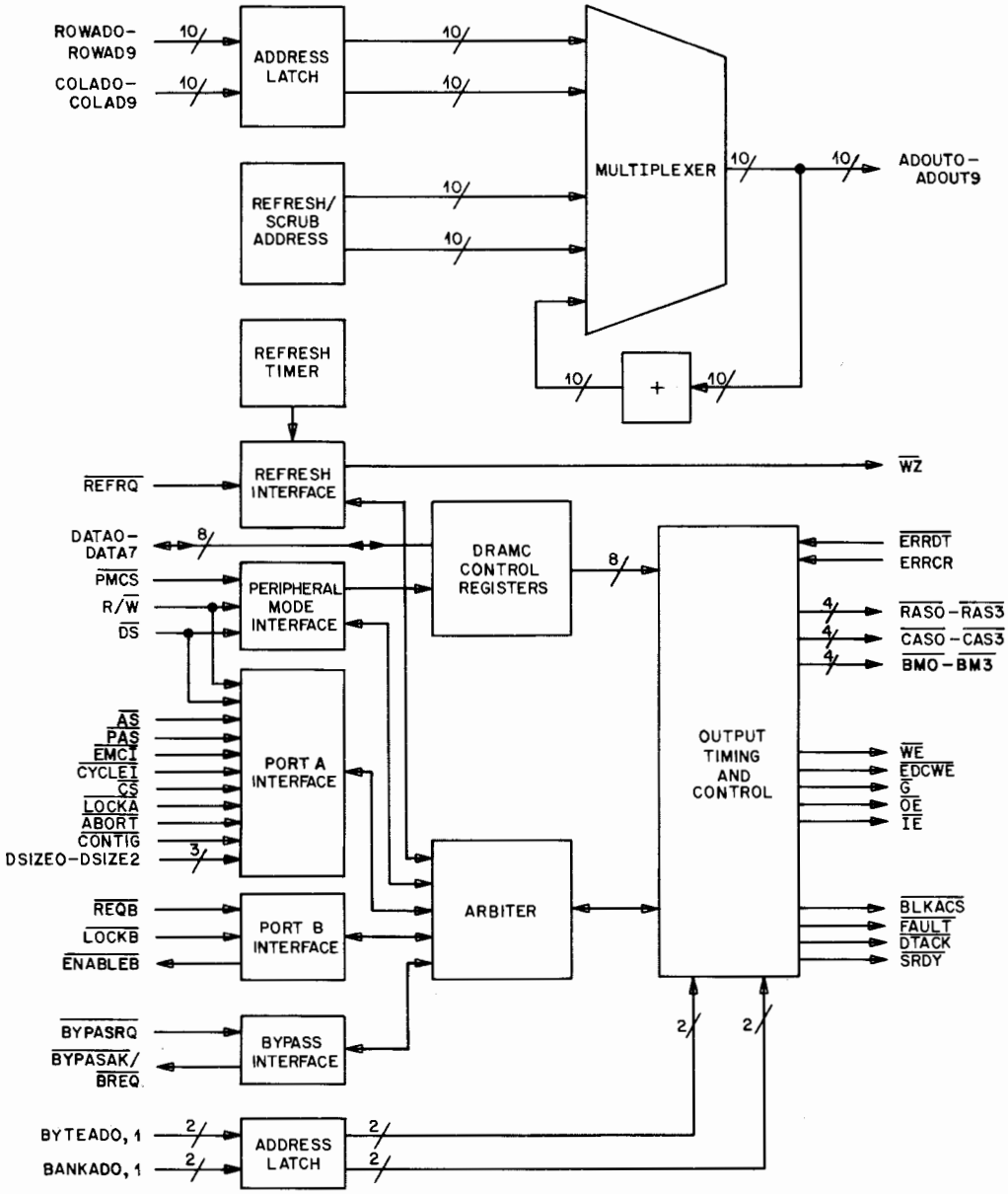
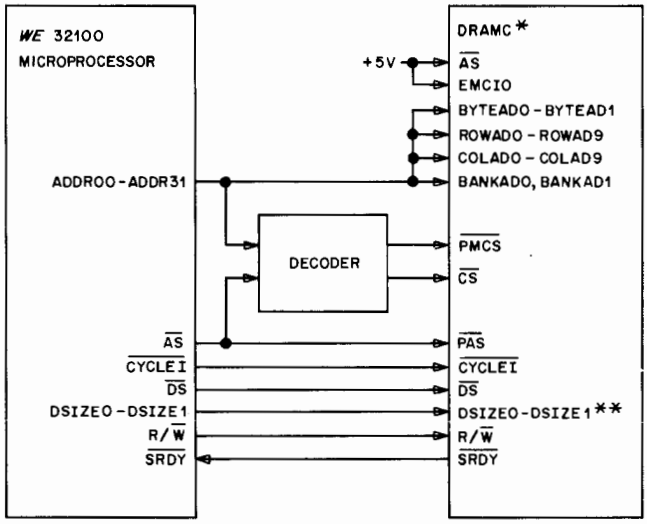
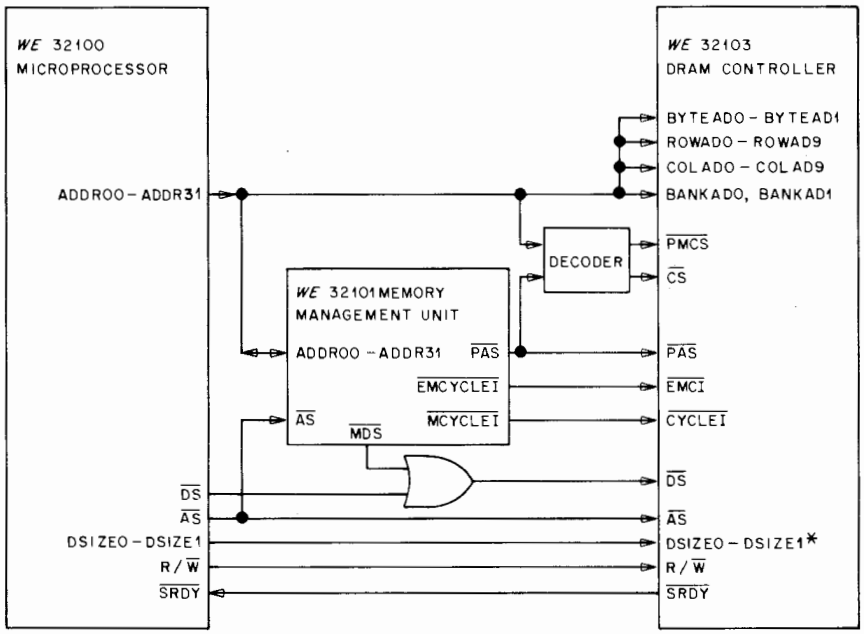


Figure 1. WE® 32103 DRAM Controller Functional Block Diagram



Note: CPU address bus connections to decoder and DRAMC are dependent on application.
* *WE* 32103 Dynamic Random Access Memory Controller.
** When a DMA Controller is used, DSIZE0–DSIZE1 connect to the DMA Controller.

Figure 2. DRAM Controller-CPU Interconnections for Single Port Operations



Note: CPU address bus connections to decoder and DRAMC are dependent on application.
* When a DMA Controller is used, DSIZE0–DSIZE1 connect to the DMA Controller.

Figure 3. DRAM Controller-CPU-MMU Interconnections

Pretranslation. The DRAM controller has the capability to begin a memory access before the memory management unit (MMU) has translated the virtual address. This early start is advantageous in virtual memory systems that use paged segments. In such a system, the low-order bits of the virtual address are the page offset, which remains unchanged after the address translation is completed. If the DRAM controller is programmed for pretranslation using the PREXLT bit in the system configuration register (R0), it drives the page offset to the DRAM as the row address (ROWAD0—ROWAD9) and asserts the row address strobes ($\overline{RAS0}$—$\overline{RAS3}$) in parallel with the MMU address translation. When the translation is complete, the DRAM controller multiplexes the column address (COLAD0—COLAD9) to the DRAM and asserts the column address strobes. This parallelism with the MMU typically saves one wait state as compared to waiting for the MMU to complete the address translation before starting the memory access.

Note: Pretranslation may be used only when the DRAM controller is operating synchronously.

If the MMU must perform miss-processing, or if the segment to be translated is contiguous, the contiguous segment input ($\overline{CONTIG}$) is asserted to inform the DRAM controller that the virtual row address driven out is not the same as the physical row address. The DRAM controller then negates the row address strobes and restarts the memory access in a normal manner. DRAM controllers configured as slave ports do not support the pretranslation mode and therefore wait until $\overline{ENABLEB}$ is asserted by the master before starting in accesses to memory.

Addressing Assignment and Effects. There is a distinct trade-off between the use of pretranslation and page size for virtual memory subsystems. If the physical memory access overlaps the MMU translation process, the DRAM controller requires that the row address be the same before and after MMU operations. Such is the case for page offsets in paged virtual memory systems. A page offset must provide the full width of the row address required by the DRAM memory array. When using higher capacity dynamic RAM devices, some of the ROWAD bits required may fall into virtual address bit fields remapped by the MMU. If pretranslation is desired for these larger DRAM sizes, the effective page size can be increased by software to map virtual addresses into physical addresses which do not remap these bits. The designer should be aware that in such a case these software modifications effectively increase the page size.

A similar tradeoff exists for systems using the page/nibble mode for multiword accesses. In page/nibble mode, the COLAD is incremented to access the next word of a multiword access. If pretranslation is desired and multiword accesses are to be performed, it is desirable for all the words to lie on the same page. This implies that the bits that are incremented should not be translated.

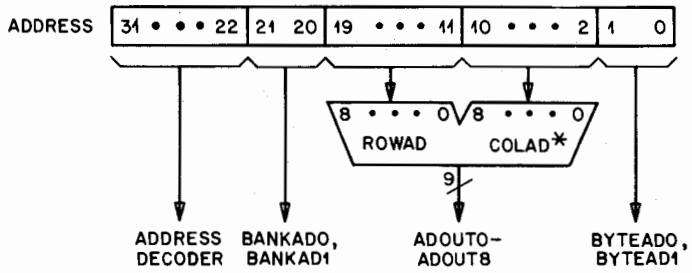
Each word should be in sequential column addresses in memory. This means that the two low-order bits of the COLAD input be the low-order bits of the address. With these facts in mind, the following rules should be used to implement pretranslation on quad-word block accesses while using the page/nibble (PAGNIB) mode access method on DRAMs:

- Append bits 2 and 3 of the virtual address to the low-order bits of COLAD, since, in PAGNIB mode, these bits will be incremented. This procedure will ensure that double- and quad-word accesses all lie on the same page.
- The ten ROWAD bits may be any combination of bits, as long as none are to be translated by the MMU and the number equals that required by memory.
- The remaining eight COLAD bits may be any combination of bits desired, as long as the total number of COLAD bits equals that required by the memory chip.

Depending on the configuration, PAGNIB mode and pretranslation might be visible to the programmer's model of the system.

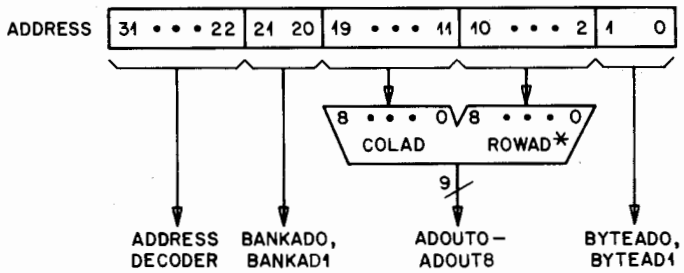
Systems using pretranslation but not supporting PAGNIB mode will cycle $\overline{RASX}$ when performing multiword accesses. The DRAM controller latches an internal version of the ROWAD that is incremented by one and used as the row address of the next word. The column address for this word is the address present at the COLAD input lines when $\overline{CASX}$ is asserted.

Synchronous Operation. When operating with *UNIX* Microsystem devices, the DRAM controller can operate synchronously to maximize



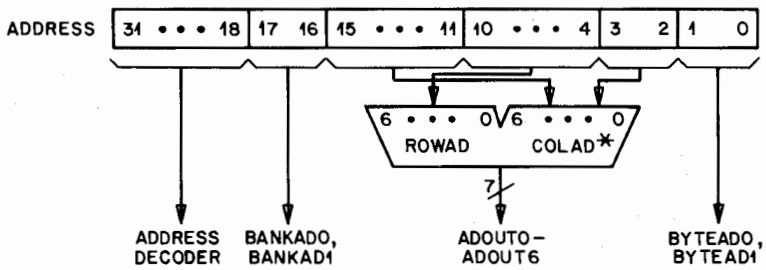
Note: For memory configurations using PAGNIB mode, the lower address bits should be connected to COLAD and the upper bits should be connected to ROWAD. When a multiword access is performed, COLAD will be incremented.
* Double- and quad-word accesses increment COLAD.

A. Typical Address Bus Interconnection with No Early RAS and PAGNIB (256K DRAM)



Note: For memory configuration using pretranslation, the lower bits must be connected to ROWAD since these bits will be placed on ADOUT and $\overline{RASX}$ asserted while the MMU translates the upper bits. In multiword accesses, ROWAD will be incremented.
* Double- and quad-word access increment ROWAD.

B. Typical Address Bus Interconnection with Early RAS and No PAGNIB (256K DRAM)



Note: For memory configuration using both PAGNIB and pretranslations, the above configuration is appropriate since the ROWAD bits must not be translated by the MMU, and the COLAD bits must be the lower memory address bits for incrementing to the next word on multiword accesses.
* Double- and quad-word access increment COLAD.

C. Typical Address Bus Interconnection with Early RAS and PAGNIB (16K DRAM)

Figure 4. Typical Address Bus Partitioning

its performance both by starting the access earlier and by returning a synchronous data acknowledge ($\overline{SRDY}$) to signal completion of the access. The DRAM controller begins a synchronous memory access as soon as it detects a cycle initiate ($\overline{CYCLEI}$) or an early cycle initiate ($\overline{EMCI}$) signal. The DRAM controller knows that the row address will be valid at the DRAM by the time $\overline{RASX}$ becomes valid. If $\overline{CS}$ does not arrive within one half cycle after physical address strobe ($\overline{PAS}$) arrives, the access is terminated and the DRAM controller prepares itself for the next access.

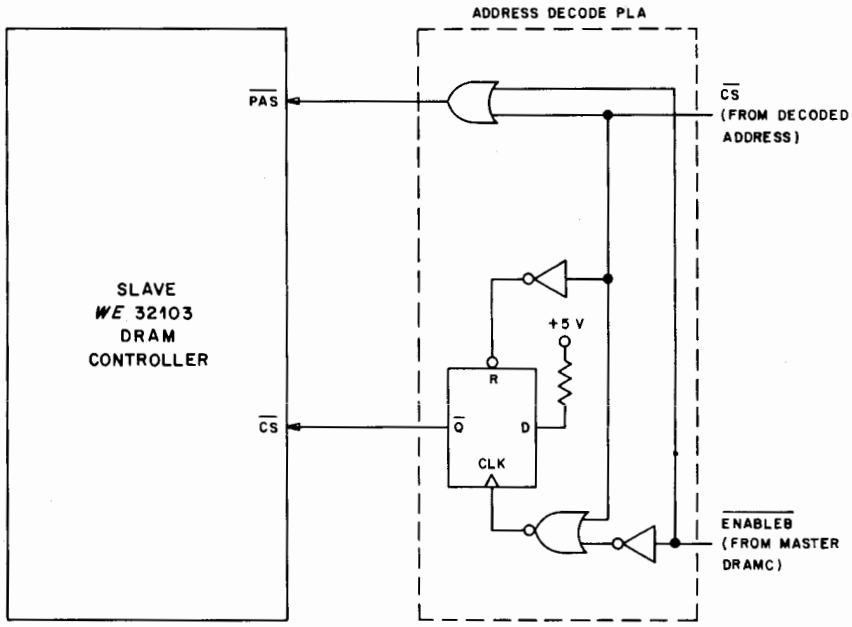
Asynchronous Operation. Even without a synchronous interface, the DRAM controller starts the access early by beginning as soon as it detects a valid $\overline{PAS}$. This early access saves the delay incurred by the address decoder in generating $\overline{CS}$. If its $\overline{CS}$ is not asserted after one half cycle, the access is terminated. The accesses that do complete because of $\overline{CS}$ assertion (most of them) end with the return of an asynchronous acknowledge to the bus master ($\overline{DTACK}$).

Multiword Data Transfers. The three data size inputs (DSIZE0—DSIZE2) are decoded to determine the size of the requested data transfer. The DRAM controller can support one, two, and four word transfers as well as one, two, and three byte accesses. Multiword transfers are recognized only for synchronous requests by asserting the block access output ($\overline{BLKACS}$) along with $\overline{SRDY}$. Asynchronous multiword requests are treated as single word requests and $\overline{BLKACS}$ is not asserted. When the DRAM controller is programmed for PAGNIB mode, the COLAD will always be modified for the next word of a multiword access. Nibble mode DRAMs have special requirements. Designers wishing to use the ports should refer to the application note **Interfacing the WE® 32103 DRAM Controller With Nibble Mode Memories.**

If pretranslation is used and PAGNIB mode is disabled, then ROWAD will be incremented for the next word of a multiword access. If both pretranslation and PAGNIB modes are used, multiword accesses will increment COLAD. For all cases described above, the address of the DRAM devices (either ROWAD or COLAD) is normally incremented by four bytes. The exception to this is double-word accesses that are not double-word aligned (ADDR02 = 1). In this case, the address to the DRAMs is decremented by four bytes as expected by the CPU. (See Figure 4.)

Table 1. Byte Mark Assertions

No Error Detection and Correction (EDC) Chip Present	
Access Type	**Byte Mark Assertion**
Full word read	1/2 cycle after $\overline{RASX}$ assertion.
Partial word read	1/2 cycle after $\overline{RASX}$ assertion.
Full word write	1/2 cycle after $\overline{RASX}$ assertion.
Partial word write	1/2 cycle after $\overline{RASX}$ assertion.
Error Detection and Correction (EDC) Chip Present	
Access Type	**Byte Mark Assertion**
Full word read	None
Partial word read	None
Partial word write	None during the read portion of the read modify write cycle. 1/2 cycle after $\overline{IE}$ assertion during the write portion of the read modify write cycle.
Full word write	1/2 cycle after $\overline{RASX}$ assertion.



Note: This circuit can easily fit within an address decoding PLA.

Figure 5. Dual-Port Circuit for $\overline{PAS}$ and $\overline{CS}$ Generation for Slave DRAM Controller

Partial-Word Data Transfers. The DRAM controller treats all partial-word reads as word reads by accessing all four bytes of the word. Partial-word writes select only the bytes to be written by asserting $\overline{BM0}$—$\overline{BM3}$, starting at the BYTEAD address and extending to as many bytes as specified by DSIZE0—DSIZE2. The bytes being accessed are assumed to never cross a word boundary. If the DRAM Controller is configured for EDC, partial word writes will be performed as read-modify-write cycles to guarantee check bit validity.

Table 1 lists byte mark assertions for various accesses and error detection and correction (EDC) configurations.

Dual-Port Configuration. A dual-ported memory system is configured by using two DRAM controllers. One is programmed as the master and performs the arbitration for the memory in addition to all refresh operations.

The other DRAM controller is programmed as the slave. It normally 3-states its outputs to the memory, avoiding bus contention with the master DRAM controller. When the master receives a request from the slave ($\overline{BREQ}$) to access the memory via its request port B ($\overline{REQB}$) input, it completes any operation in progress, then 3-states its memory outputs (all of the strobes are high at this point), and enables the slave. A bypass mode is also provided which allows the CPU on the master system bus to talk directly to slave peripherals to initialize them or issue commands. To do this, the master DRAM controller receives a bypass request ($\overline{BYPASRQ}$) from the master-port bus master and issues a bypass acknowledge ($\overline{BYPASAK}$) to grant the request to the slave bus arbiter. Figure 6 shows two DRAM controllers connected in a dual-port system.

The $\overline{BYPASAK}$/$\overline{BREQ}$ pin has different functions, depending on how the DRAM controller is configured. If the DRAM controller is configured as a master, the pin provides the $\overline{BYPASAK}$ function. When the controller is configured as a slave, the pin provides the $\overline{BREQ}$ function.

When using dual-ported configurations, if the slave-port master is any device which does not insert a dead cycle between accesses (e.g., the *WE* 32104 DMA Controller operating in burst mode), $\overline{\text{ENABLEB}}$ from the master should be qualified with slave DRAM controller $\overline{\text{CS}}$ to generate the slave DRAM controller $\overline{\text{PAS}}$. In addition, the $\overline{\text{CS}}$ on the slave side should be asserted when $\overline{\text{ENABLEB}}$ is inactive.

A typical circuit which may be used on the slave side in dual-ported configurations is illustrated on Figure 5.

Arbitration Scheme. If there is more than one request pending, the master enables the requester on the basis of the following priority scheme:

1. Refresh request (internal or external).
2. Port A accesses. While idle and not in the quiescent state, the DRAM controller enables port A. If while in this state a refresh is detected at the same time that $\overline{\text{AS}}$, $\overline{\text{PAS}}$, or $\overline{\text{CYCLE}}$ start a port A access, port A will be allowed to complete the access before servicing the refresh request.
3. Request port B ($\overline{\text{REQB}}$).
4. Peripheral mode chip select ($\overline{\text{PMCS}}$).
5. Bypass request ($\overline{\text{BYPASRQ}}$) – refresh may occur simultaneously.

Bus Exceptions. The DRAM controller allows memory accesses to be terminated abnormally by the bus master while maintaining the integrity of the data. Examples of bus exceptions are a $\overline{\text{FAULT}}$ assertion by some other peripheral in the system or a request for the bus master to relinquish the bus to another bus master. In the event of a bus exception, the DRAM controller terminates any access in progress while continuing to guarantee the minimum set-up and hold times of the DRAM address, with respect to row and column address strobes, along with the minimum pulse widths of the strobes.

Faults. If the DRAM controller detects an uncorrectable data error during a memory access, it returns $\overline{\text{FAULT}}$ to the CPU, sets the $\overline{\text{FAULT}}$ bit of the fault register (R6), and saves in the fault address register (R8—R10) the address that was being accessed. Fault handling the DRAM controller with peripheral mode reads to determine if the DRAM controller generated the fault and, if so, read the faulted address.

DRAM Interface

Figures 6 through 9 show the interfacing for several different configurations with and without error detection and correction circuitry.

Note: To avoid transmission line undershoot at the memory array, it is recommended that either an external buffer or an external series termination resistor be used on all DRAM controller outputs to the DRAM. The value of the series termination resistor should be equal to the characteristic line impedance (typically 33 Ω).

DRAM Initialization. After the configuration registers are initialized and the START bit in register 7 is set, the master DRAM controller asserts the row address strobes ($\overline{\text{RAS0}}$—$\overline{\text{RAS3}}$) eight times to "warm-up" the DRAMs – preparing them for operation. If configured with an EDC (error detection and correction) unit, the DRAM controller can be programmed by setting the memory initialization (MEMINIT) bit of system configuration register (R0) to proceed immediately from the warm-up sequence to a memory initialization sequence. The memory initialization writes a zero to all locations in memory with the appropriate check bits. The DRAM controller provides an output signal ($\overline{\text{WZ}}$) to instruct the EDC chip to output all zeros on its data outputs and to output the appropriate check bits. Since writing to all locations is relatively slow (one to two seconds with a 10-MHz clock), this memory initialization can be skipped, if desired.

Error Detection and Correction (EDC) Interface. The DRAM controller provides all of the control signals necessary to interface with EDC circuitry. During memory reads, the DRAM controller monitors the EDC output signals for errors in the data. If a single bit error occurs, the DRAM controller extends the memory access long enough to write the corrected data back to the DRAM before returning an acknowledge to the CPU. Except for the addition of a few wait states, this operation is entirely transparent to the CPU. If a multiple bit error occurs, the

EDC is unable to correct the data. In this case, DRAM controller sets the $\overline{\text{FAULT}}$ bit of the fault register, latches the faulted address in the fault address register, and returns $\overline{\text{FAULT}}$ to the CPU.

The DRAM controller can be programmed by using the ERRSCRB bit of the refresh configuration register (R1) to perform error scrubbing during refresh (see Table 5). In this mode, each time a row is refreshed, a column is selected to be read. If there is a correctable error in the data, it is corrected and written back to the DRAM. This option allows all memory locations to be scrubbed over a period of minutes with only minimal performance penalties to the system (lowering further the probability of encountering an uncorrectable parity error). The use of this error scrubbing feature is described in the *UNIX* Microsystem application note **Error Scrubbing with the WE® 32103 DRAM Controller.**

Partial word writes are treated specially for systems with EDC circuitry. The entire word is read from memory, checked for errors, the byte or bytes to be written are enabled onto the bus, new check bits for the entire word are generated, and finally, the new word is written back to DRAM.

DRAM Timing. The most important timing parameters of a DRAM system can be programmed in half-cycle increments by writing to the DRAM controller's configuration registers in peripheral mode. These include minimum pulse widths for the row and column addressing strobes, and delays for data movement and parity generation. These parameters will be described in more detail when the DRAM controller's registers are described.

Data Acknowledge. Several options are available for returning an acknowledge to the CPU. The delay from the assertion of the column address strobes to the return of an acknowledge ($\overline{\text{SRDY}}$ or $\overline{\text{DTACK}}$) can be optimized for the particular system configuration by programming the acknowledge timing register (R4) in the DRAM controller.

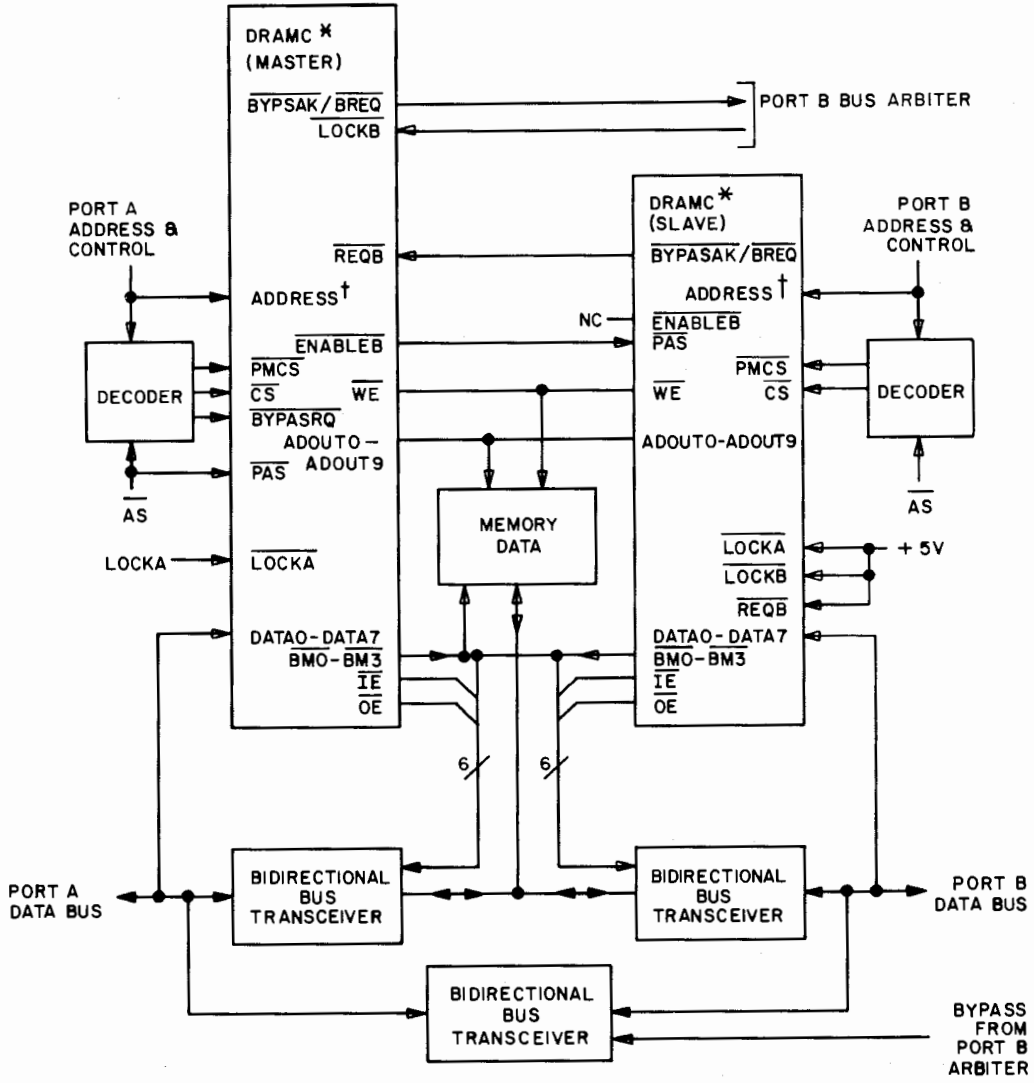
If there is no error checking on the data, the delay of waiting for error detection output can be bypassed and an acknowledge can be returned soon after $\overline{\text{CASX}}$ assertion in anticipation of the arrival of data from the DRAM. Since the DRAM controller checks its error detected ($\overline{\text{ERRDT}}$) input before the data is actually valid, that input must be tied high.

If there is simple parity checking in which all parity errors are uncorrectable, the parity $\overline{\text{FAULT}}$ output can be wired directly to the CPU's $\overline{\text{FAULT}}$ input and the DRAM controller acknowledge can be timed to arrive no sooner than the arrival of a valid parity output. Since, in this case, the DRAM controller does not generate the $\overline{\text{FAULT}}$ signal, it will never set its $\overline{\text{FAULT}}$ bit or latch the faulted address.

Full error checking by the DRAM controller requires that the acknowledge delay allow sufficient time for the DRAM controller to sample a valid parity signal at $\overline{\text{ERRDT}}$ before asserting an acknowledge.

Refresh. The DRAM controller can be programmed to provide a variety of refresh schemes using the refresh control register (R1). It has both a programmable internal refresh timer and an external refresh request input Comginations of internal, external, and distributed mode options allow several different modes:

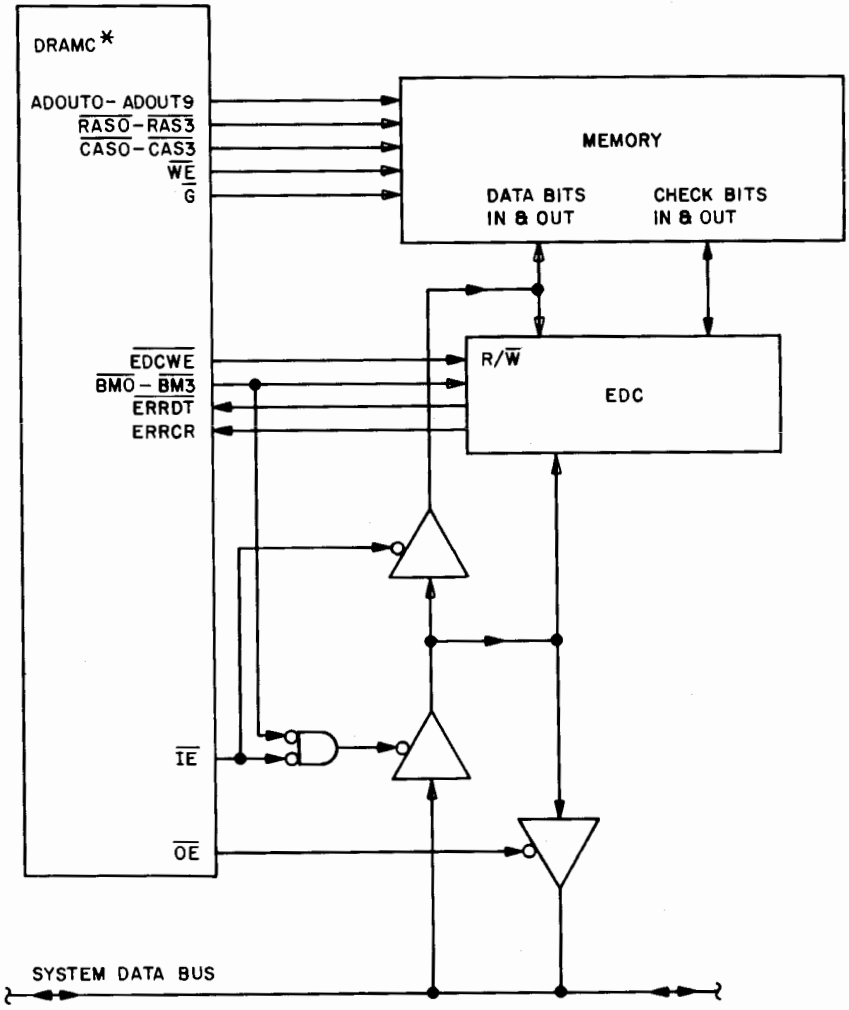
- **No Refresh** – Specific applications can access all rows of the DRAM chips at a frequent enough rate that no other refreshing is required (e.g., using memory as a CRT frame buffer, video refresh may be frequent enough to satisfy DRAM refresh requirements). In this case, the AUTORF bit should remain cleared and the input pin $\overline{\text{REFRQ}}$ tied high.
- **External Refresh** – The external refresh request input is always monitored by the DRAM controller. If the AUTORF bit is cleared, all refresh operations are initiated by this external request.



* *WE* 32103 DRAM Controller.

† Address includes BYTEADO, BYTEAD1, ROWAD0—ROWAD9, COLAD0—COLAD9, BANKAD0, and BANKAD1.

Figure 6. DRAM Controller Dual-Port Configuration Using Two DRAM Controllers



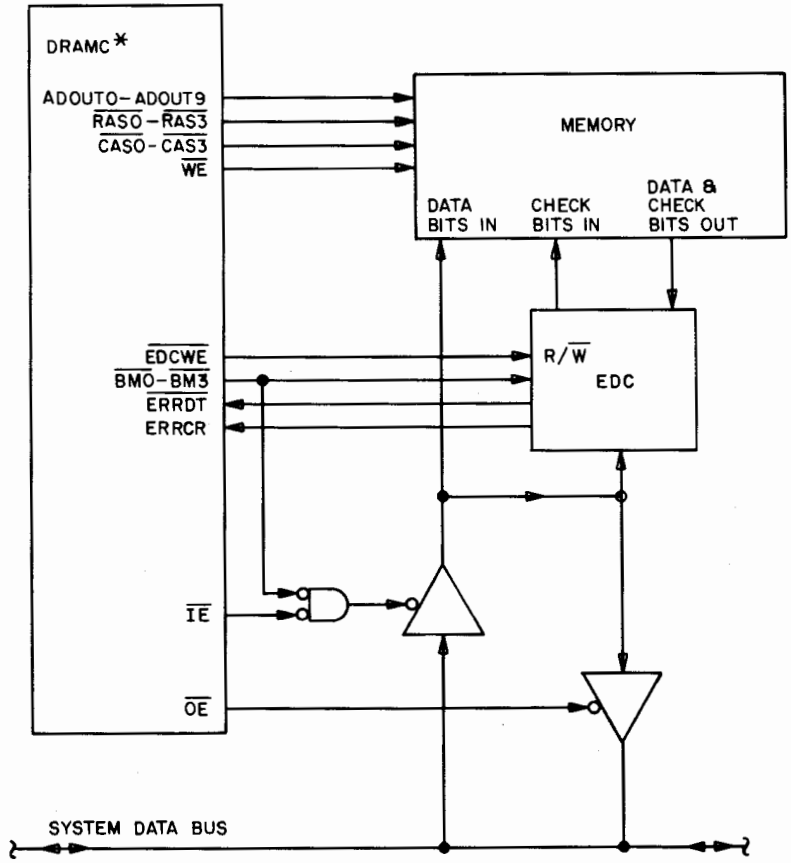
* *WE* 32103 DRAM Controller.

Figure 7. EDC Interface (x4 DRAM)

- **External Refresh With Failsafe** – If the AUTORF bit is set, a refresh operation can be initiated by either the internal timer or the external pin, whichever comes first. Every time the external request is asserted, the internal timer is reset. This mode can be used to force the DRAM controller to perform refresh while the system bus is idle (and no memory requests are pending). If the bus remains active at a rate long enough for the external refresh request not to occur, the internal timer will time out and force a refresh cycle.
- **Internal Refresh** – If $\overline{\text{REFRQ}}$ is strapped high and AUTORF is set, all refresh operations will result from the internal refresh timer timing out.

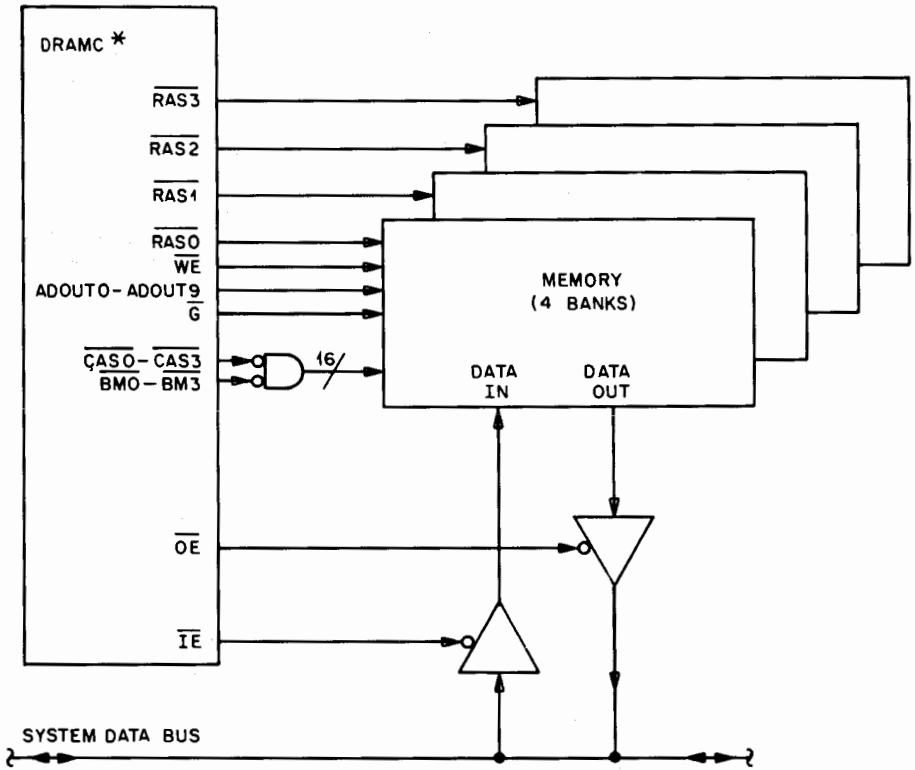
Reset

$\overline{\text{RESET}}$ assertion puts the DRAM controller into the quiescent state and clears all of the DRAM controller registers. Table 2 gives the state of the outputs while the DRAM controller is in the quiescent state and not executing a peripheral mode access.



* *WE* 32103 DRAM Controller.

Figure 8. EDC Interface (x1 DRAM)



* *WE* 32103 DRAM Controller.

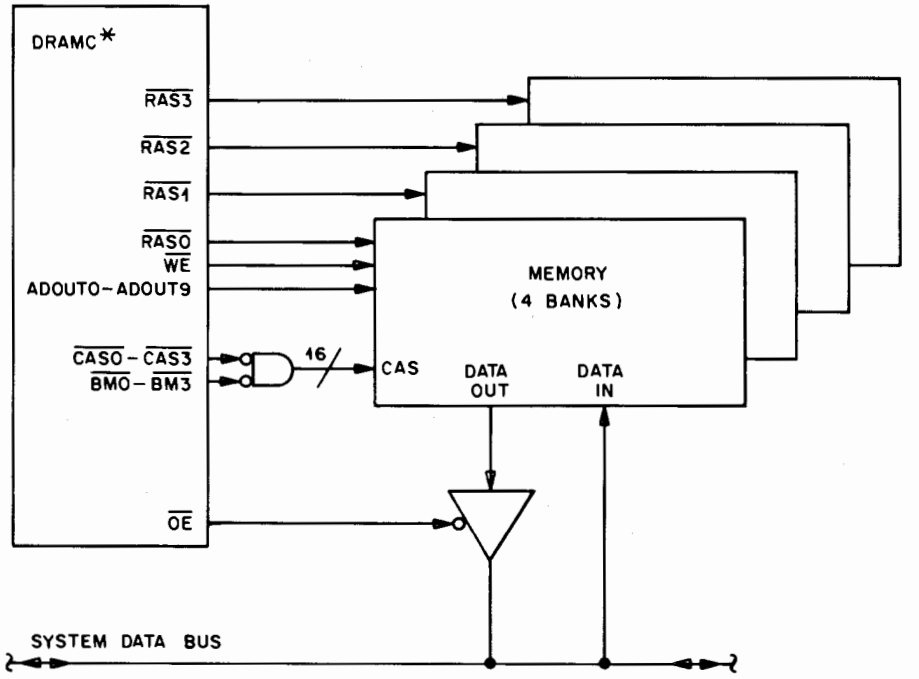
Figure 9. DRAM Controller-Memory Interconnection Without EDC (x4 DRAM)

High-Impedance State

If HIGHZ is asserted, the following DRAM controller outputs are 3-stated: ADOUT0—ADOUT9, CAS0—CAS3, RAS0—RAS3, BM0—BM3, G, EDCWE, WE, and IE.

Clock Rate Independent Refresh Mechanism

Register 11 contains an explicit refresh interval count. This count is loaded as an initial value for a down counter. The counter is decremented every fourth clock cycle. When the count reaches zero, a request is made to the on-chip access request arbiter and the explicit refresh interval count value is reloaded into the down counter. Any refresh request that runs into access contention remains pending until the DRAM controller completes the current access, at which time the refresh operation is performed.



* *WE* 32103 DRAM Controller.

Figure 10. DRAM Controller-Memory Interconnection Without EDC (x1 DRAM)

DRAM Controller Registers

The DRAM controller has eleven 8-bit registers that can be accessed in peripheral mode as given in Table 3. Registers 0 through 5 specify the memory parameters and system environment in which the DRAM controller operates. Register 6 holds a flag that is set when the DRAM controller asserts $\overline{\text{FAULT}}$. Register 7 contains a START bit which when set takes the DRAM controller out of its quiescent mode. Registers 8 through 10 hold the address of a faulted access. Register 11 contains an explicit refresh interval count. This count is loaded as an initial value for a down counter. The counter is decremented every fourth DRAM controller system clock cycle. When the count reaches zero, a request is made to the on-chip access arbiter and the explicit refresh interval count value is reloaded into the down counter. Any refresh request which runs into access contention remains pending until the DRAM controller completes the current access, at which time the refresh operation is performed.

The values of the registers are undefined on power up. $\overline{\text{RESET}}$ sets the DRAM controller registers to all zeros. Undefined registers return zero when read. Undefined bits within the configuration registers, unless otherwise noted, are writable and readable; however, these bits are reserved for future use and should not be used. They should be written with zeros. Data returned when reading register 7 provides the value of the start bit (0). All other bits are undefined and may be any combination of ones or zeros.

Table 2. Output States at Quiescent State

Signal	State	Signal	State
ADOUT0—ADOUT9	Z	$\overline{\text{ENABLEB}}$	Logic 1
$\overline{\text{BLKACS}}$	ZO	$\overline{\text{FAULT}}$	ZO
$\overline{\text{BM0}}$—$\overline{\text{BM3}}$	Z	$\overline{\text{G}}$	Z
$\overline{\text{BYPASAK}}$, $\overline{\text{WZ}}$	Logic 1	$\overline{\text{IE}}$	Z
$\overline{\text{CAS0}}$—$\overline{\text{CAS3}}$	Z	$\overline{\text{OE}}$	*
DATA0—DATA7	Z	$\overline{\text{RAS0}}$—$\overline{\text{RAS3}}$	Z
$\overline{\text{DTACK}}$	ZO	$\overline{\text{SRDY}}$	ZO
$\overline{\text{EDCWE}}$	Z	$\overline{\text{WE}}$	Z

Notes:
Z = high impedance.
ZO = high impedance, open drain.
* = state is dependent on state of R/$\overline{\text{W}}$, $\overline{\text{CS}}$, and $\overline{\text{DS}}$.

System Configuration Register (R0). The system configuration register is used to program the DRAM controller for the system configuration that it will be operating in.

Table 3. Register Address Map

COLAD2—COLAD5 5 4 3 2	Register Number	Register Name
0 0 0 0	R0	System configuration
0 0 0 1	R1	Refresh configuration
0 0 1 0	R2	RAS timing
0 0 1 1	R3	CAS timing
0 1 0 0	R4	Acknowledge timing
0 1 0 1	R5	Data timing
0 1 1 0	R6	Fault indicator bit
0 1 1 1	R7	Start bit
1 0 0 0	R8	Faulted address
1 0 0 1	R9	Faulted address
1 0 1 0	R10	Faulted address
1 0 1 1	R11	Refresh interval
1 1 0 0	R12	Reserved*
1 1 0 1	R13	Reserved*
1 1 1 0	R14	Reserved*
1 1 1 1	R15	Reserved*

* Reserved registers must be written with zeros and are not readable.

Table 4. System Configuration Register (R0)

Bit	7	6	5 4	3	2	1	0
Field	MASTER	PREXL	NOBANK1,NOBANK0	PAGNIB	MEMINIT	EDC	—

Bit	Field	Contents	Description
0	—	—	Not used.
1	EDC	Error detection and correction unit present	When set, this bit indicates that an error detection and correction unit is present. This allows partial writes to be performed with a read-modify-write sequence and correctable errors to be corrected with a similar sequence. Byte marks remain unasserted during memory reads. When cleared, this bit indicates that an error detection and correction unit is not present.
2	MEMINIT	Memory initialization	When set, the DRAM controller asserts $\overline{\text{WZ}}$ and performs a series of writes to all memory locations immediately following the warm-up sequence. This sequence is completed before any bus master is allowed to access memory. When cleared, no memory initialization is performed. Bus masters may access memory immediately after the eight warm-up cycles.

Table 4. System Configuration Register (R0) (Continued)

Bit	Field	Contents	Description
3	PAGNIB	Page/nibble mode	When set, multiple word accesses take advantage of a DRAM's page/nibble mode feature for faster double- and quad-word transfers. In this mode, $\overline{RASX}$ remains asserted while updating the column address and reasserting $\overline{CASX}$ to access the next word. When cleared, $\overline{RASX}$ will be cycled for each word in a multiple word access. **Note:** Nibble mode DRAMs have special requirements. Designers wishing to use these ports should refer to the application note **Interfacing the WE® 32103 DRAM Controller With Nibble Mode Memories.**
4—5	NOBANK0, NOBANK1	Number of banks	This field specifies how many banks of memory are in the system. If there are fewer than four banks, $\overline{CASX}$ outputs are shared to provide a higher drive capability to the DRAM, as shown below.

NOBANK0, NOBANK1 5 4	Number of Banks	R8 BANKAD0, BANKAD1 7 6	$\overline{CAS0}$—$\overline{CAS3}$ Drive
0 0	1	x x	$\overline{CAS0}$—$\overline{CAS3}$ drive bank 1
0 1	2	x 0	$\overline{CAS0}$, $\overline{CAS1}$ drive bank 1
		x 1	$\overline{CAS2}$, $\overline{CAS3}$ drive bank 2
1 0	4	0 0	$\overline{CAS0}$ drives bank 1
		0 1	$\overline{CAS1}$ drives bank 2
		1 0	$\overline{CAS2}$ drives bank 3
		1 1	$\overline{CAS3}$ drives bank 4
1 1	ND	—	—

x = don't care ND = not defined

6	PREXLT	Pretranslation	When set, the DRAM controller begins memory operations when $\overline{AS}$ is asserted by driving the CPU's virtual address from ROWAD to ADOUT and asserting $\overline{RASX}$. The access is completed when the MMU finishes its translation and asserts $\overline{CYCLEI}$. If $\overline{CONTIG}$ is asserted at the same time as $\overline{CYCLEI}$, the controller negates $\overline{RASX}$, after meeting the minimum $\overline{RASX}$ assertion time, and restarts the access using the physical ROWAD. This mode typically reduces memory access time by one cycle, when the MMU is translating paged segments. When cleared, the DRAM controller will wait to begin memory operations until $\overline{PAS}$ is asserted. At this time, it drives the COLAD to ADOUT and asserts $\overline{RASX}$. **Note:** This bit is meaningless when a DRAM controller is configured as a slave. Also, this mode can be used only during synchronous operation.
7	MASTER	Master DRAM controller	When set, the controller is configured as a master and provides full arbitration and refreshing. When cleared, the controller is configured as a slave and responds only to memory and peripheral mode operations and 3-states its DRAM outputs when not accessing memory.

Refresh Configuration Register (R1). The refresh configuration register allows the DRAM controller to be programmed to provide a variety of refresh schemes. The refresh configuration is meaningful only when the MASTER bit in the system configuration register (R0) is set.

Table 5. Refresh Configuration Register (R1)

Bit	7	6	5	4 3	2	1 0
Field	AUTORF	—	ERRSCRB	—	—	NOROW1,NOROW0

Bit	Field	Contents	Description
0—1	NOROW0, NOROW1	Number of rows	This bit specifies how many rows are in the DRAM. (NOROW0, NOROW1 = **11** is provided for testing.) **Note**: This parameter is used during memory initialization and error scrubbing.

NOROW0, NOROW1	Number of Rows
1 0	
0 0	256
0 1	1024
1 0	512
1 1	8 rows and 4 columns

2	—	—	Not used. Must be written with zero.
3—4	—	—	Not used. Must be written with zero.
5	ERRSCRB	Error scrub	This bit determines if the DRAM controller will perform error scrubbing during refresh. This must be disabled if no EDC is present. When set, error scrubbing is enabled; when cleared, error scrubbing is disabled. See application note **Error Scrubbing with the WE® 32103 DRAM Controller.**
6	—	—	Not used. Must be written with zero.
7	AUTORF	Auto refresh	When this bit is set, the DRAMC performs refresh every time its interval refresh timer times out. When cleared, refreshes are performed only when the $\overline{\text{REFRQ}}$ pin is asserted.

RAS Timing Register (R2). The RAS timing register allows the minimum pulse width of $\overline{\text{RASX}}$ to be programmed in half cycle increments.

Table 6. RAS Timing Register (R2)

Bit	7 — 6	5 — 3	2 — 1	0
Field	TPR1,TRP0	TRAS2—TRAS0	TRAH1,TRAH0	TRP2

Bit	Field	Contents	Description
1—2	TRAH0, TRAH1	Minimum row address hold time	This bit field indicates the number of half cycles that the row address remains valid on ADOUT after $\overline{\text{RASX}}$ assertion. **TRAH0, TRAH1 (2 1) — No. Half Cycles** 0 0 — 1 0 1 — 2 1 0 — 3 1 1 — undefined
3—5	TRAS0—TRAS2	Minimum $\overline{\text{RASX}}$ assertion time	This bit field indicates the minimum time that $\overline{\text{RASX}}$ must remain asserted before being negated. **TRAS0—TRAS2 (5 4 3) — No. Half Cycles** 0 0 0 — 1 0 0 1 — 2 0 1 0 — 3 0 1 1 — 4 1 0 0 — 5 1 0 1 — 6 1 1 0 — 7 1 1 1 — undefined
0,6—7	TRP0—TRP2	Minimum row precharge time	This bit field indicates the minimum time that $\overline{\text{RASX}}$ must remain high before being asserted again. **TRP0—TRP2 (0 7 6) — No. Half Cycles** 0 0 0 — 1 0 0 1 — 2 0 1 0 — 3 0 1 1 — 4 1 0 0 — 5 1 0 1 — 6 1 1 0 — 7 1 1 1 — 8

CAS Timing Register (R3). The $\overline{CAS}$ timing register allows the minimum pulse width of $\overline{CASX}$ to be programmed in half cycle increments.

Table 7. CAS Timing Register (R3)

Bit	7 6	5 4	3 2	1	0
Field	TNCP1,TNCP0	TCAS1,TCAS0	TNCAS1,TNCAS0	TASC	TCASNG

Bit	Field	Contents	Description
0	TCASNG	$\overline{CAS}$ negation delay	This bit indicates the number of phases from the negation of $\overline{SRDY}$ until the negation of $\overline{CAS}$. When set, there is a three phase delay between the negation of $\overline{SRDY}$ and the negation of $\overline{CAS}$. When cleared, this delay is equal to one phase. A phase is one half the 2x clock period.
1	TASC	Column address set-up time	This bit indicates the number of half cycles between switching to column address and the assertion of $\overline{CAS}$. When set, this bit indicates that the delay between switching to column address and the assertion of $\overline{CAS}$ is two half cycles. When cleared, this delay is one half cycle.
2—3	TNCAS0, TNCAS1	Minimum nth $\overline{CAS}$ assertion time	This field is used in place of TCAS0, TCAS1 for all accesses after the first word of a multiword transfer. This allows the user to take advantage of the faster timing of page or nibble mode DRAMs. **TNCAS0, TNCAS1 (3 2) — No. Half Cycles** 0 0 — 1 0 1 — 2 1 0 — 3 1 1 — 4
4—5	TCAS0, TCAS1	Minimum $\overline{CAS}$ assertion time	This bit field indicates the minimum time that $\overline{CAS}$ must remain asserted on the first word of a single or multiple word access before being negated again. **TCAS0, TCAS1 (5 4) — No. Half Cycles** 0 0 — 2 0 1 — 3 1 0 — 4 1 1 — 5
6—7	TNCP0, TNCP1	Minimum column precharge time	This bit field indicates the minimum time that $\overline{CAS}$ must remain high before being asserted again. **TNCP0, TNCP1 (7 6) — No. Half Cycles** 0 0 — 1 0 1 — 2 1 0 — 3 1 1 — undefined

Acknowledge Timing Register (R4). The acknowledge timing register is used to program the number of half cycles that the DRAM controller will wait from the assertion of $\overline{CASX}$ until the assertion of either synchronous ready ($\overline{SRDY}$) or data transfer acknowledge ($\overline{DTACK}$).

Table 8. Acknowledge Timing Register (R4)

Bit	7 — 5	4 — 2	1 0
Field	AK1DEL2—AK1DEL0	AKNDEL2—AKNDEL0	—

Bit	Field	Contents	Description
0—1	—	—	Not used.
2—4	AKNDEL0—AKNDEL2	Nth acknowledge delay	This field specifies the number of half cycles from the nominal assertion of $\overline{CASX}$ until the assertion of $\overline{DTACK}$ or $\overline{SRDY}$ after the first word of a multiword access (providing assertion of $\overline{ERRDT}$ is not detected by then). If the access is synchronous, $\overline{SRDY}$ is asserted with the first rising edge of CLK23 after the acknowledge delay is satisfied. Write operations (with the exception of partial word writes with EDC) return $\overline{DTACK}$ or $\overline{SRDY}$ after $\overline{CASX}$ assertion without waiting for the acknowledge delay. This delay must include the sum of the $\overline{CASX}$ valid time, the memory access time from $\overline{CASX}$ assertion, data buffer delays, and any delay incurred by error checking circuitry. For asynchronous access, the $\overline{ERRDT}$ setup time must be included in the error checking circuitry delay (it need not be included for synchronous accesses because of the cycle structure of the chip).

AKNDEL0—AKNDEL2 (4 3 2)	No. Half Cycles	AKNDEL0—AKNDEL2 (4 3 2)	No. Half Cycles
0 0 0	1	1 0 0	5
0 0 1	2	1 0 1	6
0 1 0	3	1 1 0	7
0 1 1	4	1 1 1	8

5—7	AK1DEL0—AK1DEL2	Acknowledge delay	This field specifies the number of half cycles from the nominal assertion of $\overline{CASX}$ until the assertion of $\overline{DTACK}$ or $\overline{SRDY}$ for the first word of a single or multiword access (providing assertion of $\overline{ERRDT}$ is not detected by then). If the access is synchronous, $\overline{SRDY}$ is asserted with the first rising edge of CLK23 after the acknowledge delay is satisfied. Write operations (with the exception of partial word writes with EDC) return $\overline{DTACK}$ or $\overline{SRDY}$ after $\overline{CASX}$ assertion without waiting for the acknowledge delay. This delay must include the sum of the $\overline{CASX}$ valid time, the memory access time from $\overline{CASX}$ assertion, data buffer delays, and any delay incurred by error checking circuitry. For asynchronous access, the $\overline{ERRDT}$ setup time must be included in the error checking circuitry delay (it need not be included for synchronous accesses because of the cycle structure of the chip).

AK1DEL0—AK1DEL2 (7 6 5)	No. Half Cycles	AK1DEL0—AK1DEL2 (7 6 5)	No. Half Cycles
0 0 0	1	1 0 0	5
0 0 1	2	1 0 1	6
0 1 0	3	1 1 0	7
0 1 1	4	1 1 1	8

Data Timing Register (R5). The data timing register allows for the programming of delays for data transfers.

Table 9. Data Timing Register (R5)

Bit	7 — 5	4 — 3	2 — 1	0
Field	WRDEL2—WRDEL0	TTRIST1,TTRIST0	ECDEL1,ECDEL0	—

Bit	Field	Contents	Description
0—7	—	—	Not used.
1—2	ECDEL0, ECDEL1	Error correction delay	This bit field specifies the number of additional half cycles that the controller waits if $\overline{\text{ERRDT}}$ is asserted when the AK1DEL or AKNDEL time has been satisfied. Then, if ERRCR is high, the DRAM controller writes the corrected data back to the DRAM and issues $\overline{\text{DTACK}}$ or $\overline{\text{SRDY}}$. If ERRCR is low, the DRAM controller does not write back to memory, but issues $\overline{\text{FAULT}}$.

ECDEL0, ECDEL1 2 1	No. Half Cycles
0 0	2
0 1	3
1 0	4
1 1	5

Bit	Field	Contents	Description
3—4	TTRIST0, TTRIST1	3-state time	This field indicates the number of half cycles of delay from negation of $\overline{\text{G}}$ to assertion of $\overline{\text{IE}}$ for a "late write." This delay is necessary to allow a DRAM with a bidirectional data bus time to 3-state its data bus before allowing another chip to drive the bus.

TTRIST0, TTRIST1 4 3	No. Half Cycles
0 0	0
0 1	1
1 0	2
1 1	3

Bit	Field	Contents	Description
5—7	WRDEL0— WRDEL2	Write delay	For late writes in an EDC read-modify-write sequence, this bit field specifies the number of half cycles the DRAM controller waits between assertion of $\overline{\text{IE}}$ to assertion of $\overline{\text{WE}}$. For all other writes, the DRAM controller allows this specified number of half cycles after it detects $\overline{\text{DS}}$ assertion before asserting $\overline{\text{CASX}}$. This delay allows time for data to propagate to the DRAM inputs before strobing the DRAM for a write.

WRDEL0— WRDEL2 7 6 5	No. Half Cycles
0 0 0	1
0 0 1	2
0 1 0	3
0 1 1	4
1 0 0	5
1 0 1	6
1 1 0	7
1 1 1	8

Fault Register (R6). The fault register is used to indicate that a fault was generated by the DRAM controller.

Table 10. Fault Register (R6)

Bit	7	6	0
Field	FAULT	—	

Bit	Field	Contents	Description
0—6	—	—	Not used.
7	FAULT	Fault indicator bit	This bit will be set at the same time $\overline{\text{FAULT}}$ is asserted. It is provided to inform a fault handling routine that the DRAM controller caused the fault. It is the responsibility of the fault handling routine to clear the bit after a fault. The 7 low-order bits will be zeros when read in peripheral mode.

Start Register (R7). The start register is used to take the DRAM controller out of the quiescent state which it is in after reset.

Table 11. Start Register (R7)

Bit	7	1	0
Field	—		START

Bit	Field	Contents	Description
0	START	Start bit	This bit is cleared when the DRAM controller is reset. While this bit remains cleared, the DRAM controller remains in a quiescent state with the DRAM controls 3-stated, and the chip responding only for peripheral mode operations. Memory access requests to a quiescent DRAM controller will hang the system. It is the responsibility of the software designer to initialize the DRAM controller before accesses are permitted. When set, this bit indicates that the controller is in the active state.
1—7	—	—	Not used. This bit field is not writable and may contain zeros, ones, or a combination of both.

Fault Address Registers (R8—R10). At the same time $\overline{\text{FAULT}}$ is asserted, the faulted address are latched into R8 through R10, as shown, for the three fault registers. These are read-only registers.

7 6	5 0	
BANKAD1,BANKAD0	ROWAD9— ROWAD4	Register 8 (R8)
7 4	3 0	
ROWAD3—ROWAD0	COLAD9—COLAD6	Register 9 (R9)
7 2	1 0	
COLAD5—COLAD0	BYTEAD1, BYTEAD0	Register 10 (R10)

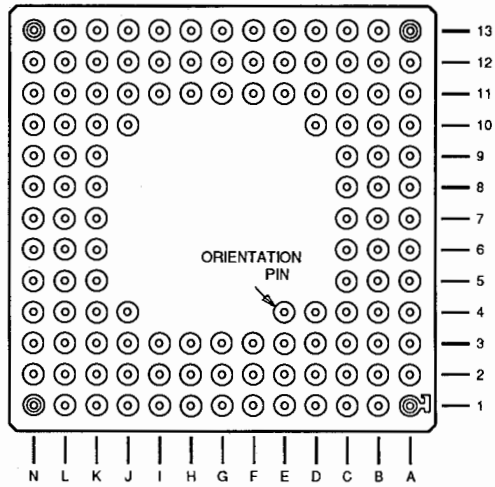
Figure 11. Fault Address Registers (R8—R10)

Refresh Interval Register (R11). The refresh interval register is used to explicitly specify the number of cycles that the DRAM controller is to wait between refresh operations. The interval is specified in increments of four (i.e., a value of one indicates an interval of four cycles, a value of two indicates an interval of eight cycles, etc.). This register must be written with the appropriate value before setting the start bit.

Pin Descriptions

The *WE* 32103 DRAM Controller is available in a 125-pin square, ceramic PGA package. Figure 12 and Tables 12—17 describe the pin assignments. A bar over a signal name indicates active as a logic 0.

The term *asserted* in the pin function description means that the signal is driven to its active state either by the DRAM controller (outputs) or an external device (inputs). The term *negated* means that the signal is driven to its inactive state. The 0 bit is the least significant bit for signals that occupy two or more pins (i.e., DSIZE0—DSIZE2). The signal type column is interpreted as an input (I), an output (O), or a bidirectional (I/O) signal.



Bottom View

Figure 12. WE® 32103 DRAM Controller 125-Pin Square, Ceramic PGA Package

Numerical Order

Table 12. WE® 32103 DRAM Controller Pin Descriptions

Pin	Name	Type	Description
A1	ROWAD8	I	Row address 8
A2	ROWAD7	I	Row address 7
A3	ROWAD6	I	Row address 6
A4	COLAD5	I	Column address 5
A5	ROWAD5	I	Row address 5
A6	COLAD4	I	Column address 4
A7	ROWAD4	I	Row address 4
A8	COLAD3	I	Column address 3
A9	ROWAD3	I	Row address 3
A10	COLAD2	I	Column address 2
A11	ROWAD2	I	Row address 2
A12	COLAD1	I	Column address 1
A13	ROWAD1	I	Row address 1
B1	COLAD8	I	Column address 8
B2	COLAD7	I	Column address 7
B3	COLAD6	I	Column address 6
B4	ADOUT7	O	Address out 7
B5	ADOUT6	O	Address out 6
B6	ADOUT5	O	Address out 5
B7	ADOUT4	O	Address out 4
B8	ADOUT3	O	Address out 3
B9	ADOUT2	O	Address out 2
B10	ADOUT1	O	Address out 1
B11	ADOUT0	O	Address out 0
B12	COLAD0	I	Column address 0
B13	ROWAD0	I	Row address 0
C1	DATA1	I/O	Data 1
C2	ADOUT8	O	Address out 8
C3	NC	—	No connection
C4	+5V	—	Power
C5	GND	—	Ground
C6	+5V	—	Power
C7	GND	—	Ground
C8	+5V	—	Power
C9	GND	—	Ground
C10	+5V	—	Power
C11	GND	—	Ground
C12	COLAD9	I	Column address 9
C13	ROWAD9	I	Row address 9
D1	DATA3	I/O	Data 3
D2	DATA0	I/O	Data 0
D3	NC	—	No connection
D4	NC	—	No connection
D10	GND	—	Ground
D11	+5V	—	Power

Table 12. WE® 32103 DRAM Controller Pin Descriptions (Continued)

Pin	Name	Type	Description
D12	ADOUT9	O	Address out 9
D13	DSIZE1	I	Data size 1
E1	DATA4	I/O	Data 4
E2	DATA2	I/O	Data 2
E3	GND	—	Ground
E4	—	—	Device socket orientation pin
E11	GND	—	Ground
E12	DSIZE0	I	Data size 0
E13	$\overline{\text{HIGHZ}}$	I	High-impedance
F1	DATA6	I/O	Data 6
F2	DATA5	I/O	Data 5
F3	NC	—	No connection
F11	+5V	—	Power
F12	DSIZE2	I	Data size 2
F13	$\overline{\text{RAS3}}$	O	Row address strobe 3
G1	$\overline{\text{LOCKA}}$	I	Lock request A
G2	NC	—	No connection
G3	GND	—	Ground
G11	GND	—	Ground
G12	$\overline{\text{RAS2}}$	O	Row address strobe 2
G13	$\overline{\text{RAS1}}$	O	Row address strobe 1
H1	$\overline{\text{REQB}}$	I	Request port B
H2	DATA7	I/O	Data 7
H3	+5V	—	Power
H11	+5V	—	Power
H12	$\overline{\text{RAS0}}$	O	Row address strobe 0
H13	$\overline{\text{CAS3}}$	O	Column address strobe 3
J1	$\overline{\text{BYPASAK}}/\overline{\text{BREQ}}$	O	Bypass acknowledgement/port B request
J2	$\overline{\text{LOCKB}}$	I	Lock request B
J3	NC	—	No connection
J11	GND	—	Ground
J12	$\overline{\text{CAS2}}$	O	Column address strobe 2
J13	$\overline{\text{CAS1}}$	O	Column address strobe 1
K1	$\overline{\text{PMCS}}$	I	Peripheral Mode Chip Select
K2	$\overline{\text{BYPASPRQ}}$	I	Bypass Request
K3	+5V	—	Power
K4	—	—	WARNING: This pin is for manufacture use only and must be tied high (+5 V).
K10	$\overline{\text{EMCI}}$	I	Early MMU cycle initiate
K11	+5V	—	Power
K12	$\overline{\text{CAS0}}$	O	Column address strobe 0
K13	$\overline{\text{ABORT}}$	I	Abort
L1	$\overline{\text{ENABLEB}}$	O	Enable port B
L2	$\overline{\text{REFRQ}}$	I	Refresh request

Table 12. WE® 32103 DRAM Controller Pin Descriptions (Continued)

Pin	Name	Type	Description
L3	GND	—	Ground
L4	+5V	—	Power
L5	GND	—	Ground
L6	+5V	—	Power
L7	CLK34	I	Clock 34
L8	CLK23	I	Clock 23
L9	GND	—	Ground
L10	+5V	—	Power
L11	GND	—	Ground
L12	$\overline{\text{CYCLEI}}$	I	Physical cycle initiate
L13	$\overline{\text{CONTIG}}$	I	Contiguous segment
M1	$\overline{\text{BM1}}$	O	Byte mark 1
M2	$\overline{\text{BM0}}$	O	Byte mark 0
M3	$\overline{\text{BM2}}$	O	Byte mark 2
M4	$\overline{\text{EDCWE}}$	O	EDC write enable
M5	$\overline{\text{WE}}$	O	Write enable
M6	ERRCR	I	Error corrected
M7	$\overline{\text{DTACK}}$	O	Data transfer acknowledge
M8	$\overline{\text{WZ}}$	O	Write zeros
M9	$\overline{\text{DS}}$	I	Data strobe
M10	BYTEAD1	I	Byte address 1
M11	$\overline{\text{CS}}$	I	Chip select
M12	BANKAD0	I	Bank address 0
M13	$\overline{\text{AS}}$	I	Virtual address strobe
N1	$\overline{\text{BM3}}$	O	Byte mark 3
N2	$\overline{\text{G}}$	O	DRAM output enable
N3	$\overline{\text{IE}}$	O	Input enable
N4	$\overline{\text{RESET}}$	I	Reset
N5	$\overline{\text{ERRDT}}$	I	Error detected
N6	$\overline{\text{SRDY}}$	O	Synchronous data ready
N7	$\overline{\text{FAULT}}$	O	Fault
N8	R/$\overline{\text{W}}$	I	Read/write
N9	$\overline{\text{OE}}$	O	Output enable
N10	$\overline{\text{BLKACS}}$	O	Block access
N11	BYTEAD0	I	Byte address 0
N12	$\overline{\text{PAS}}$	I	Physical address strobe
N13	BANKAD1	I	Bank address 1

Functional Groups

Table 13. Address and Data

Pin	Name	Type	Description
ADOUT0—ADOUT9	B11, B10, B9, B8, B7, B6, B5, B4, C2, D12	O	**Address Out.** These signals provide a multiplexed address to the DRAM array.
BANKAD0, BANKAD1	M12, N13	I	**Bank Address.** These bits select which $\overline{\text{CASX}}$ output(s) are enabled.
BYTEAD0, BYTEAD1	N11, M10	I	**Byte Address.** These are the lower two address input bits used to generate the byte marks.
COLAD0—COLAD9	B12, A12, A10, A8, A6, A4, B3, B2, B1, C12	I	**Column Address.** These address inputs provide the column address to the internal address multiplexer.
DATA0—DATA7	D2, C1, E2, D1, E1, F2, F1, H2	I/O	**Data Bus.** These bits are used to transfer data to and from the user accessible registers when in peripheral mode.
ROWAD0—ROWAD9	B13, A13, A11, A9, A7, A5, A3, A2, A1, C13	I	**Row Address.** These address inputs provide the row address to the internal address multiplexer.

Table 14. System Interface

Name	Pin(s)	Type	Function
$\overline{\text{ABORT}}$	K13	I	**Abort.** This signal, when asserted, indicates that the memory access is to be aborted (because the CPU has found the data in its instruction cache). As with any other termination of a memory access, the DRAM controller will guarantee the minimum assertion times of $\overline{\text{RASX}}$ and $\overline{\text{CASX}}$ before negating them (if they had been asserted) and the minimum hold times for the DRAM address. When the DRAM controller is used with bus masters other than the *WE* 32100 CPU, this input must be pulled high through a pull-up resistor.
$\overline{\text{AS}}$	M13	I	**Virtual Address Strobe.** When asserted, this strobe indicates that the virtual address is valid (at port A) and that the DRAM controller may begin a pretranslation access if programmed to do so. The DRAM controller latches $\overline{\text{AS}}$ synchronously on the rising edge of CLK23. This input must be tied high if there is no MMU in the system or if the DRAM controller is configured as the slave of a dual port configuration. This input should always be tied to the CPU's $\overline{\text{AS}}$ on DRAM controllers configured as a master if there is an MMU (even if the pretranslation feature is not used) in order to satisfy address hold requirements.
$\overline{\text{BLKACS}}$	N10	O	**Block Access.** This signal indicates that the DRAM controller can perform a double-or quad-word read/write operation as requested by the bus master. Block access is only allowed for synchronous memory transfers. If an asynchronous block request occurs, this signal is not asserted and the DRAM controller transfers only the first word of the request.

Table 14. System Interface (Continued)

Name	Pin(s)	Type	Function
$\overline{\text{BYPASAK}}$/ $\overline{\text{BREQ}}$	J1	O	**Bypass Acknowledgement, Port B Request.** This signal is used to indicate that the bypass request is being granted. $\overline{\text{BYPASAK}}$ remains asserted until $\overline{\text{BYPASRQ}}$ is negated. The DRAM controller may perform refresh during this time. When configured as a slave, this signal is used to make an access request to the master.
$\overline{\text{BYPASRQ}}$	K2	I	**Bypass Request.** This signal, when asserted, indicates a request to interconnect the master and slave buses in a dual-port environment. This mode is provided to enable the master to bypass the memory subsystem and talk directly to devices on the slave bus. It has the lowest priority among refresh, port A accesses, peripheral mode accesses, and slave mode accesses.
$\overline{\text{CONTIG}}$	L13	I	**Contiguous Segment.** This signal indicates that the physical address generated by the MMU is not the result of a paged translation. It means that the lower order bits (11 low-order bits for 2K pages) of the physical address may not be the same as the lower bits of the virtual address. The DRAM controller will negate $\overline{\text{RASX}}$, after meeting the $\overline{\text{RASX}}$ assertion time, if it has been asserted for a pretranslation access. It will then restart the access with the new physical address. This signal should be tied high if the pretranslation feature is not used. When the DRAM controller is used with bus masters other than the *WE* 32100 CPU, this input must be pulled high through a pull-up resistor. **Note:** If an MMU is present, this signal must be wired to the corresponding MMU signal.
$\overline{\text{CS}}$	M11	I	**Chip Select.** Selects the DRAM array under the control of the DRAM controller for a data transfer. The DRAM controller may actually initiate the memory access before receiving a chip select (because $\overline{\text{AS}}$, $\overline{\text{PAS}}$, $\overline{\text{EMCI}}$, or $\overline{\text{CYCLEI}}$ were asserted), but if $\overline{\text{CS}}$ is not detected within one half cycle of detection of $\overline{\text{PAS}}$, the DRAM controller terminates the access. If the DRAM controller is programmed for pretranslation and $\overline{\text{CS}}$ is negated before the DRAM controller completes the access by returning $\overline{\text{DTACK}}$, $\overline{\text{SRDY}}$, or $\overline{\text{FAULT}}$ (due to the CPU responding to a bus exception such as $\overline{\text{RRREQ}}$, $\overline{\text{RETRY}}$, or $\overline{\text{FAULT}}$), there must be at least two cycles before a new $\overline{\text{AS}}$ is asserted which would result in a new chip select to the DRAM controller.
$\overline{\text{CYCLEI}}$	L12	I	**Physical Cycle Initiate.** This signal, when asserted, starts the synchronous bus cycle. In the pretranslation mode, it continues a synchronous bus cycle that began with $\overline{\text{AS}}$ assertion. This signal must be tied high if not used.
$\overline{\text{DS}}$	M9	I	**Data Strobe.** During a read operation, this signal indicates that data can be placed on the data bus. During a write operation, this signal, when asserted, indicates that valid data is on the data bus.

Table 14. System Interface (Continued)

Name	Pin(s)	Type	Function
DSIZE0—DSIZE2	E12, D13, F12	I	**Data Size.** These inputs specify the size of the memory transaction. These inputs are used to indicate whether the transaction is byte, halfword, word, double word, or quad-word in the current bus cycle. The following table shows how the DSIZE0—DSIZE2 inputs are interpreted. **DSIZE 2 1 0** — **Size of Transaction** 0 0 0 — Three-byte 0 0 1 — Zero-byte 0 1 0 — Three-word 0 1 1 — Quad-Word 1 0 0 — Word 1 0 1 — Double-Word 1 1 0 — Halfword 1 1 1 — Byte
$\overline{\text{DTACK}}$	M7	O	**Data Transfer Acknowledge.** This is the asynchronous acknowledge to inform the system bus master that valid data has been transferred as requested.
$\overline{\text{EMCI}}$	K10	I	**Early MMU Cycle Initiate.** When asserted, this signal starts the synchronous bus cycle. In pretranslation mode, the signal continues a synchronous bus cycle that began with $\overline{\text{AS}}$ assertion. This signal must be tied high if not used.
$\overline{\text{ENABLEB}}$	L1	O	**Enable Port B.** In a dual-port environment; the master DRAM controller enables the slave DRAM controller with this signal after receiving $\overline{\text{REQB}}$. The signal is asserted by the master after completing any operations in progress and 3-stating its outputs to the DRAM array.
$\overline{\text{FAULT}}$	N7	O	**Fault.** When asserted, this signal indicates that an uncorrectable memory error was detected. An uncorrectable memory error detected during an error scrub mode refresh will not cause $\overline{\text{FAULT}}$ to be asserted.
$\overline{\text{LOCKA}}$	G1	I	**Lock Request A.** This signal has meaning only in a dual-port environment. If this signal is asserted before the master DRAM controller's master bus request access terminates, the master arbiter will delay service for requests to shared memory from the slave DRAM controller. Service will be denied until this signal is negated. This signal must remain asserted until the interlocked operation has been completed. In nondual-port configurations, this pin must be tied high.
$\overline{\text{LOCKB}}$	J2	I	**Lock Request B.** This signal has meaning only in a dual-port environment. If this signal is asserted before a port B access terminates, the arbiter in the master DRAM controller will disallow nonrefresh accesses into the memory from port A until $\overline{\text{LOCKB}}$ is negated. In nondual-port configurations, this pin must be tied high.

Table 14. System Interface (Continued)

Name	Pin(s)	Type	Function
$\overline{PAS}$	N12	I	**Physical Address Strobe.** This signal indicates that the physical address is valid. In a dual-ported configuration, this signal on the slave must be connected to $\overline{ENABLEB}$ on the master.
$\overline{PMCS}$	K1	I	**Peripheral Mode Chip Select.** This signal selects the DRAM controller for a peripheral mode access.
$\overline{REFRQ}$	L2	I	**Refresh Request.** When asserted, this signal tells the DRAM controller to perform a refresh. It will always be honored, regardless of the setting of the AUTORF bit in the configuration registers. Assertion of this signal also resets the internal refresh timer. This signal must be tied high if not used.
$\overline{REQB}$	H1	I	**Request Port B.** This signal indicates to the master DRAM controller that port B is requesting a memory access. This signal must be tied high on the DRAM controller in a single-port environment and on the slave DRAM controller of a dual-port environment.
R/$\overline{W}$	N8	I	**Read-Write.** This signal indicates whether the transaction is a read or a write.
$\overline{SRDY}$	N6	O	**Synchronous Data Ready.** This is the synchronous acknowledge to inform the system bus master that valid data has been transferred as requested. The signal is always negated by the DRAM controller one cycle after being asserted.
$\overline{WZ}$	M8	O	**Write Zeros.** This signal is used only in an EDC configuration and tells the EDC to drive zeros onto the DRAM data bus. Corresponding check bits are also generated. It is asserted only during memory initialization.

Table 15. DRAM Interface Signals

Name	Pin(s)	Type	Function
$\overline{BM0}$—$\overline{BM3}$	M2, M1, M3, N1	O	**Byte Mark.** These outputs select the byte(s) written to memory during partial and full-word writes. During reads without EDC all are asserted independently of the data size. During reads with EDC, none are asserted.
$\overline{CAS0}$—$\overline{CAS3}$	K12, J13, J12, H13	O	**Column Address Strobe.** These outputs are used by the DRAM array to latch the column address present on the address bus. In a system with no EDC, they must be qualified by the byte marks to select which byte(s) of a word is accessed.

Table 15. DRAM Interface Signals (Continued)

Name	Pin(s)	Type	Function
$\overline{EDCWE}$	M4	O	**EDC Write Enable.** This signal is the read/write control for the EDC. When asserted, the EDC generates check bits on its data. For configurations with by-four DRAMs, its assertion causes the EDC to latch the DRAM data for use during a read-modify-write sequence.
ERRCR	M6	I	**Error Corrected.** This signal informs the DRAM controller that the error detected is correctable and the corrected data is now valid. This input must be tied low if there is no EDC in the system.
$\overline{ERRDT}$	N5	I	**Error Detected.** This signal informs the DRAM controller that an error has been detected in the data being read from memory. This input must be tied high if error checking is not provided by the system.
$\overline{G}$	N2	O	**DRAM Output Enable.** This signal drives the $\overline{G}$ inputs of memory devices that have multiplexed data in and data out (e.g., "by-four memories"). Assertion of $\overline{G}$ allows the DRAMs to drive the data bus.
$\overline{IE}$	N3	O	**Input Enable.** This signal enables the data input buffers to drive data from the system data bus to the memory data bus. Systems that incorporate EDC should qualify this signal with byte marks.
$\overline{OE}$	N9	O	**Output enable.** This signal enables the data output buffers to drive data from the memory data bus onto the system data bus.
$\overline{RAS0}$—$\overline{RAS3}$	H12, G13, G12, F13	O	**Row Address Strobe.** These outputs are used by the DRAM array to latch the row address present on the address bus.
$\overline{WE}$	M5	O	**Write Enable.** This signal drives the $\overline{WE}$ inputs on the DRAM array. It enables them for a write if $\overline{CASX}$ is not yet asserted and forces a late write if $\overline{CASX}$ is already asserted.

Table 16. Clocks

Name	Pin(s)	Type	Function
CLK23	L8	I	**Clock.** This clock input leads CLK34 by 90°.
CLK34	L7	I	**Clock.** The falling edge of this input signifies the beginning of phase one.

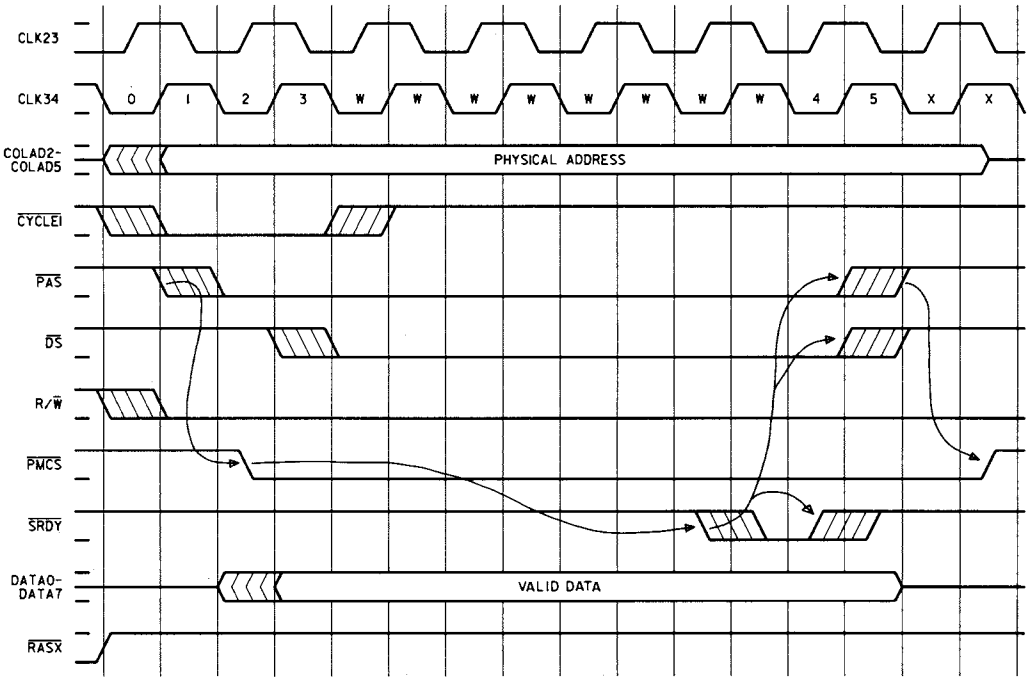
Table 17. Miscellaneous Signals

Name	Pin(s)	Type	Function
$\overline{\text{HIGHZ}}$	E13	I	**High Impedance.** When asserted, this signal puts all DRAM controller outputs into the high impedance state. This signal is provided for testing purposes. This signal must be pulled high through a pull-up resistor.
$\overline{\text{RESET}}$	N4	I	**Reset.** This signal resets the DRAM controller. It must be asserted for at least three cycles to guarantee that it is recognized by the DRAM controller. This signal must be pulled high through a pull-up resistor.

Protocol Diagrams

Figures 13 through 30 show the behavior of the DRAM controller for several bus transactions with arrows to show the sequences of "cause and effect."

Figures 13 and 14 detail a peripheral mode access. Synchronous single- and double-word transactions with and without an MMU and with and without an early cycle initiate are shown on Figures 15 through 21 and on Figures 27 through 30. Figures 22 through 24 detail the DRAM controller's behavior while configures with an EDC. Asynchronous transactions are illustrated on Figures 25 and 26.



Note: No $\overline{\text{RASX}}$ assertion before $\overline{\text{CS}}$ assertion when start bit is inactive.

Figure 13. Synchronous Peripheral Mode Write (No MMU, No EDC, Start Bit Not Yet Set)

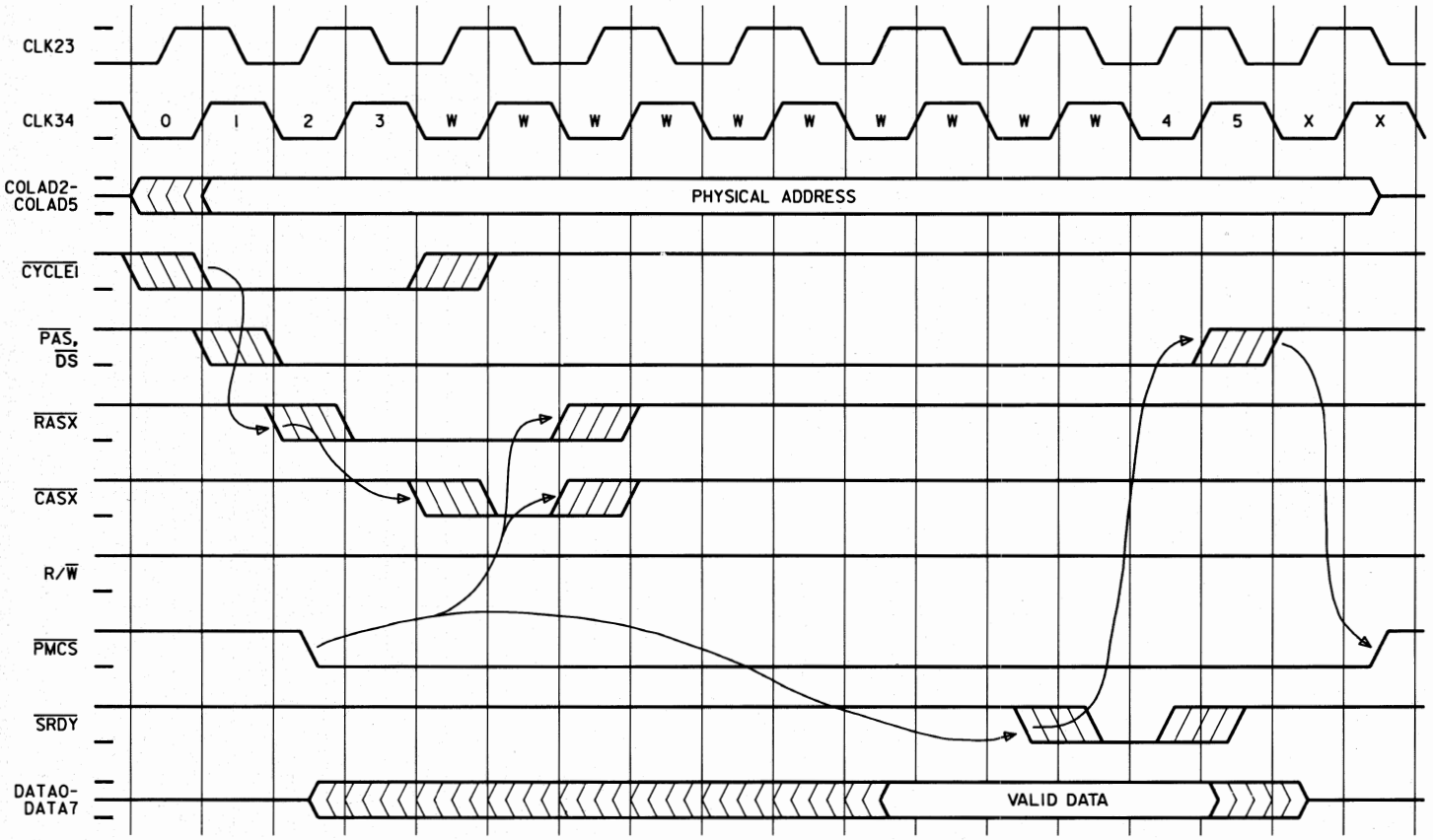
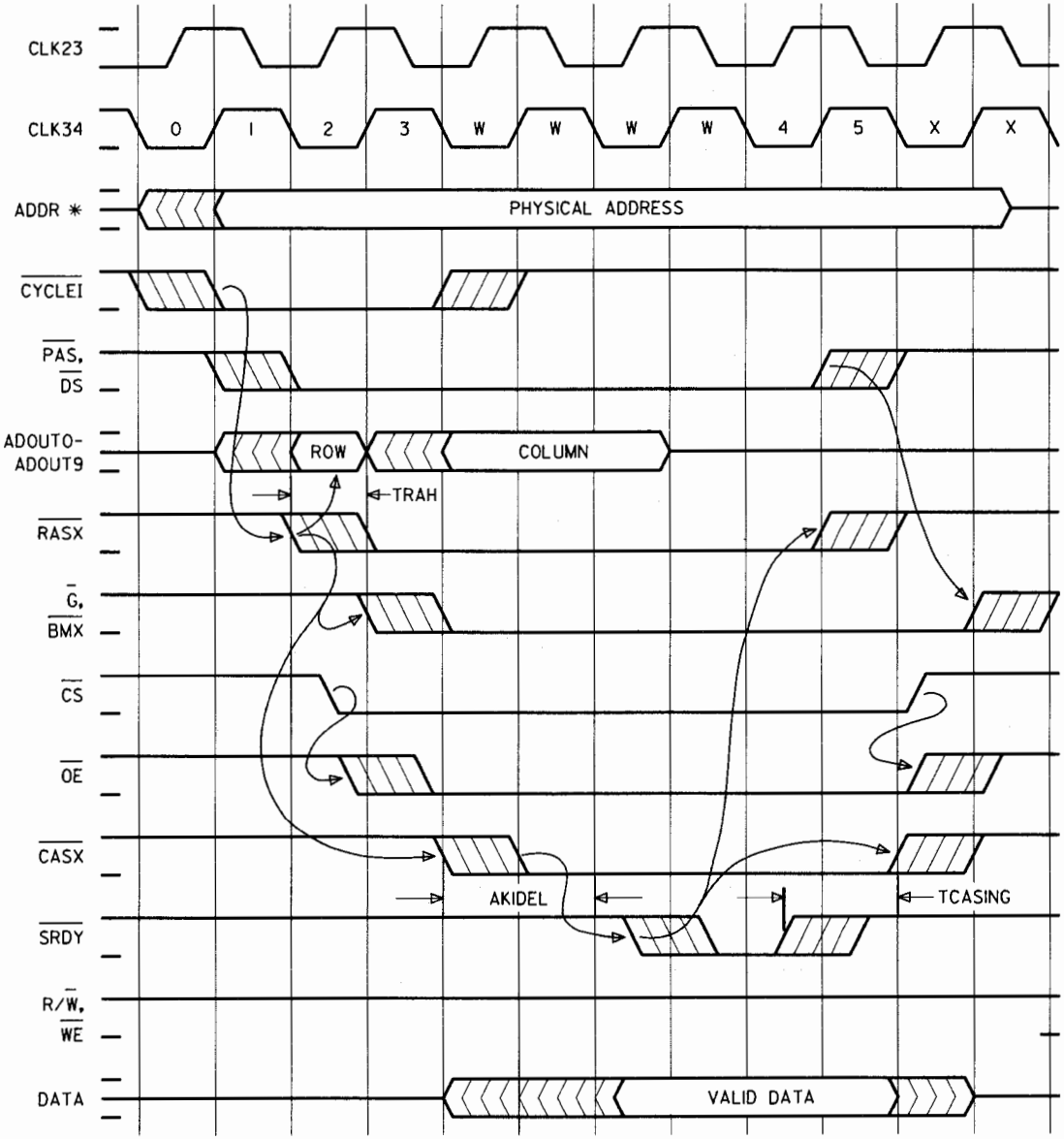
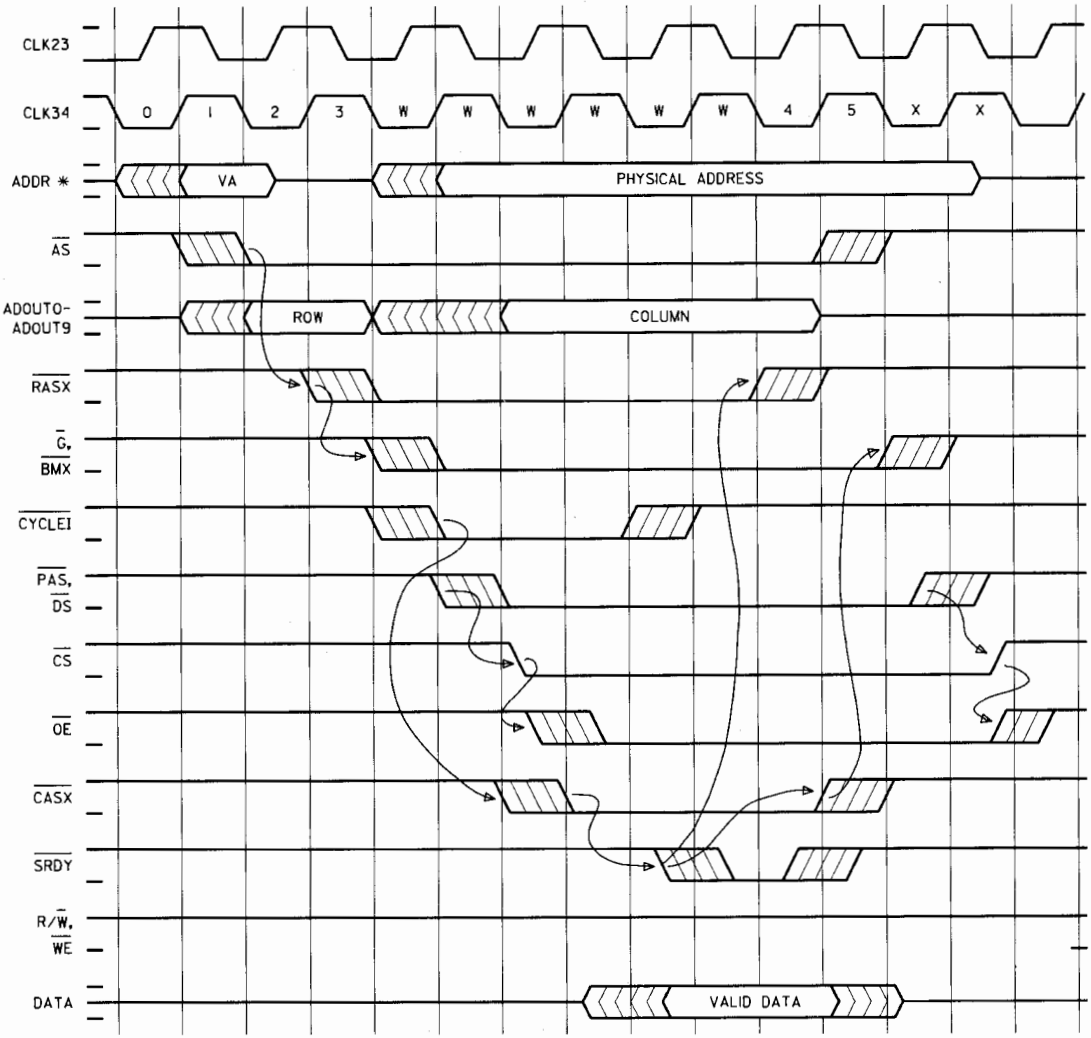


Figure 14. Synchronous Peripheral Mode Read Access (No MMU, No EDC, Start Bit Set)



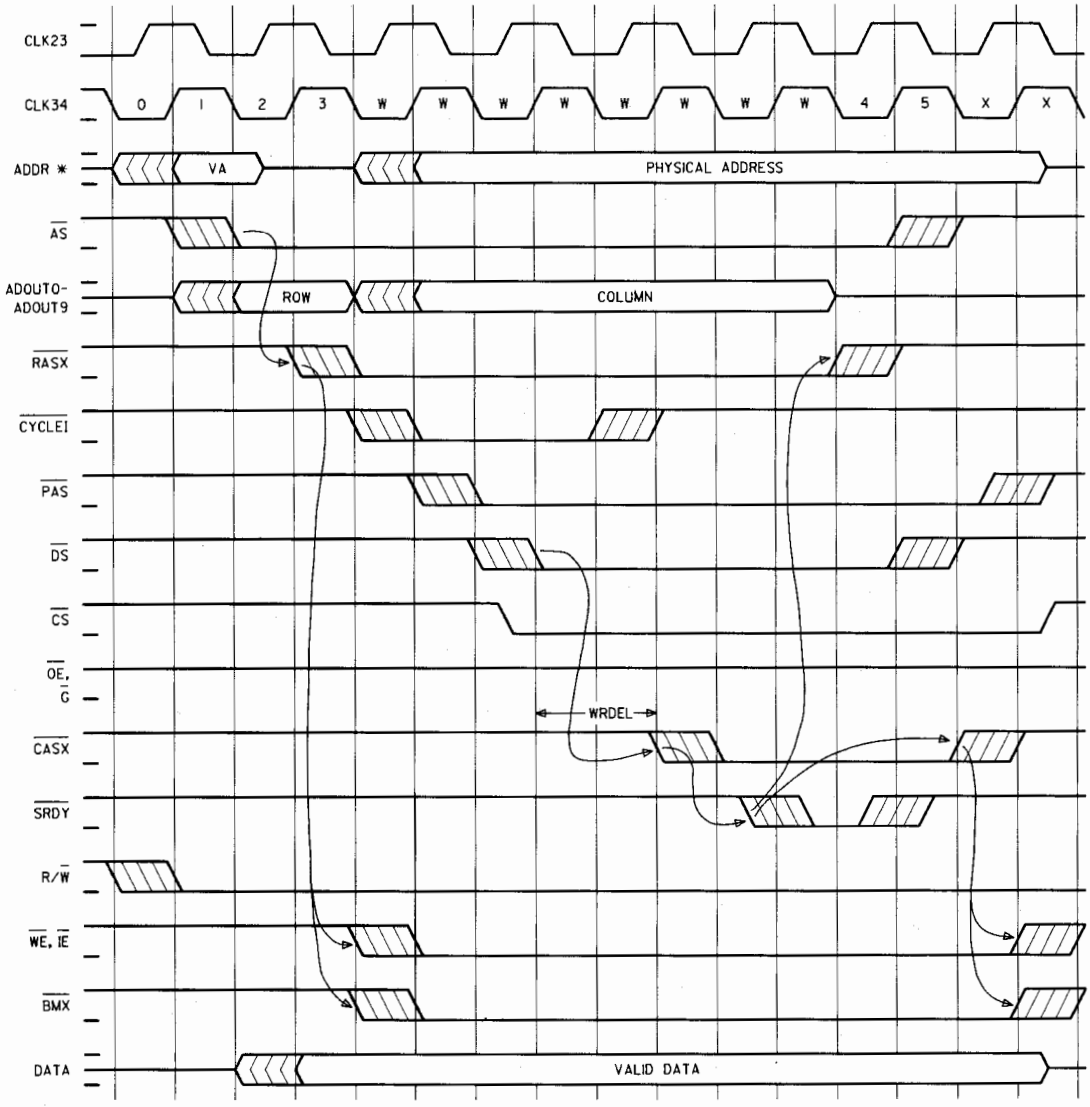
* ADDR is made up of BYTEAD0, BYTEAD1, ROWAD0—ROWAD9, COLAD0—COLAD9, BANKAD0, BANKAD1.

Figure 15. Single-Word Read (No MMU, Synchronous, No EDC or EDC Without Error Detection)



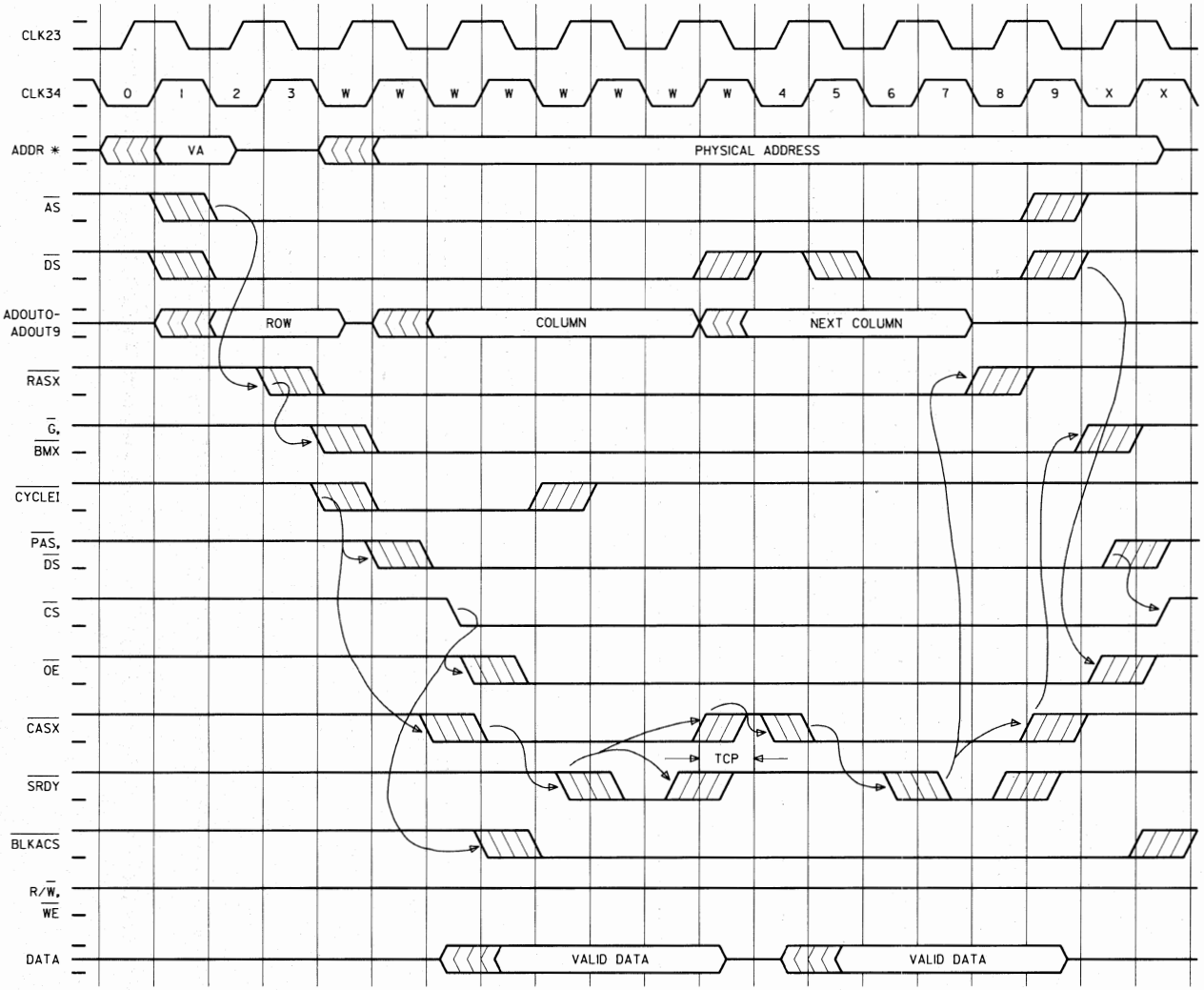
* ADDR is made up of BYTEAD0, BYTEAD1, ROWAD0—ROWAD9, COLAD0—COLAD9, BANKAD0, BANKAD1.

Figure 16. Single-Word Read (MMU Present, Synchronous, Pretranslation, No EDC, No Early Cycle Initiate)



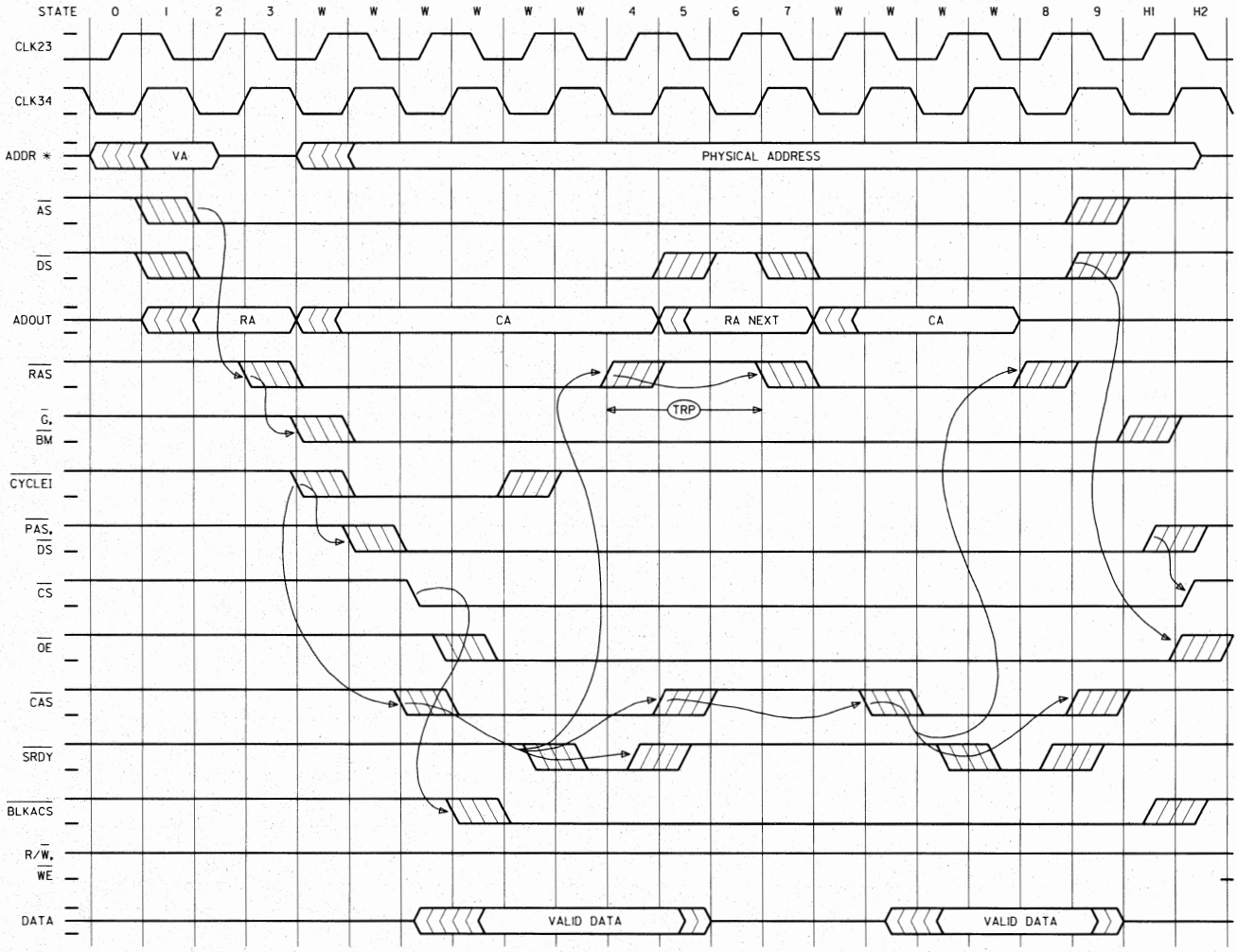
* ADDR is made up of BYTEAD0, BYTEAD1, ROWAD0—ROWAD9, COLAD0—COLAD9, BANKAD0, BANKAD1.

Figure 17. Single-Word Write (MMU, Synchronous, Pretranslation, No EDC, No Early Cycle Initiate)



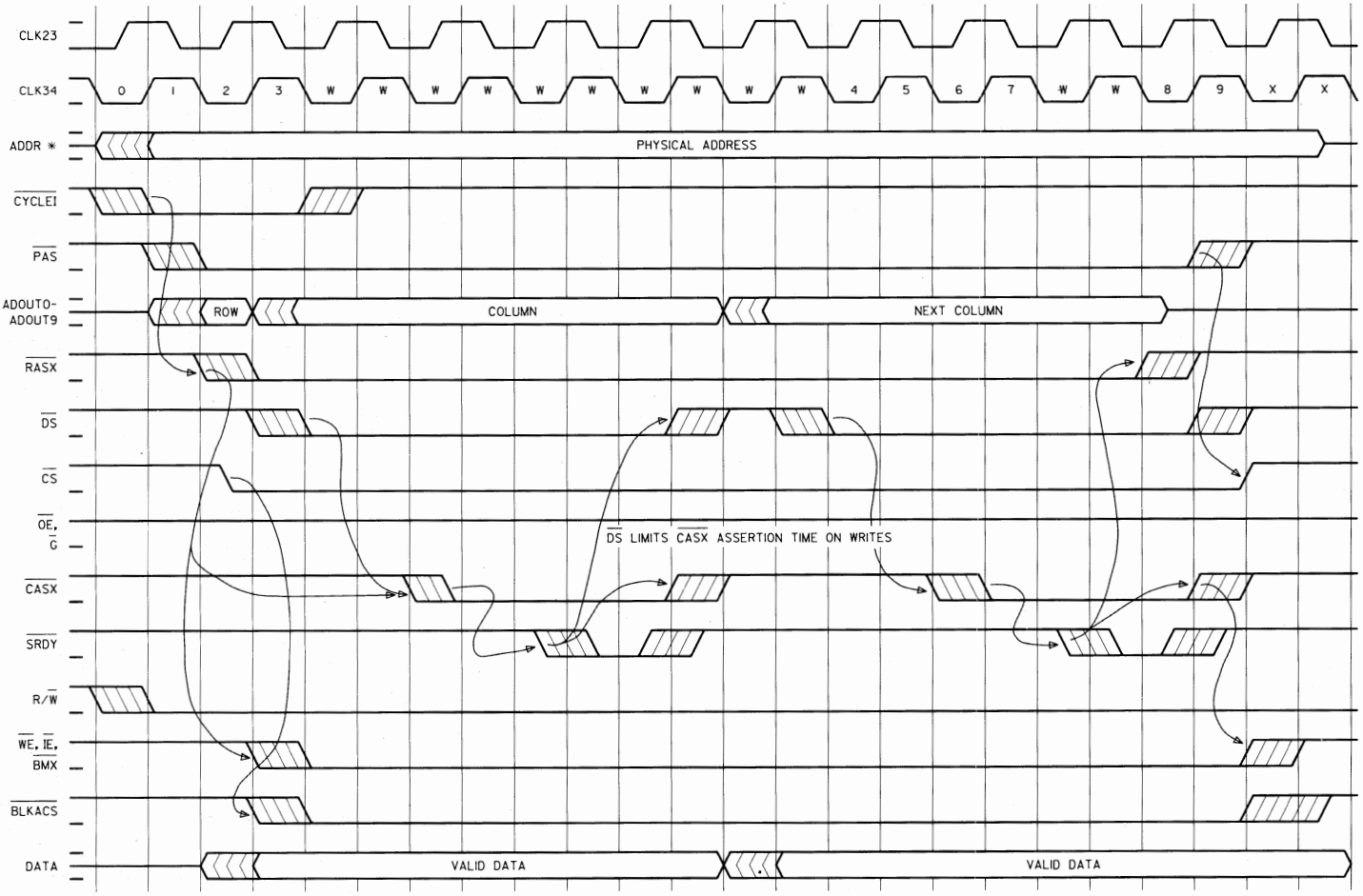
* ADDR is made up of BYTEAD0, BYTEAD1, ROWAD0—ROWAD9, COLAD0—COLAD9, BANKAD0, BANKAD1.

Figure 18. Double-Word Read (MMU, Synchronous, Pretranslation, Page Mode Access on Second Word, No Early Cycle Initiate)



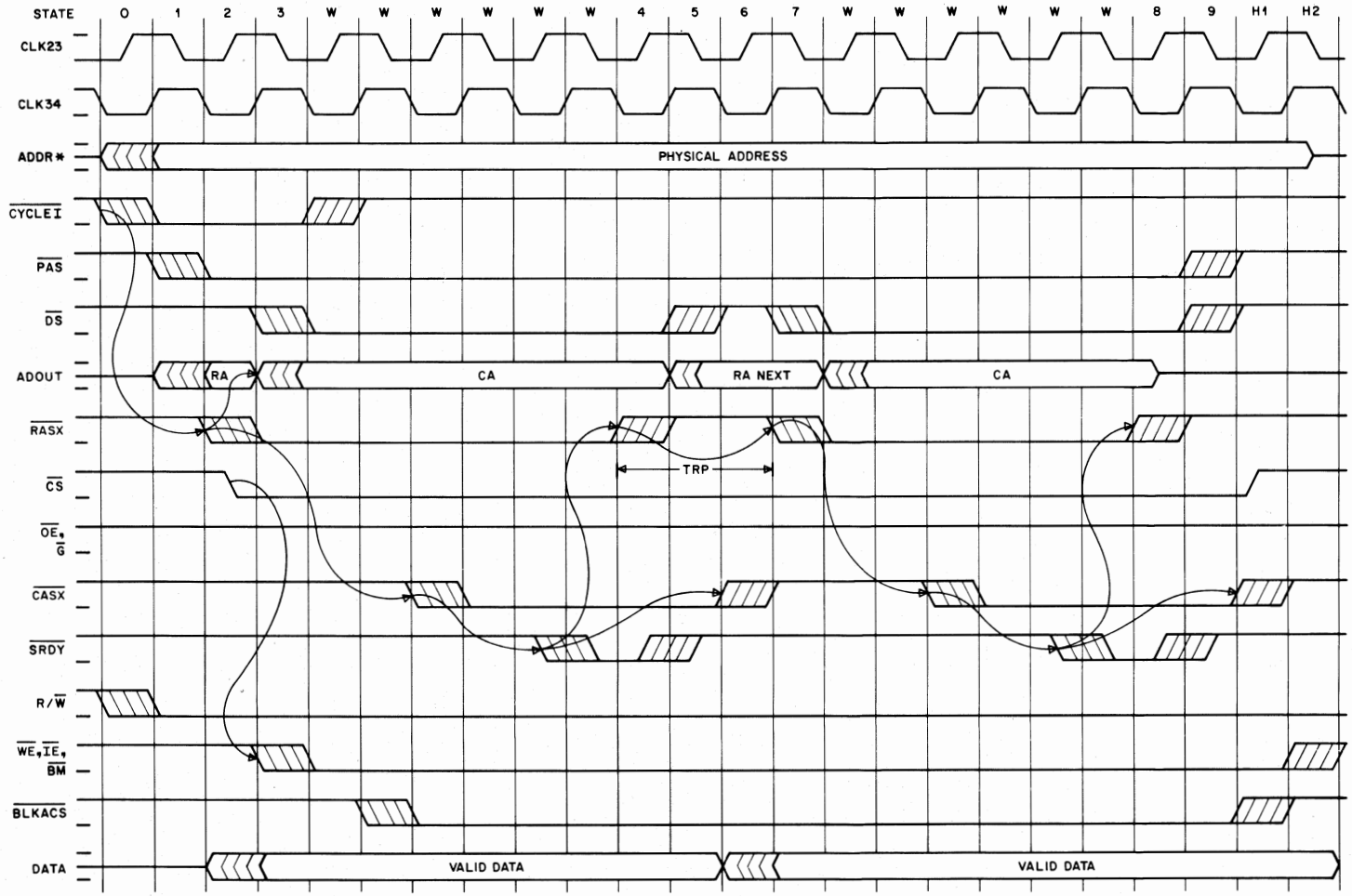
* ADDR is made up of BYTEAD0, BYTEAD1, ROWAD0—ROWAD9, COLAD0—COLAD9, BANKAD0, BANKAD1.

Figure 19. Double-Word Read (MMU, Synchronous, Pretranslation, Cycle RAS Instead of Page-Nibble Mode of DRAM, No EDC, No Early Cycle Initiate)



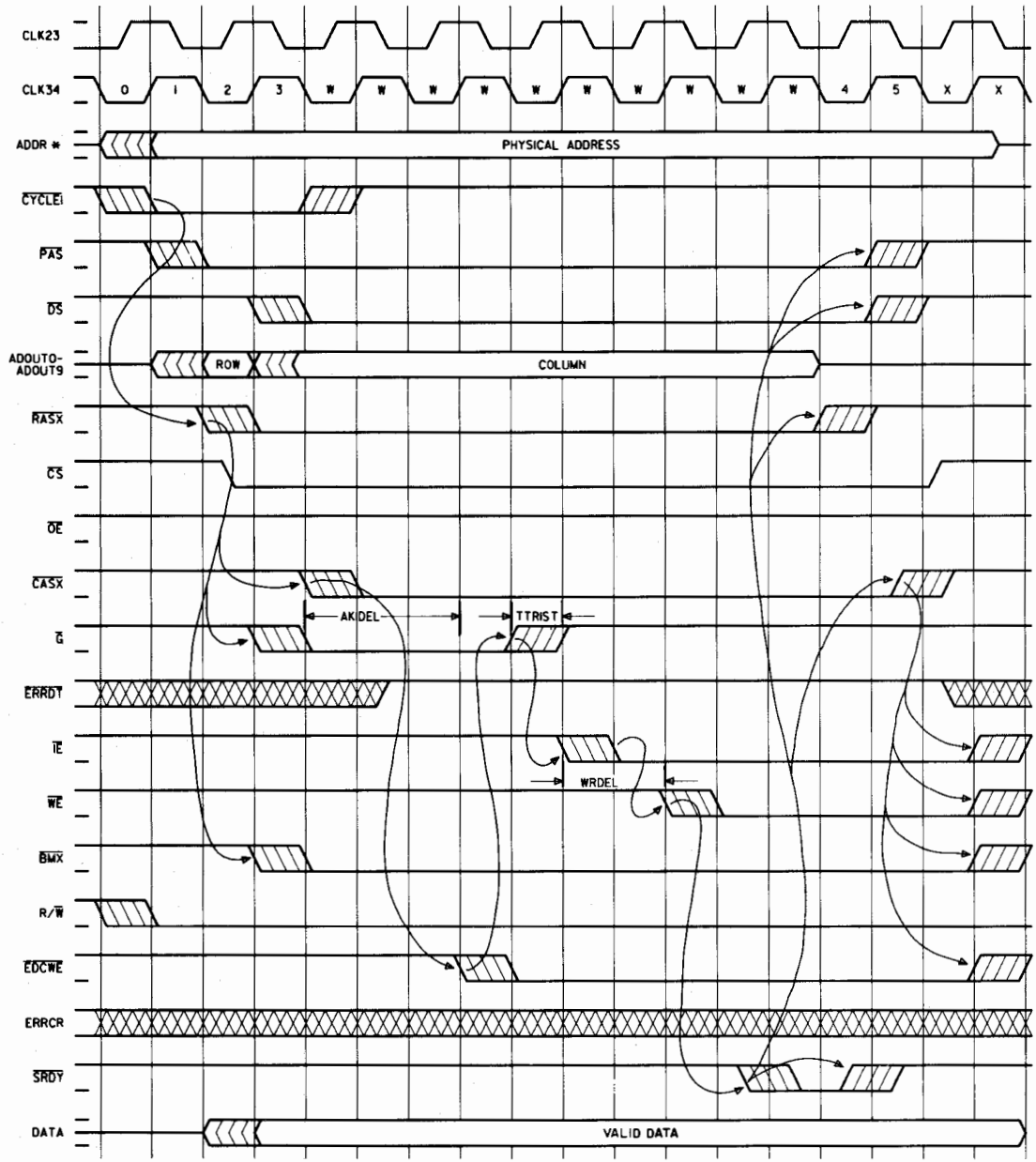
* ADDR is made up of BYTEAD0, BYTEAD1, ROWAD0—ROWAD9, COLAD0—COLAD9, BANKAD0, BANKAD1.

Figure 20. Double-Word Write (No MMU, Synchronous, Page Mode Access on Second Word)



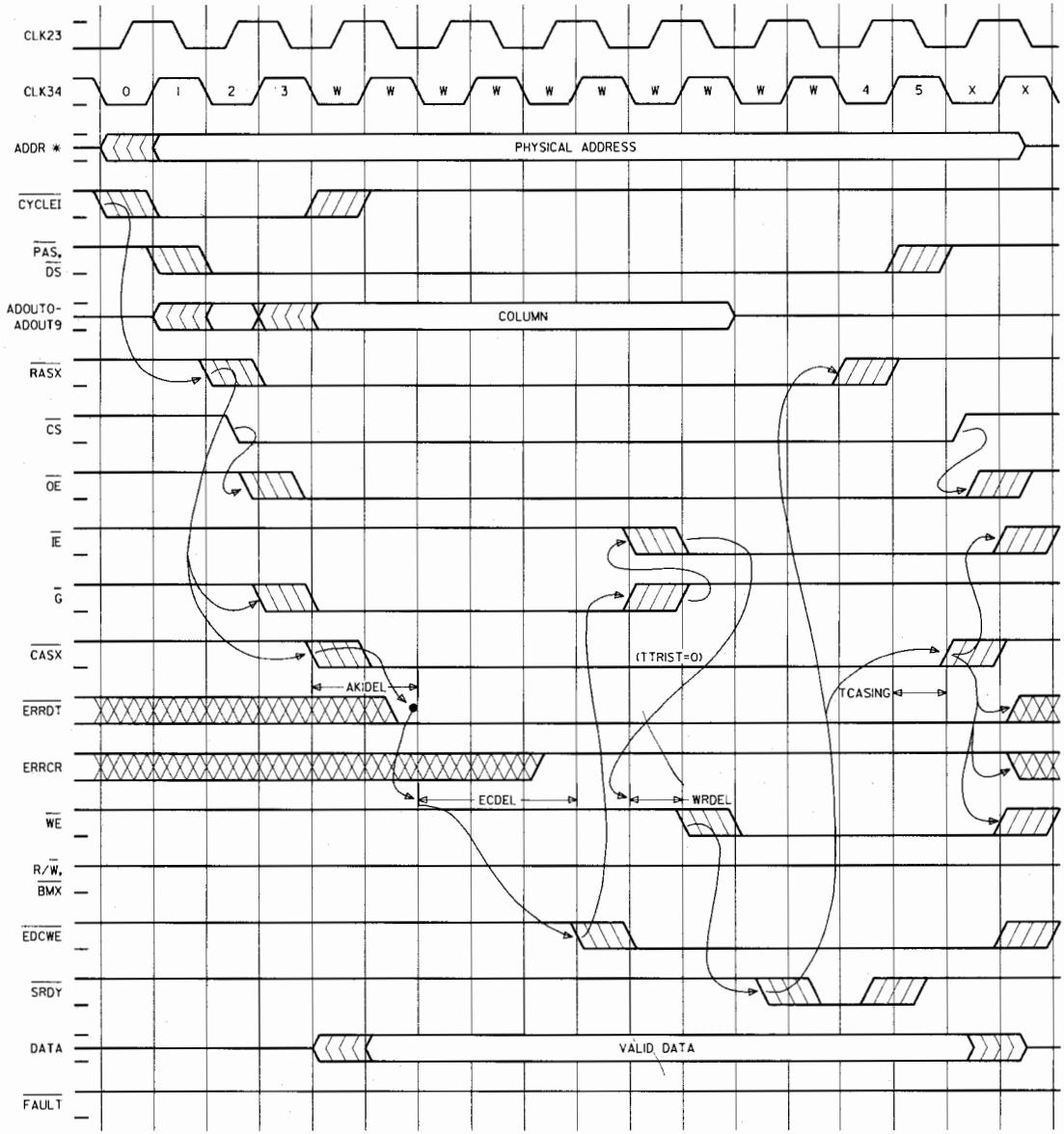
* ADDR is made up of BYTEAD0, BYTEAD1, ROWAD0—ROWAD9, COLAD0—COLAD9, BANKAD0, BANKAD1.

Figure 21. Double-Word Write (No MMU, CYCLE RAS Instead of Page-Nibble Mode of DRAM, No EDC)



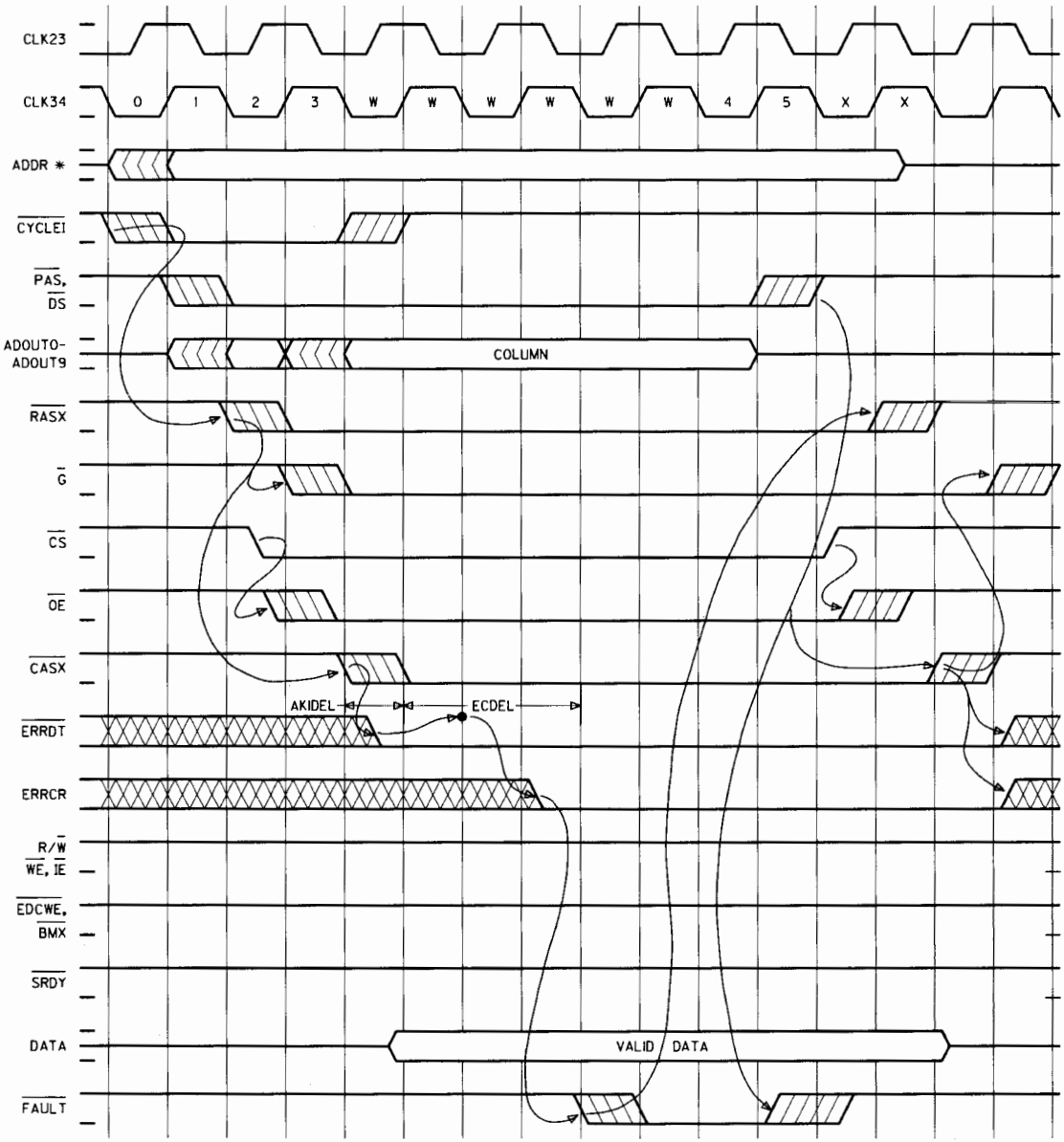
* ADDR is made up of BYTEAD0, BYTEAD1, ROWAD0—ROWAD9, COLAD0—COLAD9, BANKAD0, BANKAD1.

Figure 22. Partial-Word Write (No MMU, EDC, Synchronous, No Error Detected During Read)



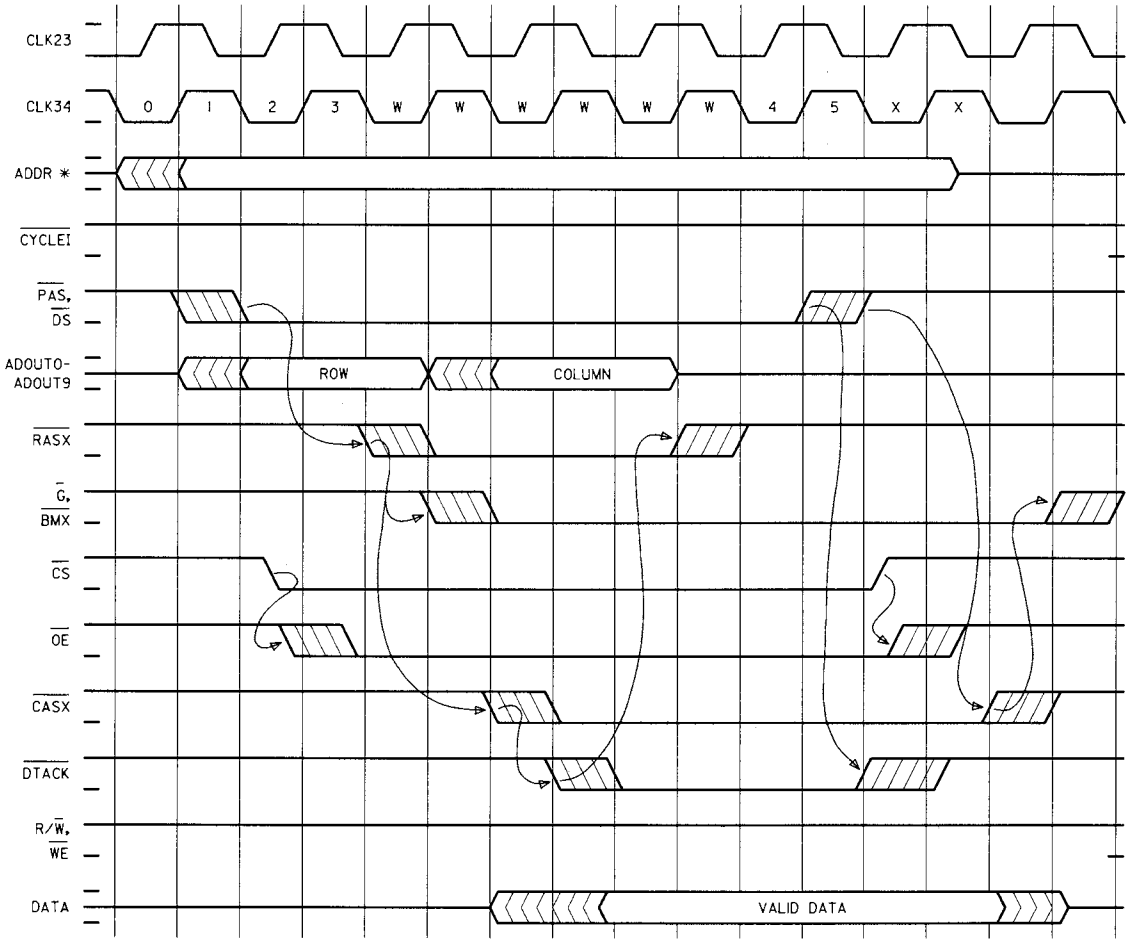
* ADDR is made up of BYTEAD0, BYTEAD1, ROWAD0—ROWAD9, COLAD0—COLAD9, BANKAD0, BANKAD1.

Figure 23. Single-Word Read (No MMU, Synchronous, Correctable Error)



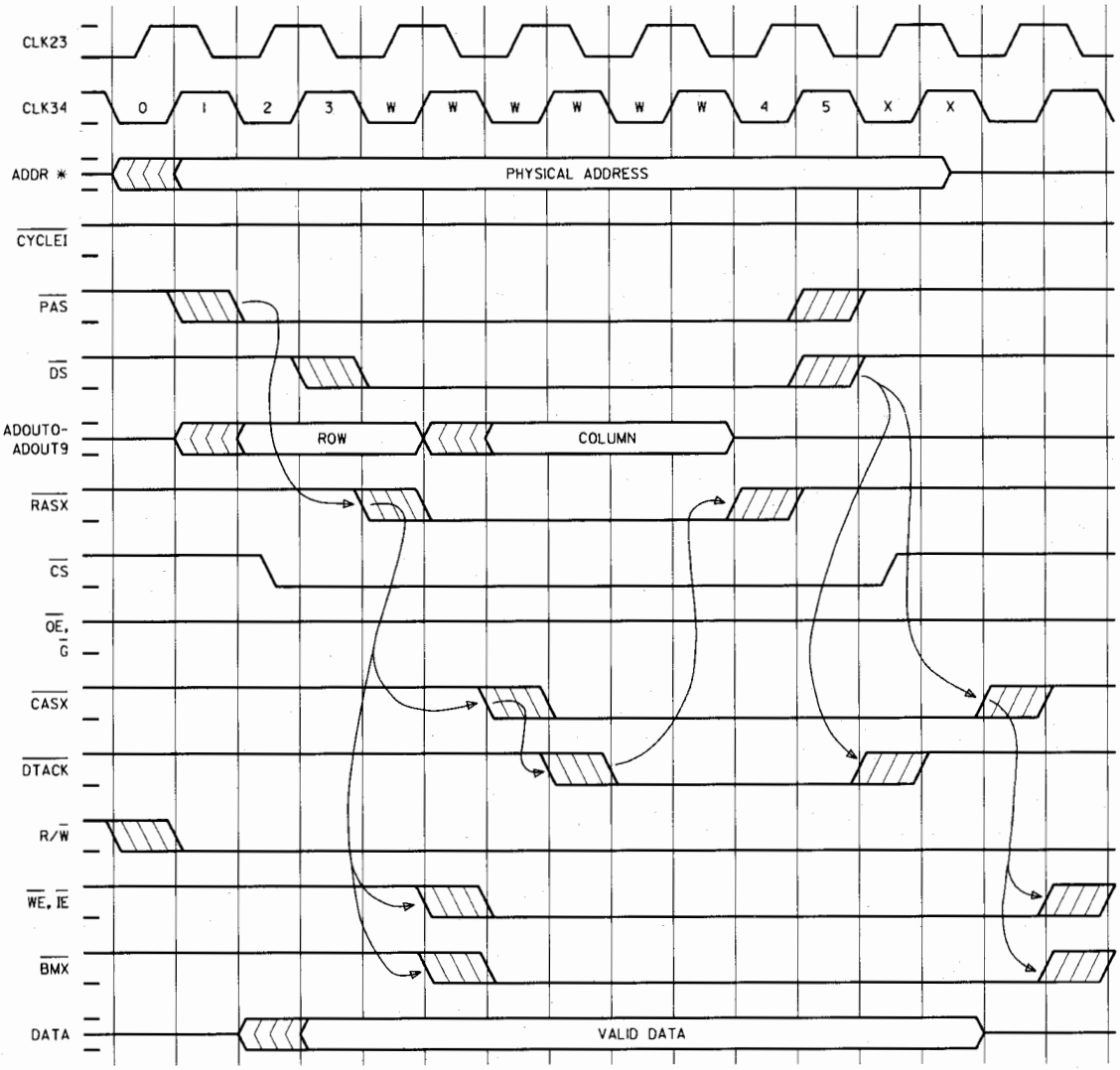
* ADDR is made up of BYTEAD0, BYTEAD1, ROWAD0—ROWAD9, COLAD0—COLAD9, BANKAD0, BANKAD1.

Figure 24. Single-Word Read (No MMU, Synchronous, EDC, Uncorrectable Error Found)



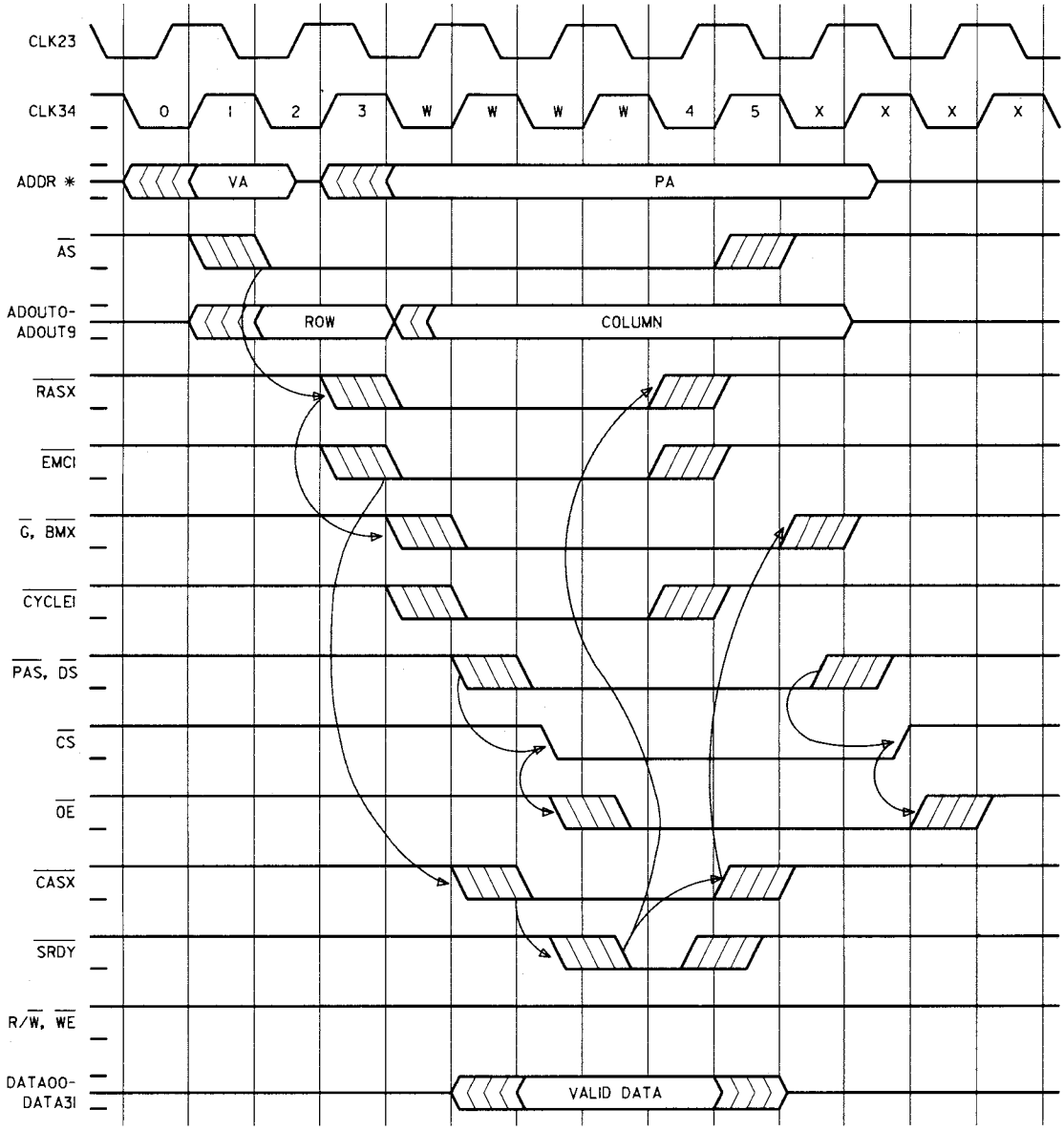
* ADDR is made up of BYTEAD0, BYTEAD1, ROWAD0—ROWAD9, COLAD0—COLAD9, BANKAD0, BANKAD1.

Figure 25. Single-Word Read (No MMU, Asynchronous)



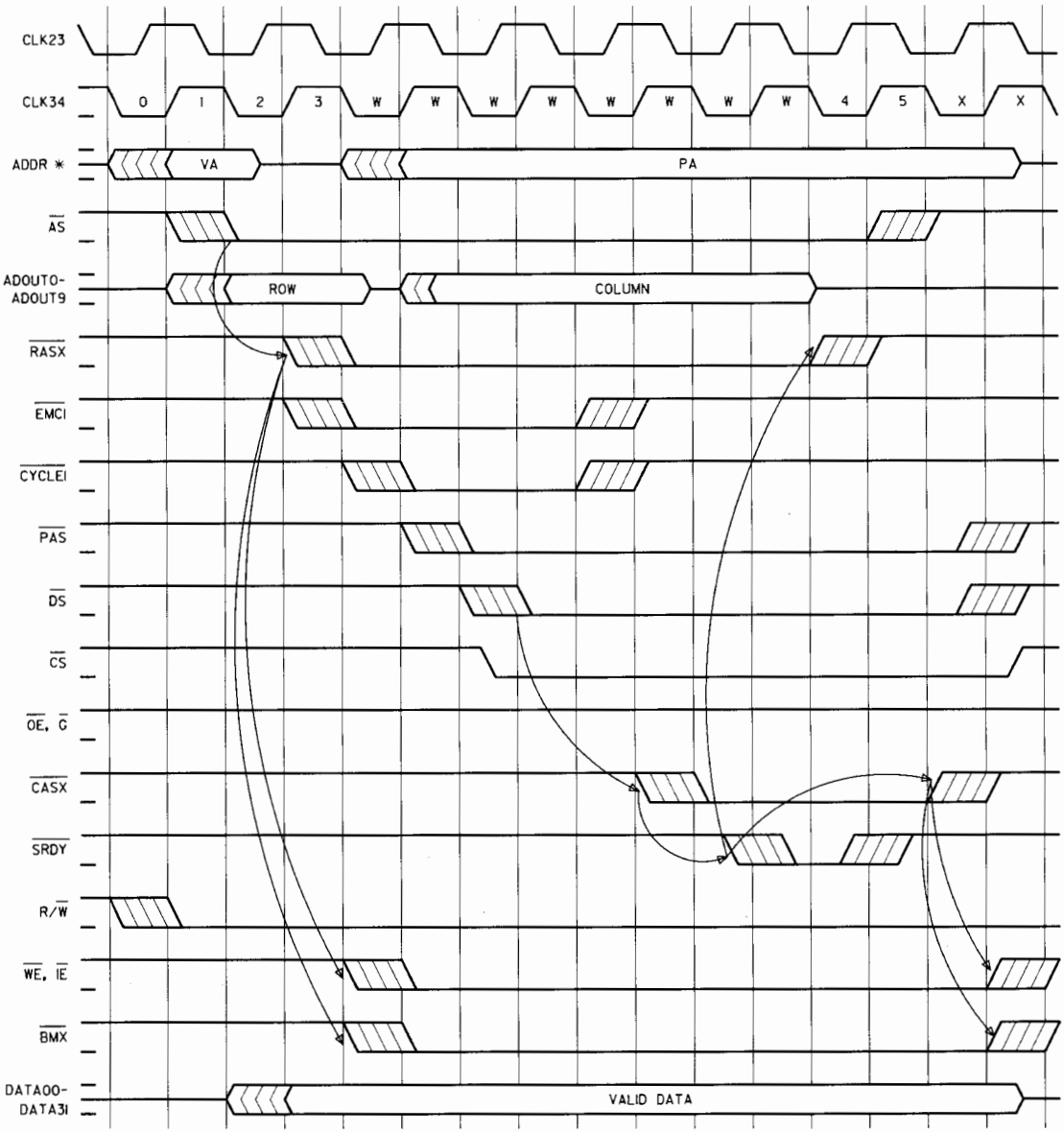
* ADDR is made up of BYTEAD0, BYTEAD1, ROWAD0—ROWAD9, COLAD0—COLAD9, BANKAD0, BANKAD1.

Figure 26. Single-Word Write (No MMU, Asynchronous, No EDC)



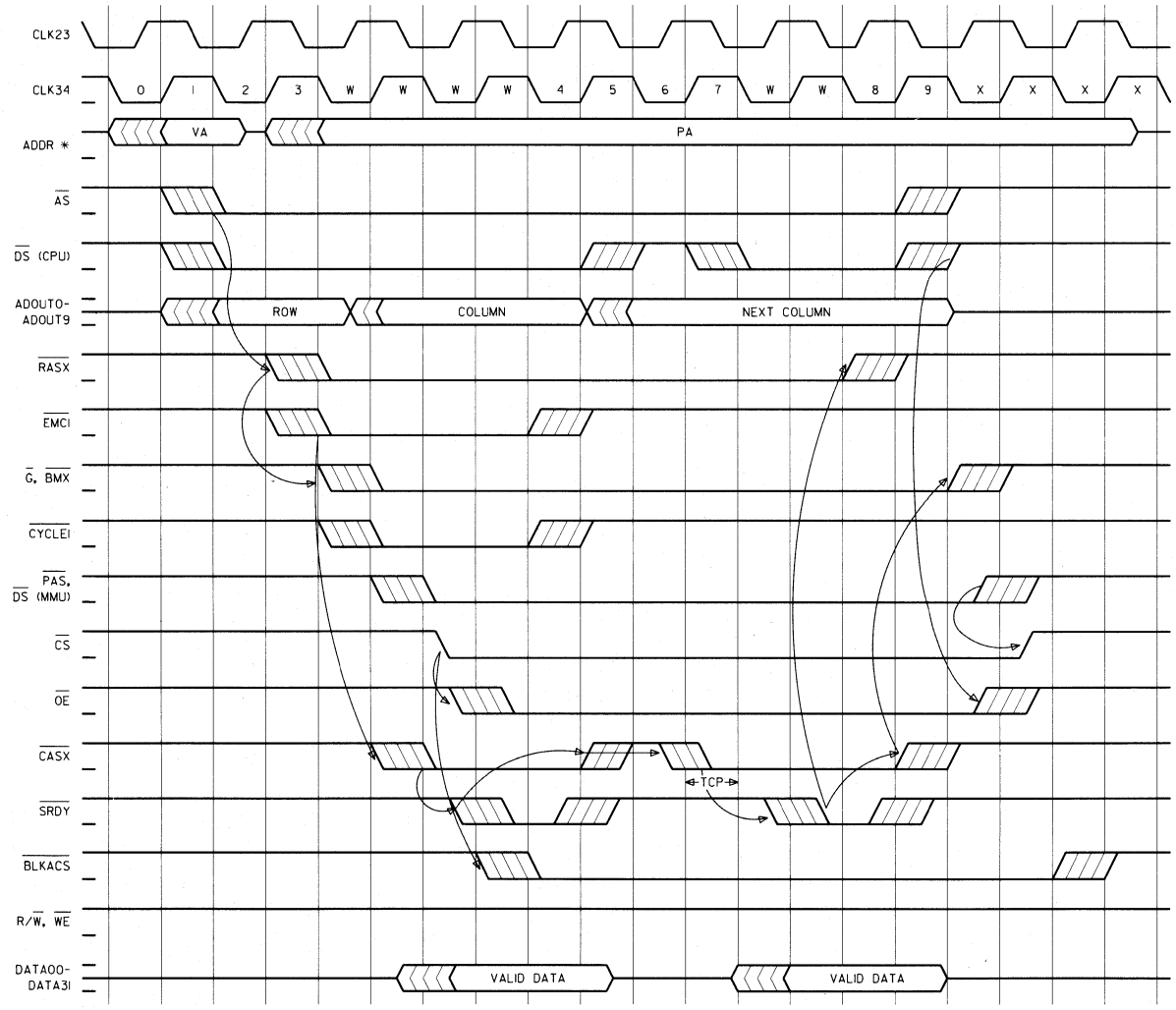
* ADDR is made up of BYTEAD0, BYTEAD1, ROWAD0—ROWAD9, COLAD0—COLAD9, BANKAD0, BANKAD1.

Figure 27. Single-Word Read (MMU, Synchronous, Pretranslation, No EDC, Early Cycle Initiate)



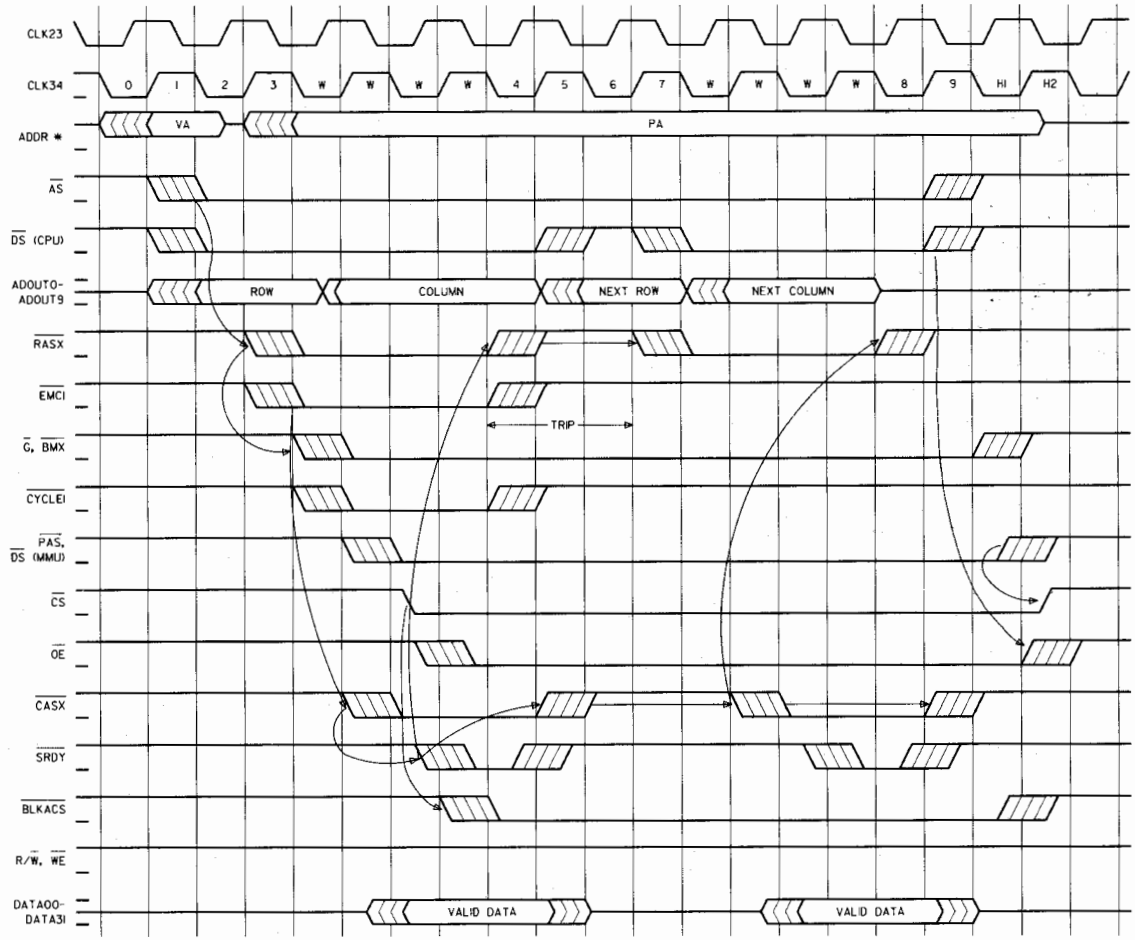
* ADDR is made up of BYTEAD0, BYTEAD1, ROWAD0—ROWAD9, COLAD0—COLAD9, BANKAD0, BANKAD1.

Figure 28. Single-Word Write (MMU, Synchronous, Pretranslation, No EDC, Early Cycle Initiate)



* ADDR is made up of BYTEAD0, BYTEAD1, ROWAD0—ROWAD9, COLAD0—COLAD9, BANKAD0, BANKAD1.

Figure 29. Double-Word Read (MMU, Synchronous, Pretranslation, Page Mode Access on Second Word, Early Cycle Initiate)



* ADDR is made up of BYTEAD0, BYTEAD1, ROWAD0—ROWAD9, COLAD0—COLAD9, BANKAD0, BANKAD1.

Figure 30. Double-Word Read (MMU, Cycle RAS Instead of PAGNIB Mode, No EDC, Early Cycle Initiate)

Characteristics

Timing Characteristics

All TTL timing specifications are referenced to and from 0.8 V for a low voltage and 2.0 V for a high voltage. CMOS clock references are to and from Vcc/2. All min and max values are in ns. Notes, which may appear in any column, are indicated by numbers or single letters in parentheses and appear at the end of the table.

Table 18. Timing Characteristics

Num	Symbol	Description	10 MHz		14 MHz		18 MHz	
			Min	Max	Min	Max	Min	Max
1	tBKDVC2XH	Bank address set-up time (2, 3)	35	—	25	—	19	—
2	tC2XHBKDX	Bank address hold time (3)	50	—	36	—	28	—
3	tDSZVC2XH	Data size set-up time (4)	0	—	0	—	0	—
4	tC2XHDSZX	Data size hold time (5)	45	—	32	—	25	—
5	tRWDVC2XH	Row address set-up time (5)	35	—	25	—	19	—
6	tC2XHRWDX	Row address hold time (3)	50	—	36	—	28	—
7A	tCLDVC2XH	Column address set-up time (3)	35	—	25	—	19	—
7B	tCLDVC2XL	Column address set-up time (13)	15	—	11	—	8	—
8A	tPMCLCLD	Column address hold time	150	—	107	—	84	—
8B	tC2XLCLDX	Column address hold time (13)	135	—	97	—	76	—
9A	tBTDVC2XH	Byte address set-up time (5)	15	—	11	—	8	—
9B	tBTDVC2XL	Byte address set-up time (4)	35	—	25	—	19	—
9C	tBTDVASBL	Byte address set-up time	10	—	7	—	6	—
10A	tC2XLBTDX	Byte address hold time (4)	45	—	32	—	25	—
10B	tASBLBTDX	Byte address hold time	40	—	29	—	22	—
11A	tCYCLC23L	Cycle initiate set-up time	12	—	9	—	7	—
11B	tCYCLCYCH	Cycle initiate hold time	200	—	143	—	112	—
12A	tECYLC23H	Early cycle initiate set-up time	12	—	9	—	7	—
12B	tECYLECYH	Early cycle initiate hold time	250	—	179	—	139	—
13	tASBLC23H	Address strobe set-up time (4)	15	—	11	—	8	—
14A	tPASVC2XL	Physical address strobe set-up time (1)	15	—	11	—	8	—
15A	tDSBLC2XL	Data strobe set-up time	15	—	11	—	8	—
15B	tPMCHDSBH	Data strobe negation set-up time	0	—	0	—	0	—
16	tPASLCSOV	Chip select delay time	—	40	—	29	—	22
17	tCSOLC2XL	Chip select set-up time (1)	15	—	11	—	8	—
18	tPASHCSOH	Chip select hold time	0	—	0	—	0	—
19	tPMCLC2XL	Peripheral mode chip select set-up time (1)	15	—	11	—	8	—
21	tRWVC2XH	Read/write set-up time (4)	0	—	0	—	0	—
22	tRWHC2XH	Read/write hold time (5)	45	—	32	—	25	—
25	tCNTLC23L	Continuous segment set-up time	13	—	9	—	7	—
29	tABTLC2XL	Abort set-up time (1)	15	—	11	—	8	—

Num	Symbol	Description	10 MHz		14 MHz		18 MHz	
			Min	Max	Min	Max	Min	Max
30	tRASLABTH	Abort width	90	—	65	—	50	—
31A	tERTLC2XH	Asynchronous error detected set-up time (9)	40	—	29	—	22	—
31B	tEDTHC23H	Synchronous error detected set-up time (10)	20	—	14	—	11	—
32	tCASHEDTX	Error detected hold time (14)	0	—	0	—	0	—
33	tECRHC2XH	Error corrected set-up time (11)	40	—	29	—	22	—
34	tCASHECRX	Error corrected hold time	0	—	0	—	0	—
35	tRSTLC2XL	Reset set-up time (1)	15	—	11	—	8	—
36	tRSTLRSTL	Reset valid	300	—	212	—	167	—
37	tLKALCSOH	Lock request A set-up time (12)	50	—	36	—	28	—
38	tCSOLLKAH	Lock request A hold time (12)	150	—	107	—	84	—
39	tLKBLRQBH	Lock request B set-up time (12)	50	—	36	—	28	—
40	tEMBLLKBX	Lock request B hold time	50	—	36	—	28	—
41	tRFQLC2XL	Refresh request set-up time (1)	15	—	11	—	8	—
42	tRFQLRFQH	Refresh request hold time	100	—	72	—	56	—
43	tDATVDSBL	Data bus set-up time	0	—	0	—	0	—
44	tDSBHDATX	Data bus hold time	0	—	0	—	0	—
45	tBYQLC2XL	Bypass request set-up time (1)	15	—	11	—	8	—
46	tBYKLBYQH	Bypass request hold time	0	—	0	—	0	—
47	tRQBLC2XL	Request port B set-up time (1)	15	—	11	—	8	—
48	tENBLRQBH	Request port B hold time	0	—	0	—	0	—
69	tRWDVADOV	Row address to address out propagation time	—	63	—	45	—	35
70	tADOVRASL	Address out set-up time (ROW) (A)	17	—	17	—	17	—
71A	tADOVCASL	Address out set-up time (COLUMN, TASC=0) (B)	17	—	17	—	17	—
71B	tADOVCASL	Address out set-up time (COLUMN, TASC=1)	67	—	53	—	45	—
71C	tC2XHCA1V	First column address mux propagation time	—	50	—	36	—	28
71D	tC2XHCASV	Successive column address mux propagation time	—	50	—	36	—	32
72A	tRASLADOV	Address out hold time (ROW)	(C) –3	—	(C) –2	—	(C) –2	—
72B	tCASLADOX	Address out hold time (COLUMN)	(G) –3	—	(G) –2	—	(G) –2	—
72C	tCASLADOX	Address out hold time	(H) –3	—	(H) –2	—	(H) –2	—
73A	tC34HRASL	Row address strobe assertion	—	50	—	36	—	28

Num	Symbol	Description	10 MHz		14 MHz		18 MHz	
			Min	Max	Min	Max	Min	Max
73B	tRASHRASL	Row address strobe precharge	(D) −5	—	(D) −4	—	(D) −4	—
73C	tC2XHRASH	Row address strobe negation	—	50	—	36	—	28
74	tRASLRASH	Address strobe assertion time	(E) −5	—	(E) −4	—	(E) −4	—
75A	tC2XHCASL	Column address strobe assertion	—	41	—	29	—	23
75B	tC23HCASH	Column address strobe negation (synchronous) (R)	—	(Q) +41	—	(Q) +29	—	(Q) +23
75C	tC2XHCASH	Column address strobe negation	—	41	—	29	—	23
75D	tCASHCASL	Column address strobe precharge	(F) −5	—	(F) −4	—	(F) −4	—
75E	tCASLCASH	Column address strobe assertion width	(G) −5	—	(G) −4	—	(G) −4	—
75F	tCASLCASH	Column address strobe assertion width	(H) −5	—	(H) −4	—	(H) −4	—
75G	tC2XLCASL	Column address strobe assertion	—	(L) +69	—	(L) +49	—	(L) +38
76A	tC2XHFATL	Fault assertion	—	(J) +(K) +55	—	(J) +(K) +39	—	(J) +(K) +31
76B	tCLXHFATL	Fault assertion	—	55	—	39	—	31
77A	tDSBHFATZ	Fault HI-Z	—	55	—	39	—	31
78A	tG10H1EOL	Address strobe negation time set-up	(M) −10	—	(M) −7	—	(M) −6	—
78B	tC2XHG10L	Output enable assertion	—	55	—	40	—	30
78C	tC2XHG10H	Output enable negation	—	50	—	36	—	28
79	tC23HSRYH	Synchronous data ready HI-Z	—	55	—	39	—	31
80A	tC2XHDTAL	Data acknowledge assertion no EDC OR with EDC & no error detected	—	(J) +50	—	(J) +36	—	(J) +28
80B	tC2XHDTAI	Data acknowledge assertion with EDC & correctable error detected	—	(J) +(K) +(L) +(M) +50	—	(J) +(K) +(L) +(M) +36	—	(J) +(K) +(L) +(M) +28
80C	tC2XHDTAL	Data acknowledge assertion	—	50	—	36	—	28
82	tDSHBDTAH	Data acknowledge HI-Z	—	55	—	39	—	31
83	tC23HSRYL	Synchronous data ready assertion	—	50	—	36	—	28

Num	Symbol	Description	10 MHz		14 MHz		18 MHz	
			Min	Max	Min	Max	Min	Max
84A	tWEOLCASL	Write enable set-up time no EDC OR with EDC & full write	30	—	24	—	17	—
84B	tIEOLWEOL	Input enable set-up time	(L) −10	—	(L) −7	—	(L) −6	—
84C	tC2XHWEOL	Write enable assertion	—	50	—	36	—	28
84D	tC2XHWEOH	Write enable negation	—	50	—	36	—	28
85	tCASHWEOH	Write enable hold time	30	—	24	—	17	—
86	tG10LCASL	DRAM output enable set-up time	0	—	0	—	0	—
87	tC2XHBYKL	Bypass acknowledgement assertion	—	50	—	36	—	28
88	tC2XHBYKH	Bypass acknowledgement negation	—	50	—	36	—	28
89A	tC2XHECWL	EDC write enable assertion (full write)	—	50	—	36	—	28
89B	tC2XHECWL	EDC write enable assertion (partial write, no error)	—	(J) +50	—	(J) +36	—	(J) +28
89C	tC2XHECWL	EDC write enable (partial write, correctable error)	—	(J) +(K) +50	—	(J) +(K) +36	—	(J) +(K) +28
89D	tC2XHECWH	EDC write enable negation	—	50	—	36	—	28
90	tCASHECWH	EDC write enable hold time	30	—	24	—	17	—
92	tC24LBKSL	Block access assertion	—	50	—	36	—	28
93	tCSOHBKSH	Block access HI-Z	—	55	—	39	—	31
94	tCSOLOEOL	Output enable assertion	—	55	—	40	—	30
95	tSDBHOEOH	Output enable negation	—	65	—	47	—	36
96A	tC2XHIEOL	Input enable assertion	—	50	—	36	—	28
96B	tC2XHIEOL	Input enable assertion	—	(M) +50	—	(M) +36	—	(M) +28
96C	tC2XHIEOH	Input enable negation	—	50	—	36	—	28
97	tCASHIEOL	Input enable hold time	30	—	24	—	17	—
98A	tHIGZOUTZ	All outputs 3-stated	—	100	—	72	—	56
99A	tC2XHENBL	Enable port B assertion	—	50	—	36	—	28
99B	tC2XHROBL	Slave request port B assertion	—	50	—	36	—	28
100A	tC2XHENBH	Enable port B negation	—	50	—	36	—	28
100B	tCASHHIGV	Master HI-Z delay (dual port)	190	—	135	—	104	—
100C	tENBLSHOV	Slave signals exit HI-Z (dual port)	190	—	135	—	104	—
100D	tADOVRASL	Row address assertion delay	50	—	28	—	20	—
100E	tENBLRASL	Row address slave assertion	240	—	171	—	131	—
100F	tRQBHENBH	Enable negation	140	—	100	—	76	—
100G	tRQBHMHOV	Master outputs exit HI-Z	140	—	100	—	76	—

Num	Symbol	Description	10 MHz		14 MHz		18 MHz	
			Min	Max	Min	Max	Min	Max
100H	tCASHADDZ	Slave 3-state memory	390	—	278	—	215	—
100I	tRASHADOZ	Slave 3-state memory	290	—	206	—	159	—
101A	tC2XHWZOL	Write zero assertion	—	50	—	36	—	28
101B	tC2XHWZOH	Write zero negation	—	50	—	36	—	28
102	tCASHWZOH	Write zero hold time	0	—	0	—	0	—
103	tDATVSRYL	Data bus set-up time	0	—	0	—	0	—
104	tDSBHDATZ	Data bus HI-Z	—	50	—	36	—	28
105A	tC2XHBMOL	Byte mark assertion time	—	50	—	36	—	28
105B	tBMOLCASL	Byte mark set-up time	30	—	24	—	17	—
105C	tC2XHBMOH	Byte mark negation	—	50	—	36	—	28
106	tCASHBMOH	Byte mark hold time	30	—	24	—	17	—
107A	tBYKHRASZ	Master signals of note T exit HI-Z after $\overline{\text{BYPASAK}}$ negation	—	50	—	36	—	28
107B	tADOZENBL	Set-up signals of note T enter HI-Z	50	—	36	—	28	—
108	tC2XHADOL	Address out (master) exits HI-Z	0	50	0	36	0	28
109A	tC2XHSHOV	Slave signals of note T exit HI-Z after $\overline{\text{PAS}}$ assertion	0	—	0	—	0	—
109B	tENBLSHOZ	Slave signals of note T enter HI-Z after enable negation	0	—	0	—	0	—
120	tC23L1C23H2	Clock 23 rise time	—	4	—	4	—	4
121	tC23H2C23L1	Clock 23 fall time	—	4	—	4	—	4
122	tC23L1C23L1	Clock 23 period (Tc)	100	—	71.4	—	55.6	—
123	tC24L1C34H2	Clock 34 rise time	—	4	—	4	—	4
124	tC34H2C34L1	Clock 34 fall time	—	4	—	4	—	4
125	tC34L1C34L1	Clock 34 period (Tc)	100	—	71.4	—	55.6	—
126	tC23J	Clock 23 jitter	—	0.5	—	55.6	—	0.5
127	tC23E	Clock 23 duty cycle error	—	2	—	2	—	2
128	tC34J	Clock 34 jitter	—	0.5	—	0.5	—	0.5
129	tC34E	Clock 34 duty cycle error	—	2	—	2	—	2
130	tSKEW	Clock skew	—	3	—	2	—	2

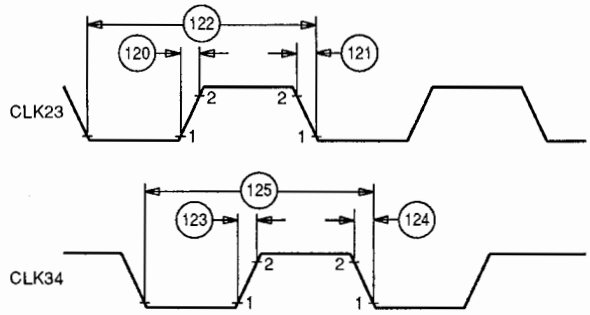
Notes:
1. This is an asynchronous signal; setup times are specified for testing and informational purposes. The setup times guarantee recognition at the next clock edge.
2. The rising edge of CLK2X is defined as the rising or falling edge of CLK34. The DRAM controller generates a 2X clock from CLK34 for internal use. Its outputs can therefore change on either the rising or falling edge of CLK34. The particular edge used depends on how the device has been programmed via the user-accessible internal register.
3. CLK2X transition which produces $\overline{\text{CASX}}$ assertion.
4. CLK2X transition which recognizes $\overline{\text{AS}}$ or $\overline{\text{PAS}}$ assertion.
5. CLK2X transition which produces $\overline{\text{RAS}}$ assertion.
6. $\overline{\text{AS}}$ or $\overline{\text{PAS}}$ negation, whichever comes earlier.
7. Minimum hold time from $\overline{\text{SRDY}}$, $\overline{\text{DTACK}}$, $\overline{\text{FAULT}}$ assertion is 2, 3, and 2 times note 3 CLK2X periods, respectively.
8. Tc is defined by timing characteristics 122 and 125 and Figure 31.

9. Set-up and hold times $\overline{ERRDT}$ are with respect to the clock boundary on which any $\overline{CAS0}$—$\overline{CAS3}$ is nominally asserted, plus the programmable AK1DEL for the first access, or plus the programmable AKNDEL for subsequent accesses in multiple accesses to the DRAM.
10. The rising edge of CLK23 following note 9 being met.
11. Note 9 plus ECDEL (programmable).
12. $\overline{LOCKA}$ must be set up before negation of $\overline{CS}$ of the first transaction in the locked sequence. $\overline{LOCKB}$ must be set up before negation of $\overline{REQB}$ of the first transaction in the locked sequence.
13. CLK2X transition which recognizes $\overline{PMCS}$ assertion.
14 CLK2X transition which asserts $\overline{SRDY}$ or $\overline{DTACK}$.

A. Signal 70 met if signal 5B is guaranteed.
B. Signal 71A met if signal 7A is guaranteed.
C. Determined by the number of half-cycles (Tc/2) programmed in the TRAH register.
D. Determined by the number of half-cycles (Tc/2) programmed in the TRP register.
E. Determined by the number of half-cycles (Tc/2) programmed in the TRAS register.
F. Determined by the number of half-cycles (Tc/2) programmed in the TNCP register.
G. Determined by the number of half-cycles (Tc/2) programmed in the TCAS register.
H. Determined by the number of half-cycles (Tc/2) programmed in the TNCAS register.
J. Determined by the number of half-cycles (Tc/2) programmed in the AK1DEL or AKNDEL register, whichever applies.
K. Determined by the number of half-cycles (Tc/2) programmed in the ECDEL register.
L. Determined by the number of half-cycles (Tc/2) programmed in the WRDEL register.
M. Determined by the number of half-cycles (Tc/2) programmed in the TTRIST register.
N. Whichever is later.
P. Whichever is earlier.
Q. Determined by the number interval programmed in the TCASNG register.
R. CLK23 transition which produces $\overline{SRDY}$ assertion.
S. CLK2X transition which recognizes $\overline{DS}$ assertion.
T. The following outputs are 3-stated by $\overline{HIGHZ}$ assertion: ADOUT0—ADOUT9, $\overline{RAS0}$—$\overline{RAS3}$, $\overline{CAS0}$—$\overline{CAS3}$, $\overline{BM0}$—$\overline{BM3}$, $\overline{WE}$, $\overline{G}$, $\overline{BYPASAK}$, $\overline{ENABLEB}$, $\overline{IE}$, $\overline{OE}$, $\overline{EDCWE}$, $\overline{WZ}$, $\overline{SRDY}$, $\overline{DTACK}$, $\overline{FAULT}$, $\overline{BLKACS}$, and DATA0—DATA7.
U. In dual-port configurations, the following signals are 3-stated by a quiescent slave DRAM controller and by a master DRAM controller and by a master DRAM controller that has relinquished ownership of the shared output lines to an active slave DRAM controller: ADOUT0—ADOUT9, $\overline{RAS0}$—$\overline{RAS3}$, $\overline{CAS0}$—$\overline{CAS3}$, $\overline{BM0}$—$\overline{BM3}$, $\overline{WE}$, $\overline{G}$, $\overline{IE}$, $\overline{EDCWE}$, and $\overline{WZ}$. The outputs which are not 3-stated by a master DRAM controller that has relinquished ownership of the shared output lines to an active slave DRAM controller are: $\overline{BYPASAK}$, $\overline{ENABLEB}$, $\overline{OE}$, $\overline{SRDY}$, $\overline{DTACK}$, $\overline{FAULT}$, $\overline{BLKACS}$, and DATA0—DATA7. The data I/O interface to the DRAM controller need not be 3-stated because it taps off on the CPU side of the data bus transceiver rather than the DRAM side of the data bus transceivers.

Timing Diagrams

These timing diagrams represent a subset of all possible transactions and are intended only to display and clarify timing relationships.

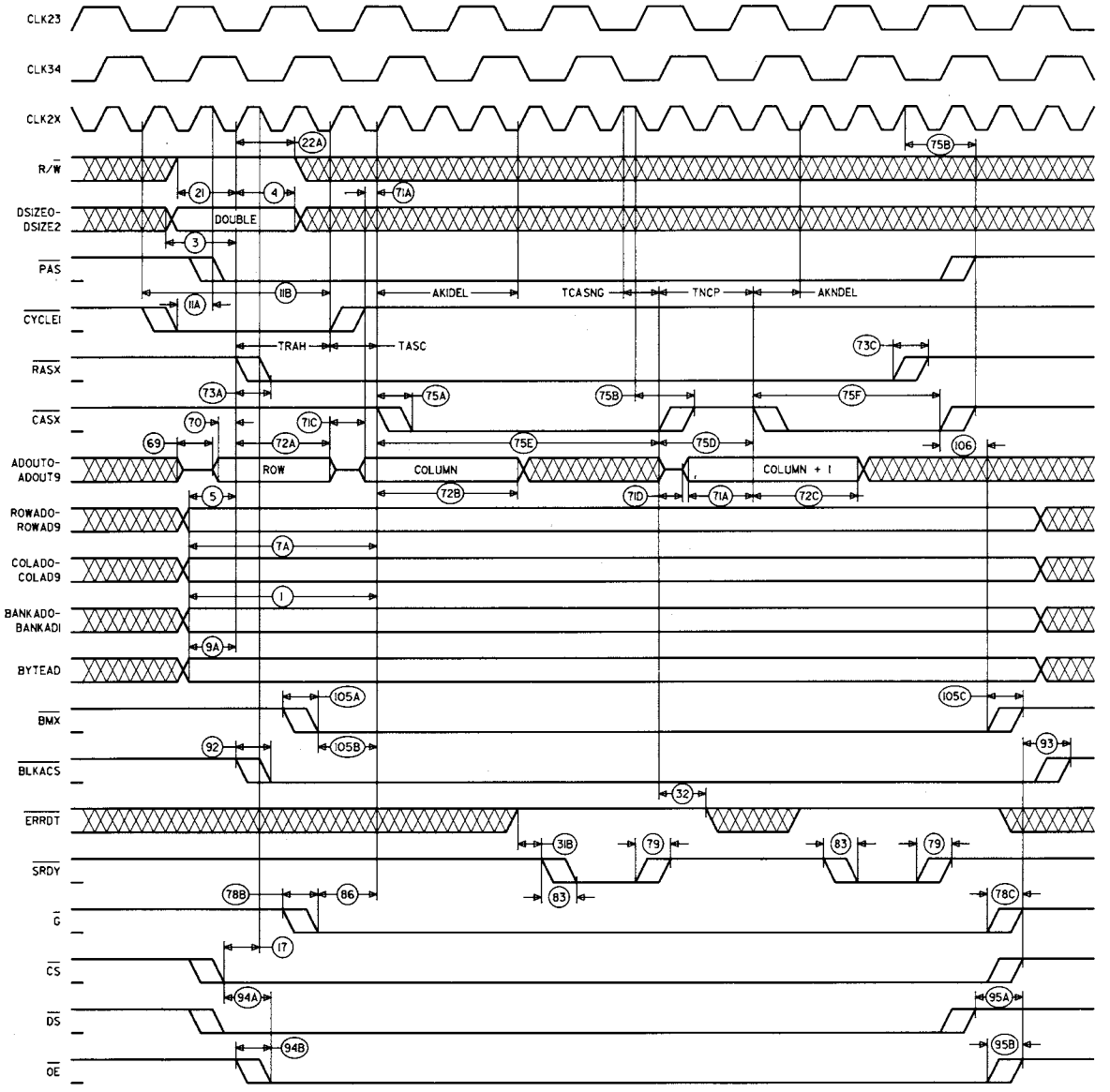


Notes:
Duty Cycle Error – The duty cycle of each clock input may deviate from 50% but should not exceed timing specification numbers 127 and 129 for both CLK23 and CLK34.

Skew – Nominally CLK23 leads CLK34 by 90°. This phase lead should not deviate more than timing specification number 130.

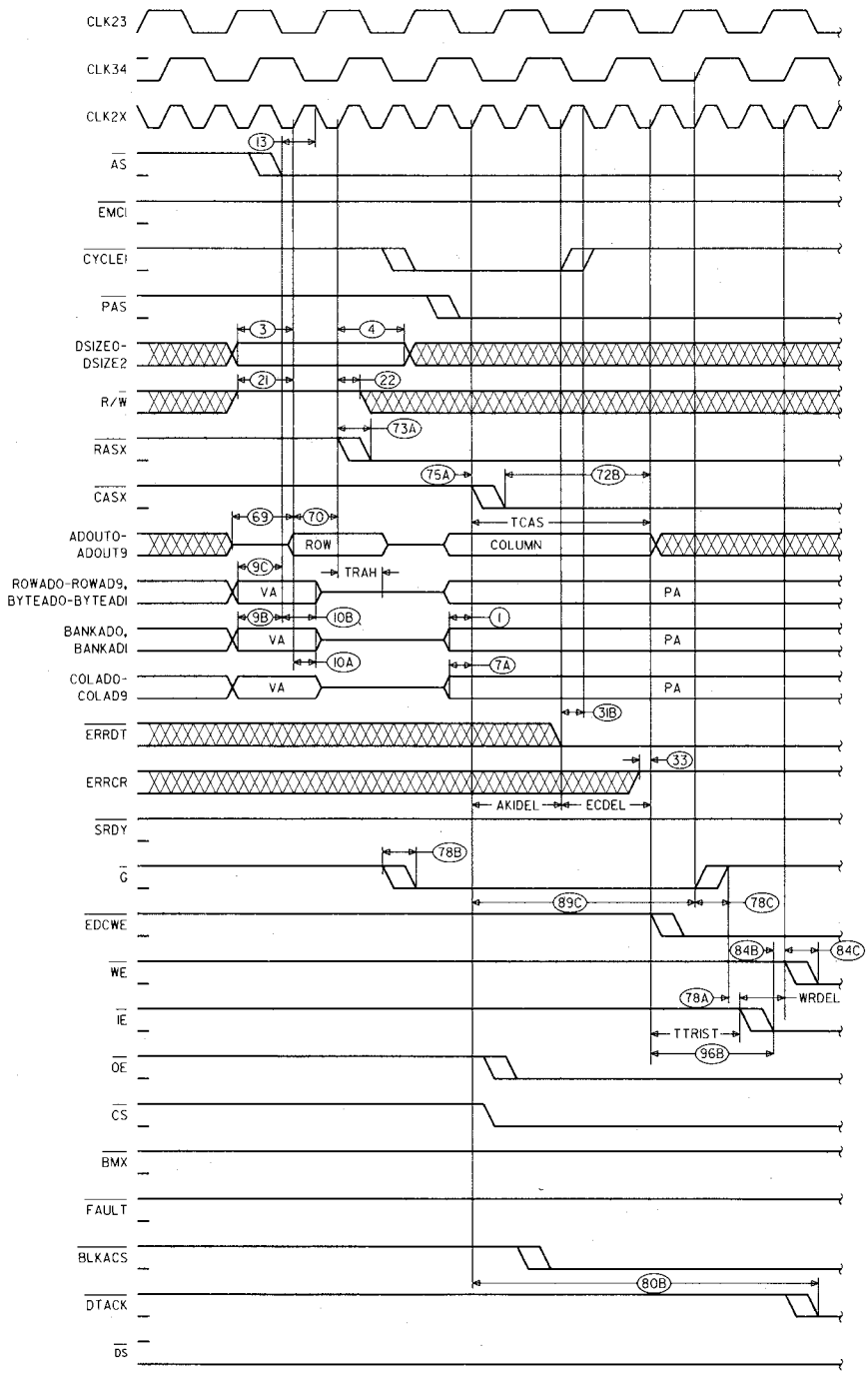
Jitter – The period of each clock input may deviate from its nominal value and should not exceed timing specification numbers 126 and 128 for both CLK23 and CLK34.

Figure 31. Clock Inputs



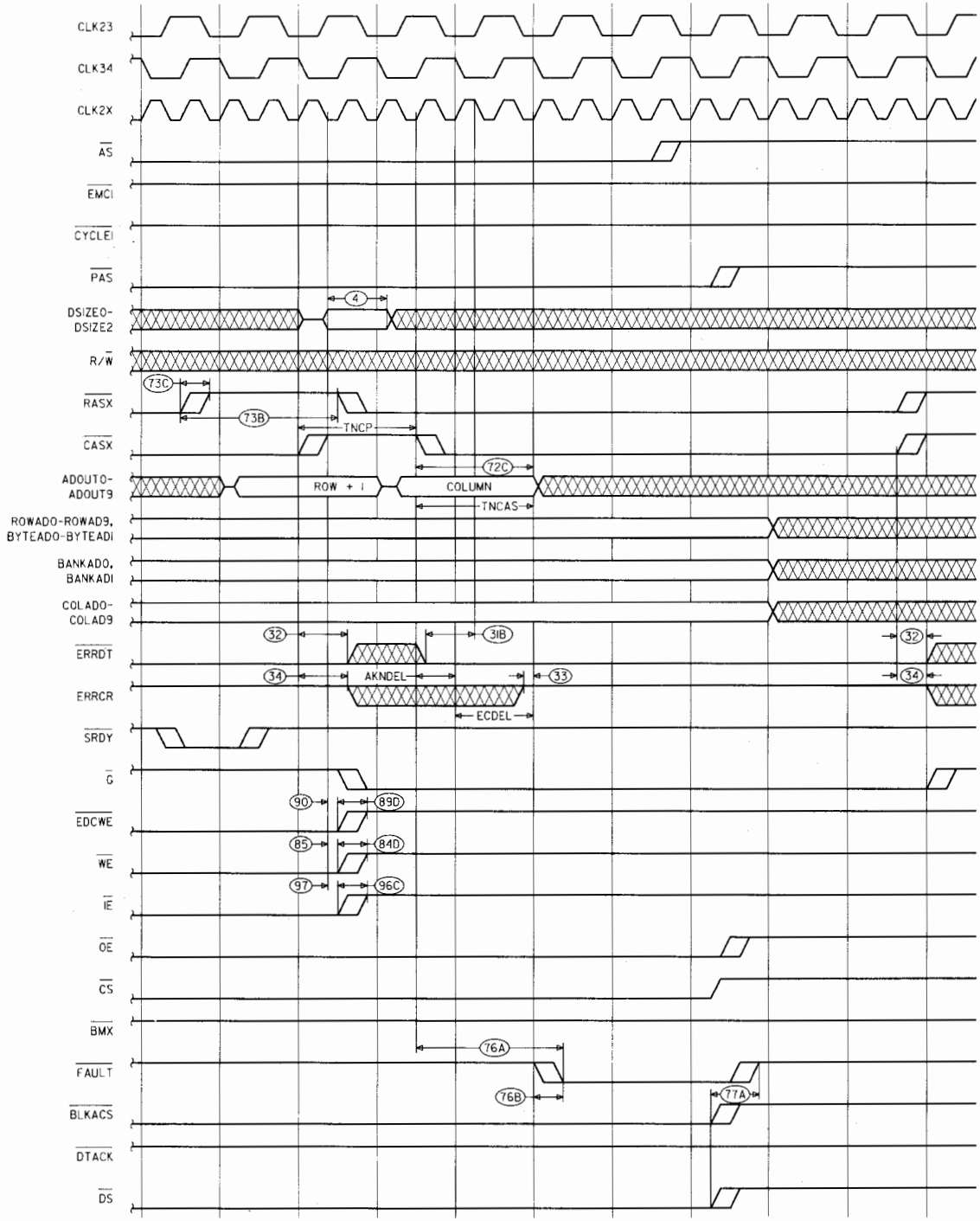
Note: TRAH=2; AKNDEL=1; TASC=1; TNCP=2; AKIDEL=3.

Figure 32. Synchronous Double-Word Read (No MMU, No EDC, PAGNIB Mode)



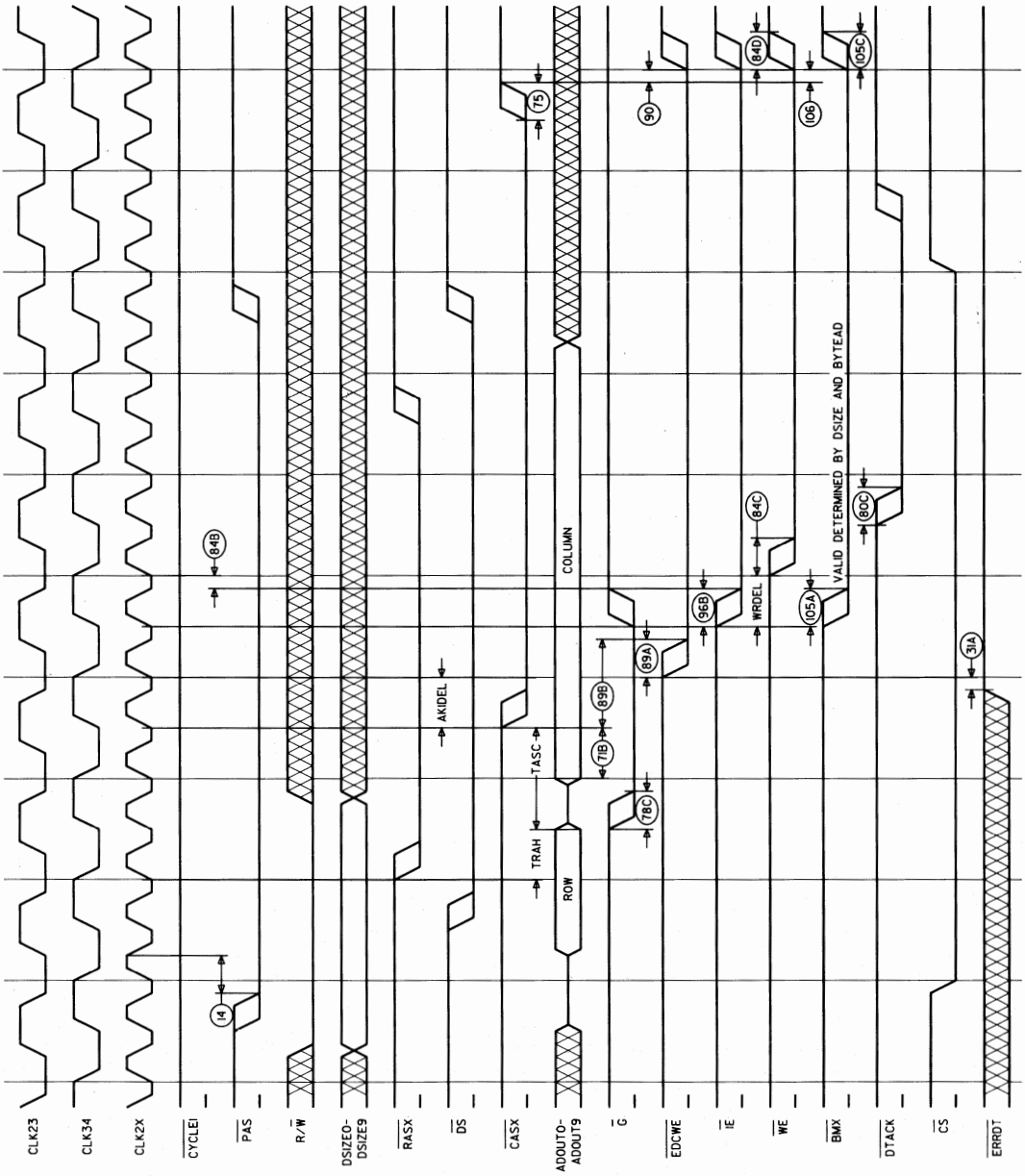
Note: DTACK would be asserted here if the access was to be asynchronous (no CYCLEI) correctable error on first word, uncorrectable error on second word.

Figure 33. Synchronous Double-Word Read (MMU, Pretranslation, EDC) (Sheet 1 of 2)



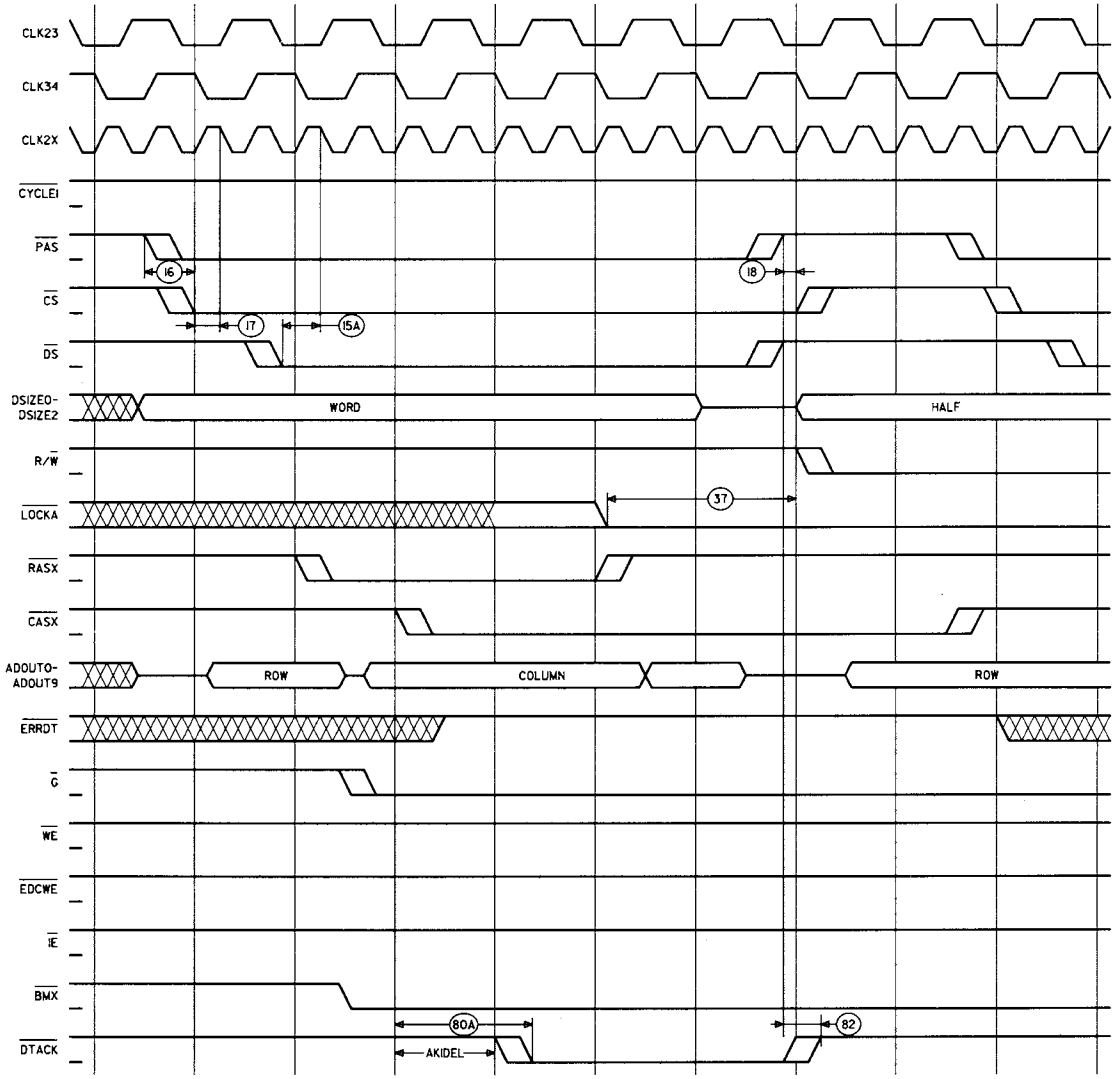
Note: Correctable error on first word, uncorrectable error on second word.

Figure 33. Synchronous Double-Word Read (Sheet 2 of 2)



Note: All programmable delays = minimum values.

Figure 34. Asynchronous Partial-Word Write (No MMU, EDC)



Note: All programmable delays = minimum values.

Figure 35. Asynchronous Interlocked Partial-Word Write (No MMU, No EDC) (Sheet 1 of 2)

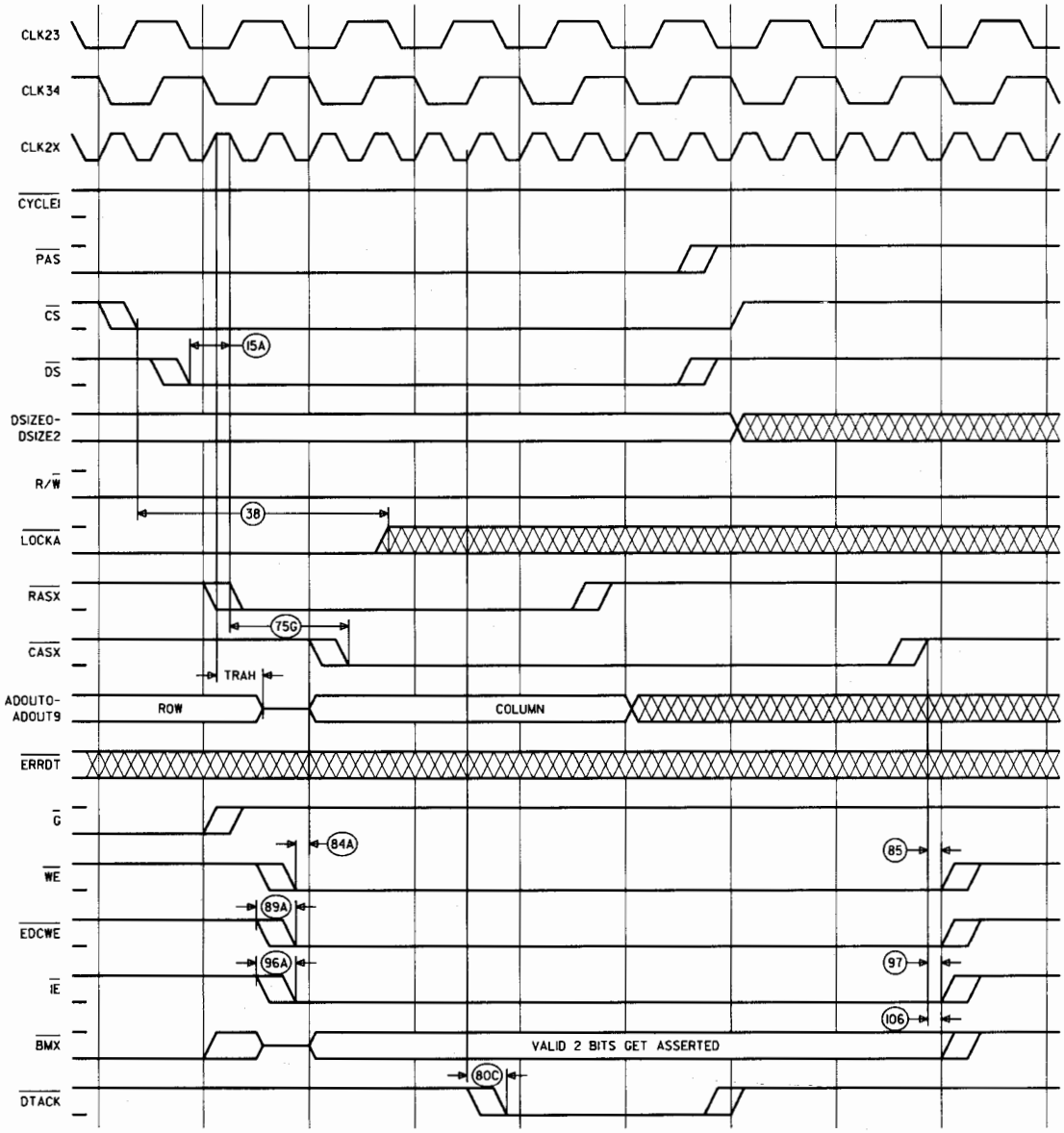


Figure 35. Asynchronous Interlocked Partial-Word Write (No MMU, No EDC) (Sheet 2 of 2)

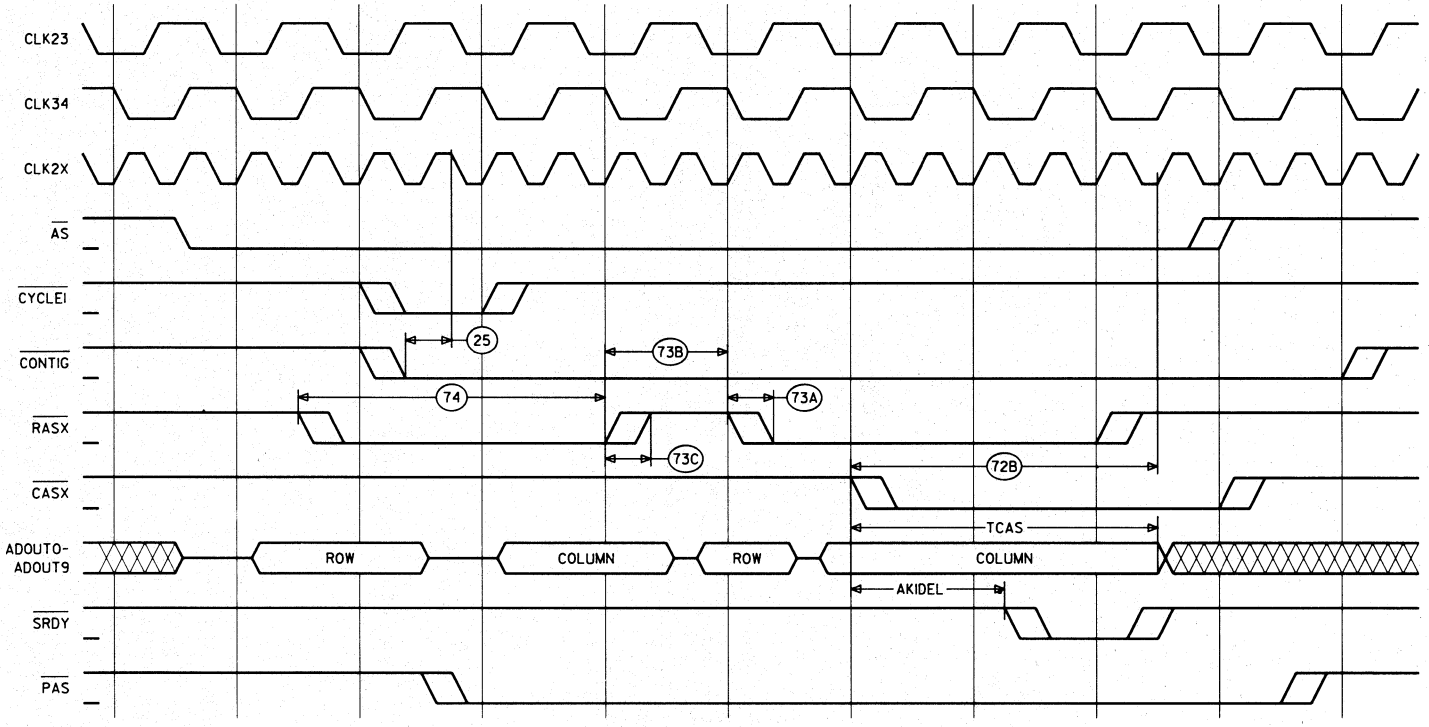


Figure 36. Read-Synchronous Pretranslation $\overline{\text{CONTIG}}$

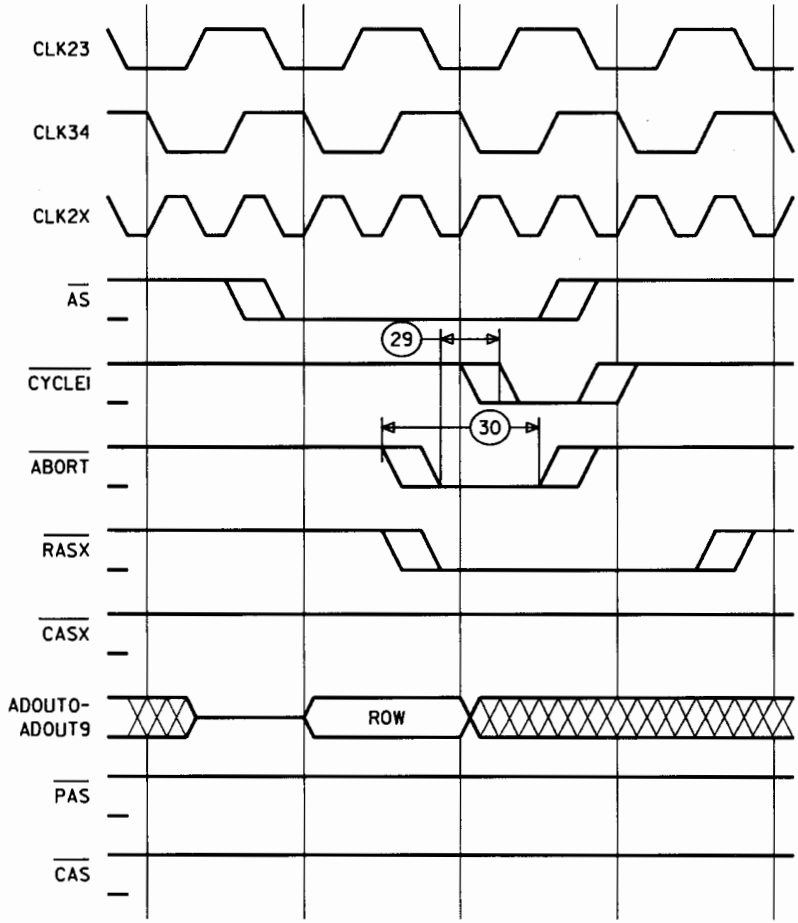


Figure 37. Read-Synchronous Pretranslation ABORT

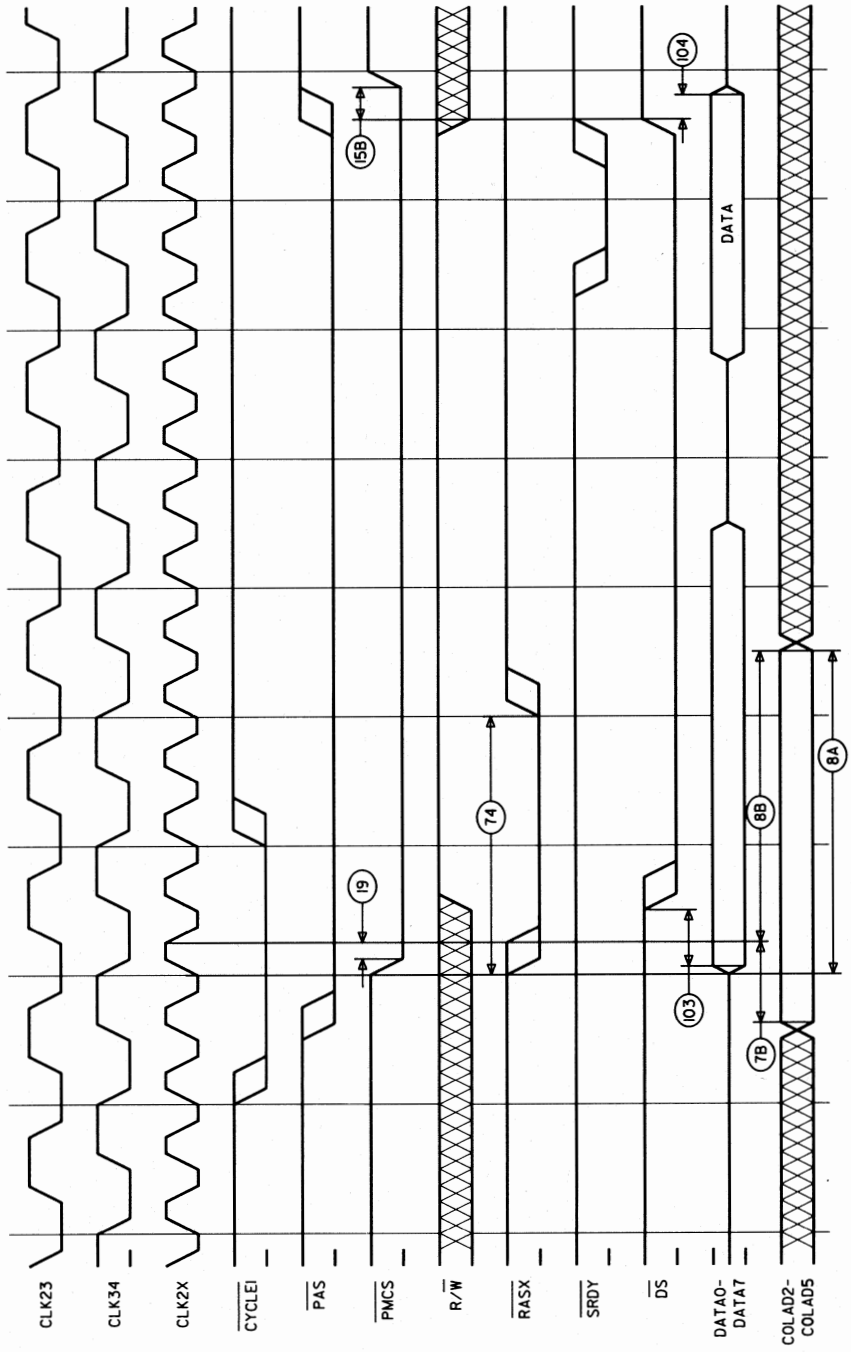


Figure 38. Synchronous-Peripheral Read After Leaving Quiescent Mode

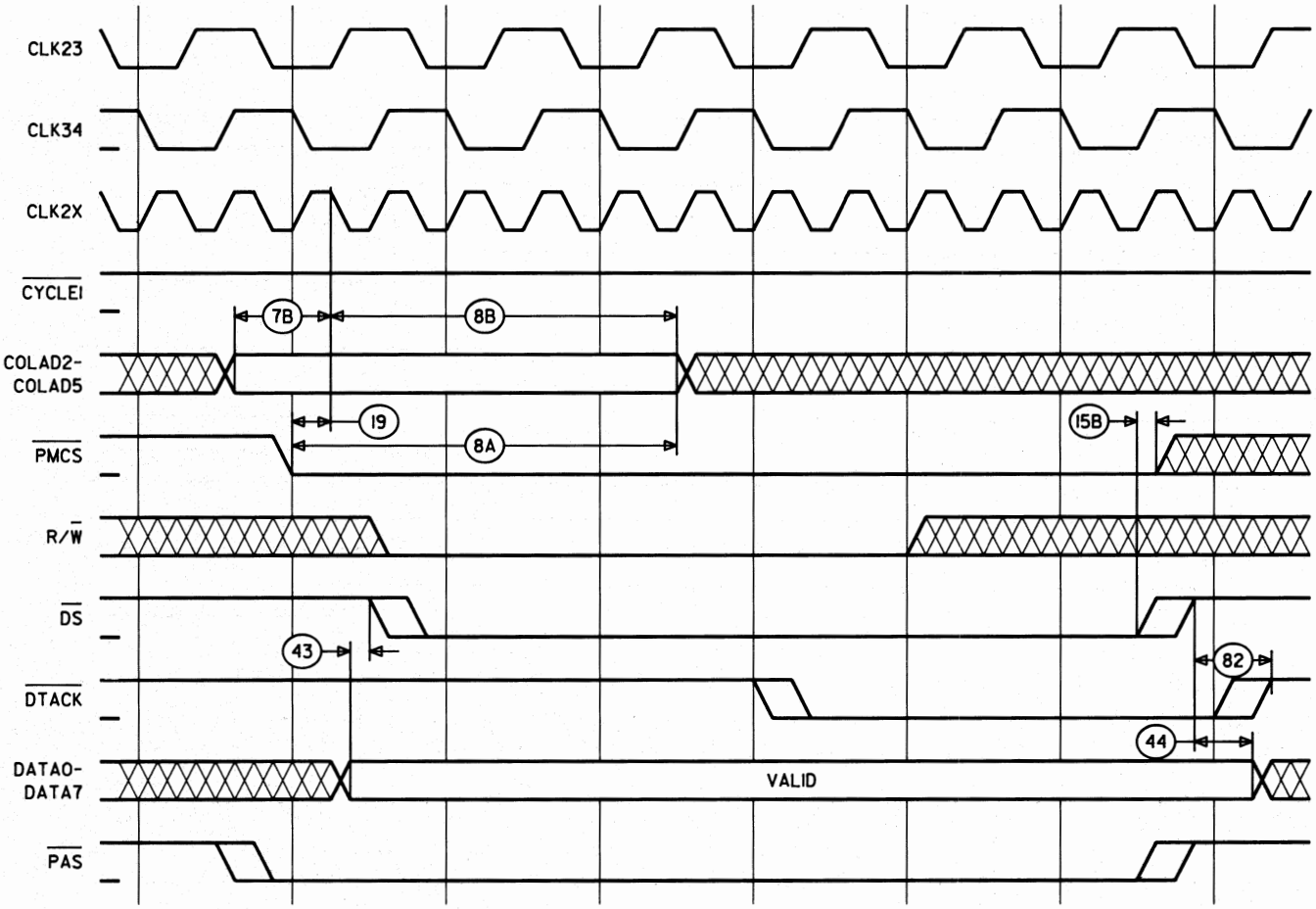


Figure 39. Peripheral Write Asynchronous (Quiescent State)

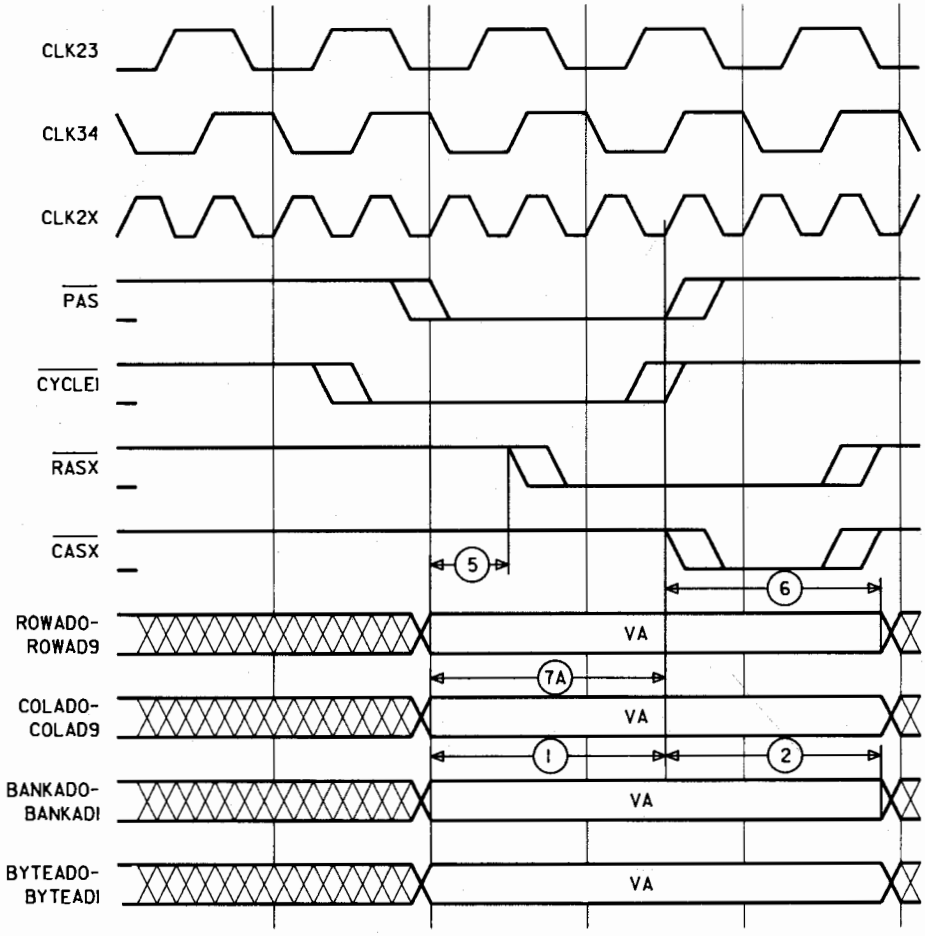


Figure 40. Synchronous Access with Bus Exception

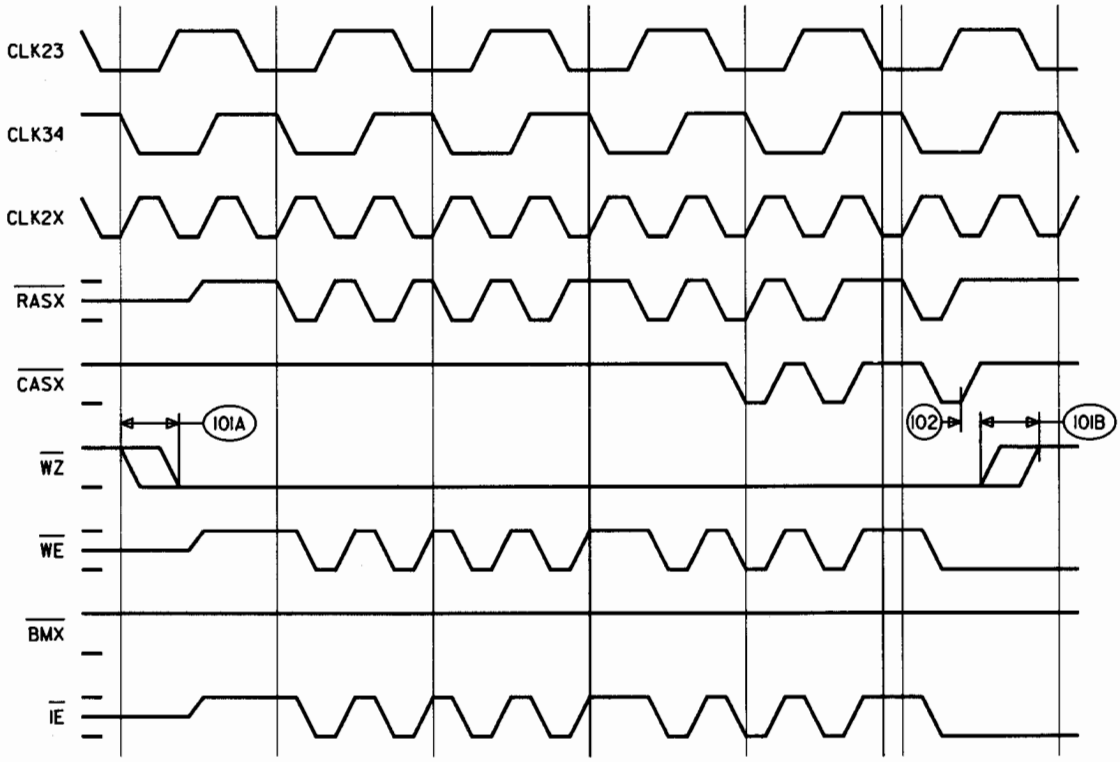
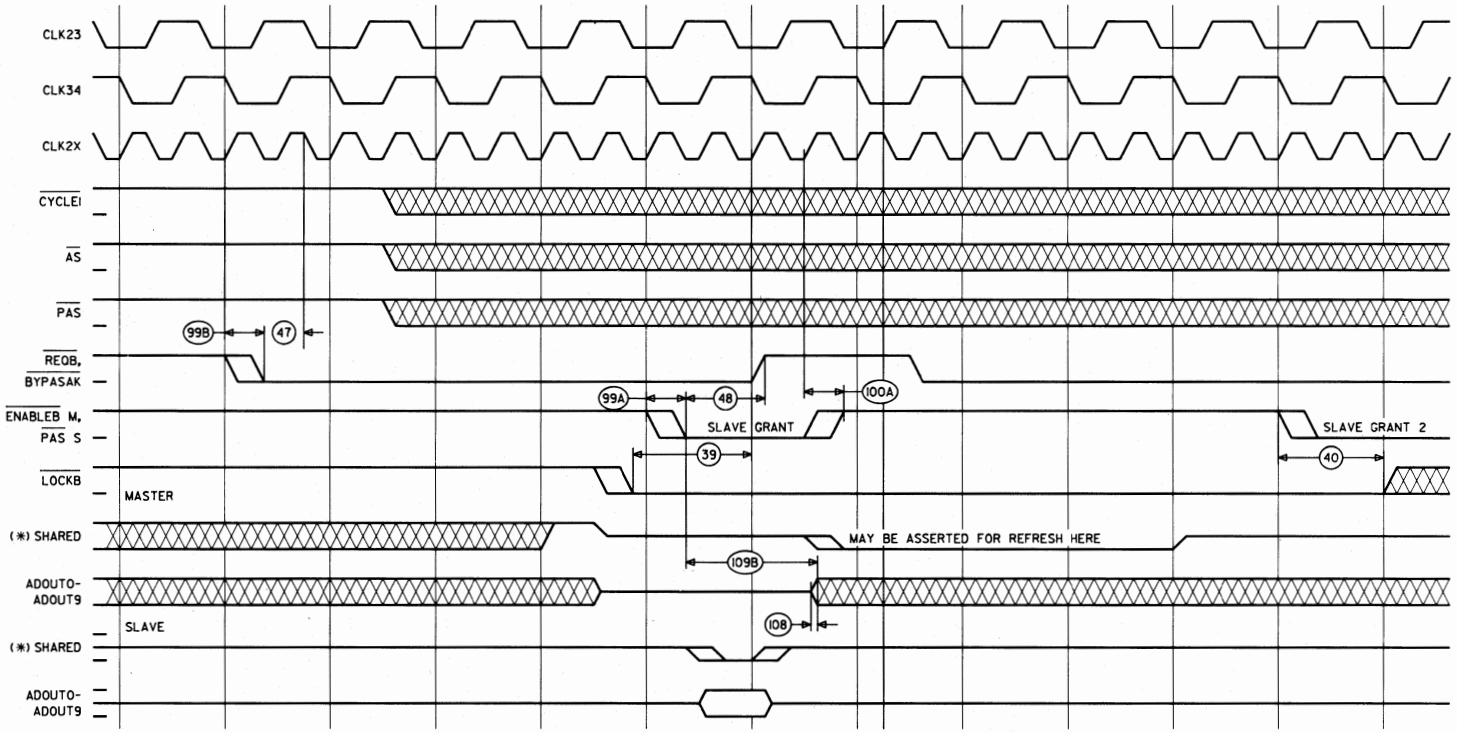


Figure 41. Memory Initialization Sequence



Note: Asterisks indicate shared outputs – $\overline{WE}$, $\overline{RASX}$, $\overline{CASX}$, $\overline{EDCWE}$, $\overline{G}$, $\overline{IE}$; S – indicates slave signals; M – indicates master signals.

Figure 42. Dual Port (Slave Read, Interlocked, Write, $\overline{LOCKB}$ Asserted)

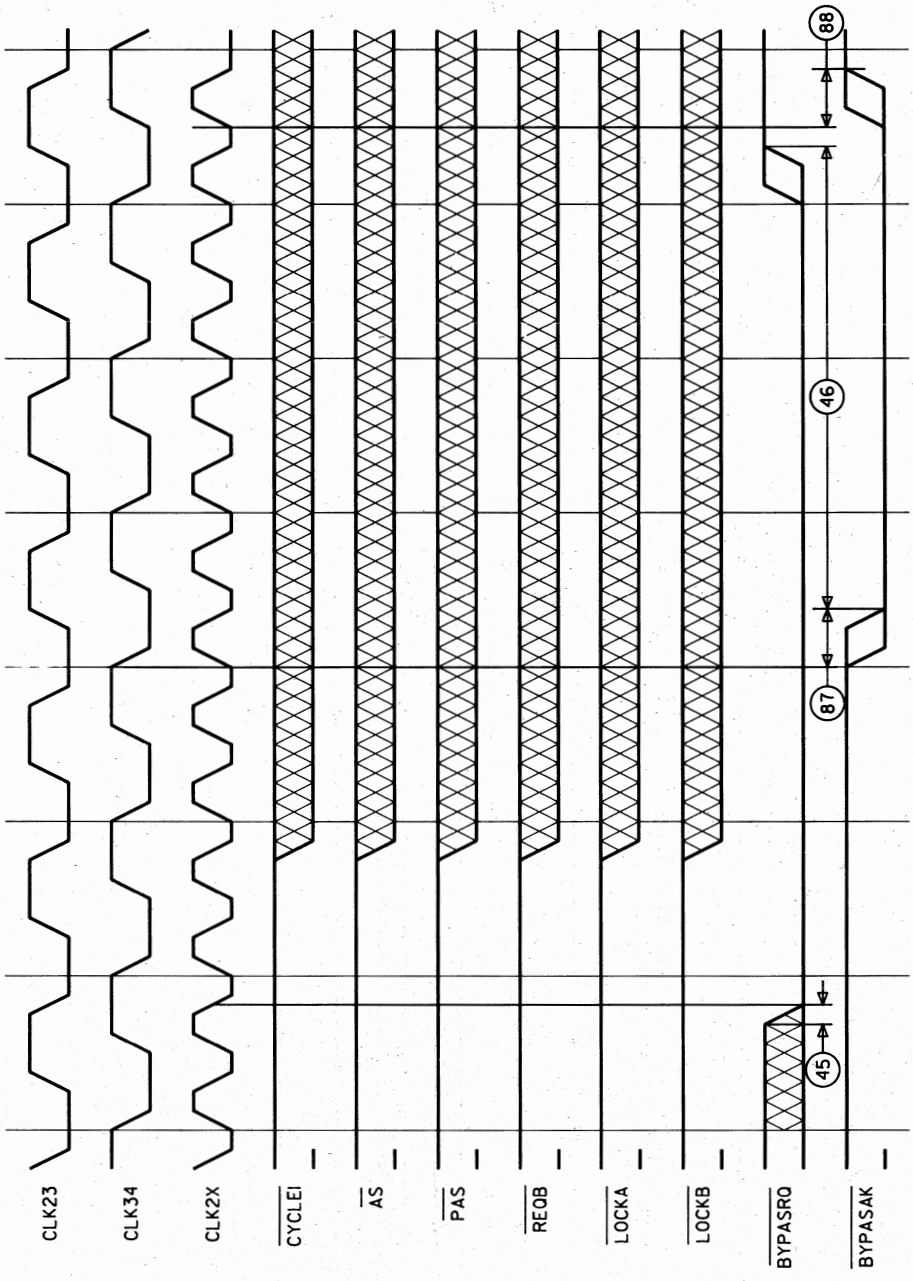
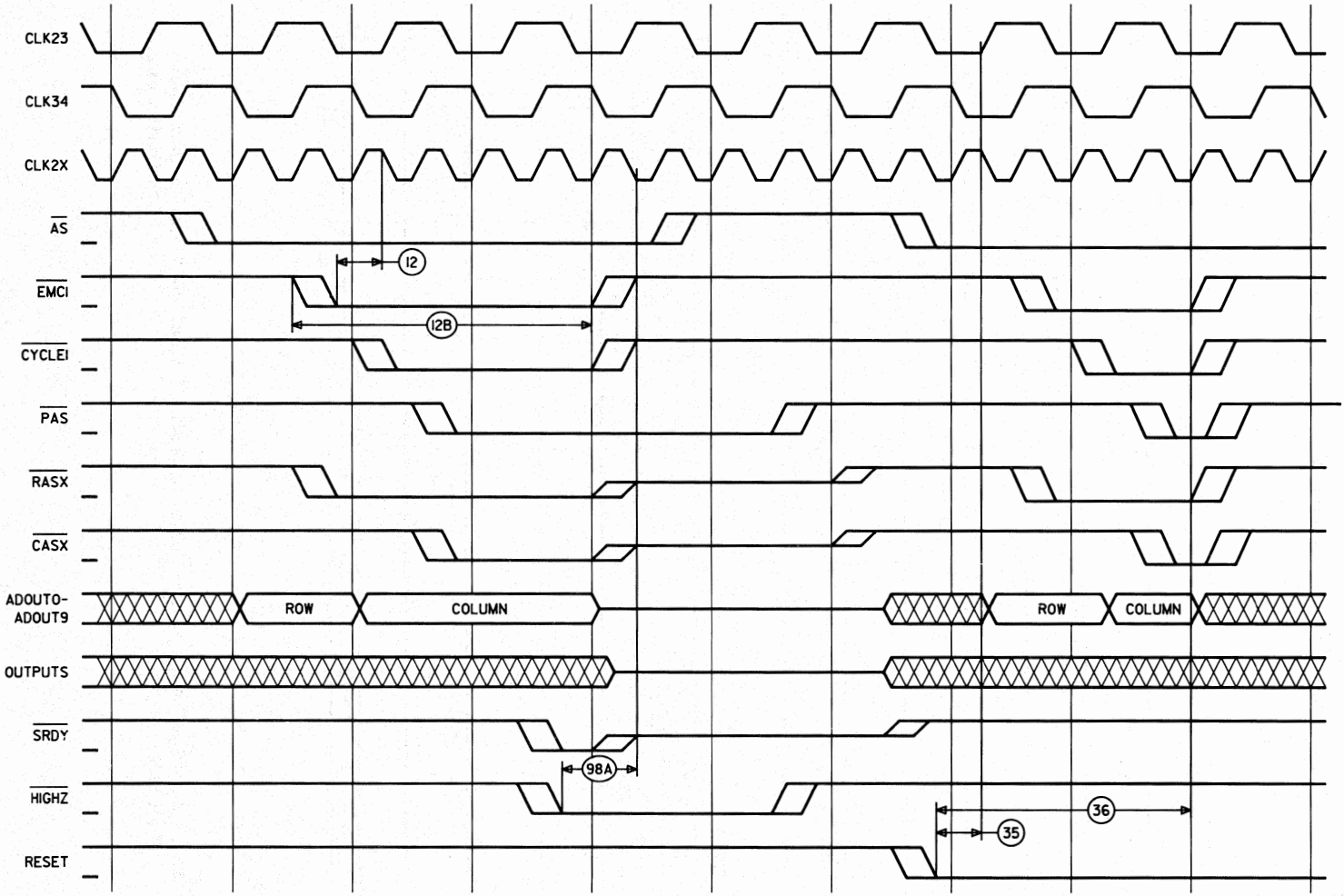


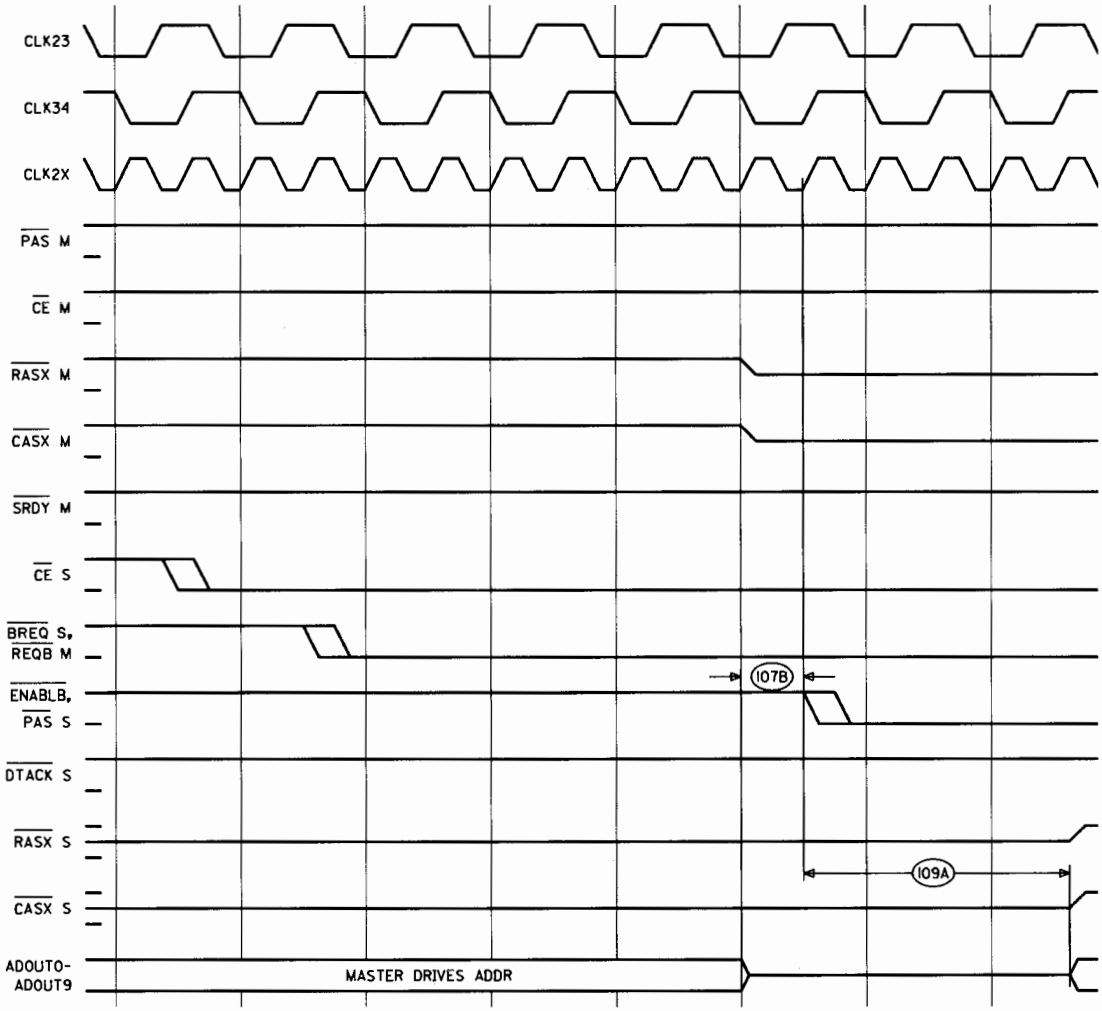
Figure 43. Bypass Access Sequence



Note: Applying RESET turns a prior master DRAM controller into a slave DRAM controller. If prior to RESET, master granted shared memory to slave, RESET will negate $\overline{\text{ENABLEB}}$. This output driver is not 3-stated and is only asserted if DRAM controller is a master.

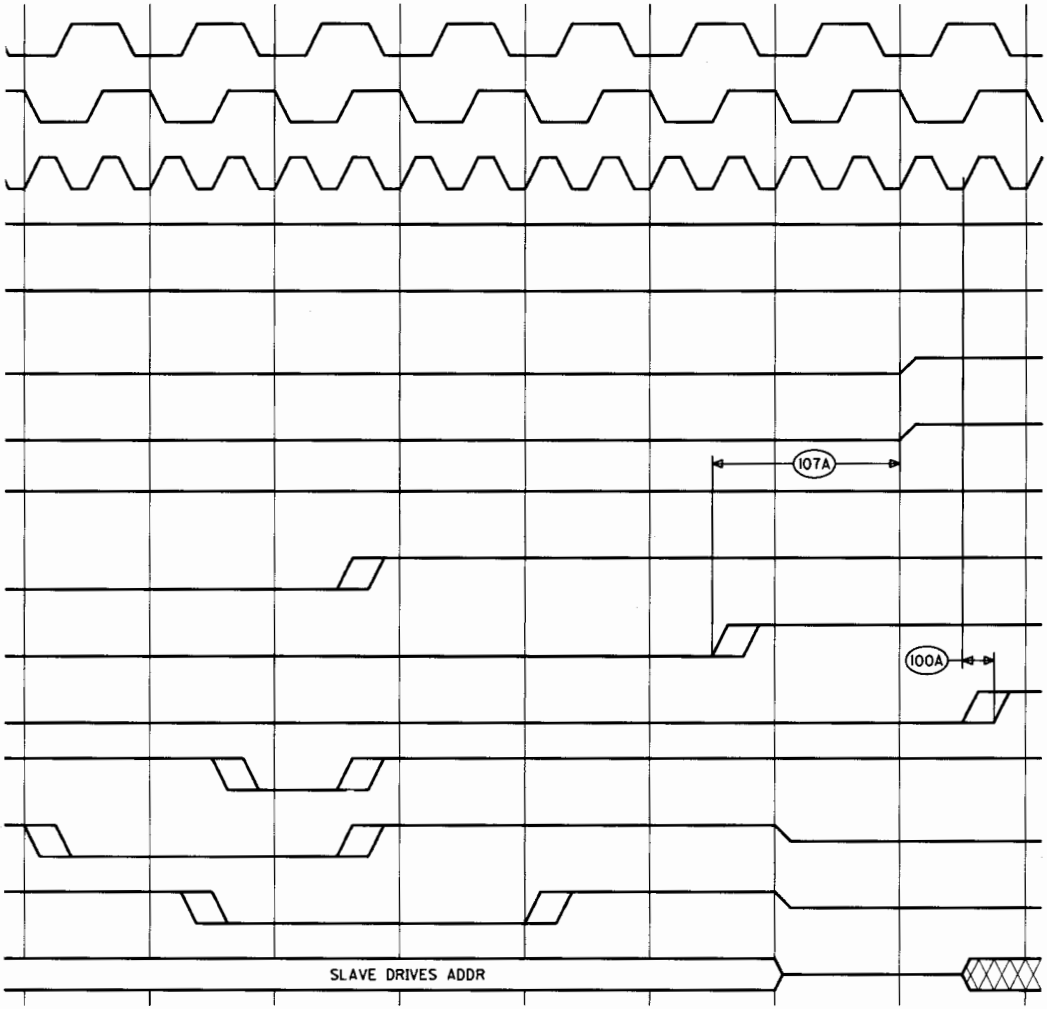
Figure 44. High Impedance and Reset Sequence

Figures 45 through 47 are contained on the following six pages. For convenient reading, each drawing consists of two facing pages.



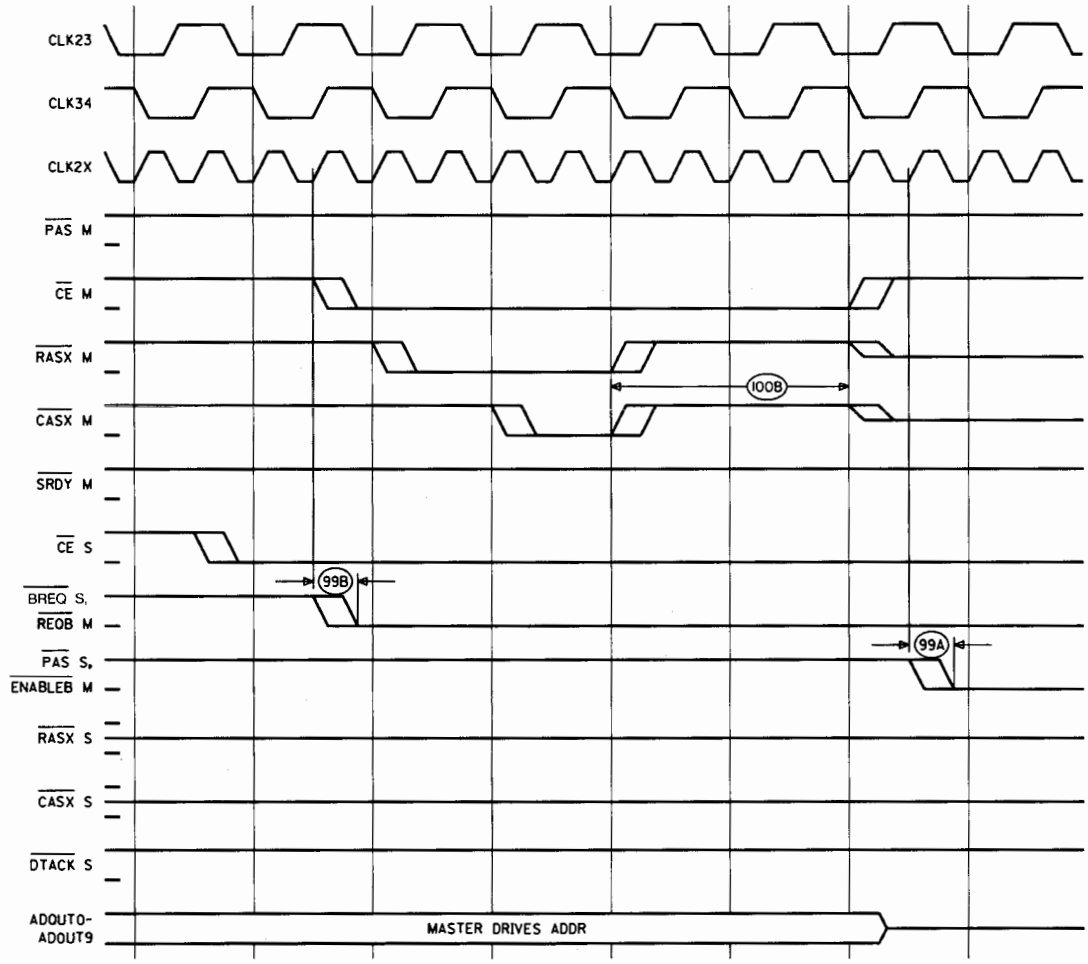
Note: S – indicates slave signals; M – indicates master signals.

Figure 45. Memory Bus Transfer From Master to Slave – Master Idle When Request Issued (Left Side) (Sheet 1 of 2)



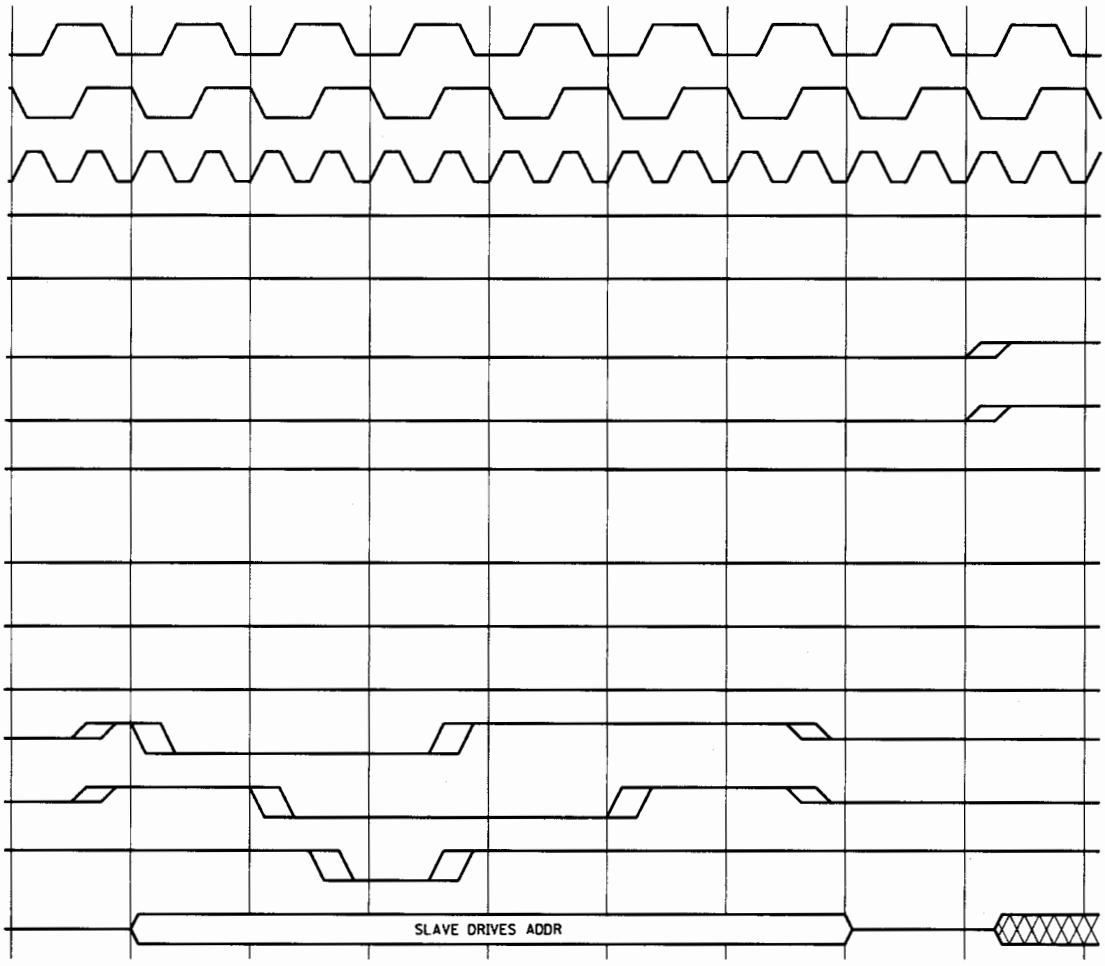
Note: S – indicates slave signals; M – indicates master signals.

Figure 45. Memory Bus Transfer From Master to Slave – Master Idle When Request Issued (Right Side) (Sheet 2 of 2)



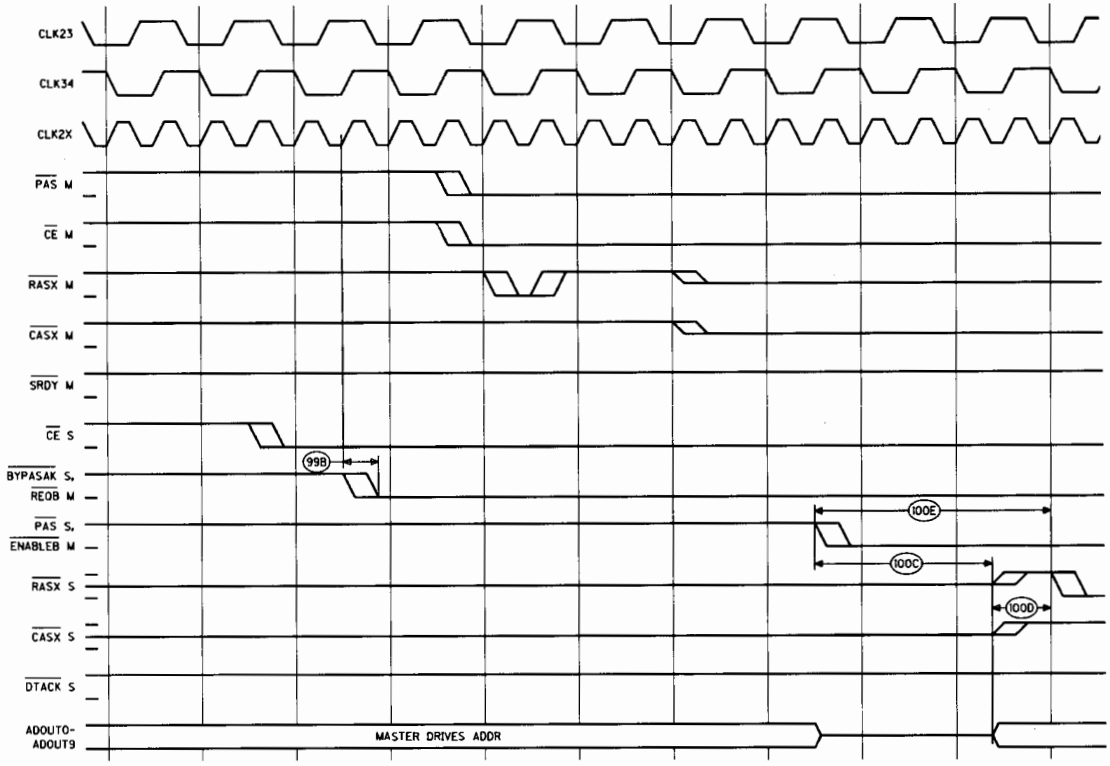
Note: S – indicates slave signals; M – indicates master signals.

Figure 46. Memory Bus Transfer From Master to Slave – Master $\overline{CASX}$ to Slave $\overline{RASX}$ Timing (Left Side) (Sheet 1 of 2)



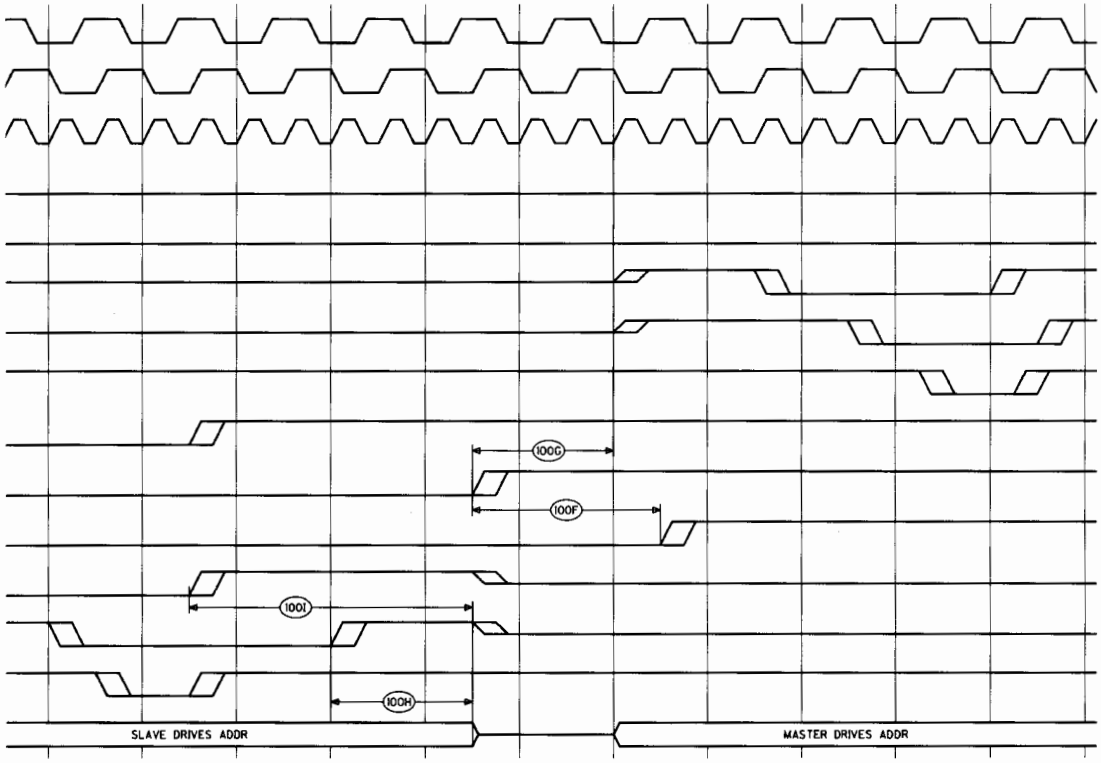
Note: S – indicates slave signals; M – indicates master signals.

Figure 46. Memory Bus Transfer From Master to Slave – Master CAS to Slave RAS Timing (Right Side) (Sheet 2 of 2)



Note: S – indicates slave signals; M – indicates master signals.

Figure 47. Memory Bus Transfer From Master to Slave (After REFRESH) to Master (Left Side) (Sheet 1 of 2)



Note: S – indicates slave signals; M – indicates master signals.

Figure 47. Memory Bus Transfer From Master to Slave – (After REFRESH) to Master (Right Side) (Sheet 2 of 2)

Electrical Characteristics

Inputs

All inputs, except the two CMOS input clocks, are TTL compatible. The TTL input and clock inputs are specified separately below.

Table 19. DC Input Parameters

Inputs		Min	Nom	Max	Unit
TTL input voltage	high-level	2.0	—	V_{CC}+0.5	V
	low-level	−0.5	—	0.8	V
CMOS clocks input voltage	high-level	V_{CC}−1.3	—	V_{CC}+0.5	V
	low-level	0	—	0.8	V
TTL input loading current (2.0 V ≤ V_{IH} ≤ V_{CC})	high-level	0	—	0.01	mA
TTL input loading current (0 V ≤ V_{IL} ≤ 0.8 V)	low-level	−0.01	—	0	mA
CMOS clock input loading current (V_{CC}−1.3 V ≤ V_{IH} ≤ V_{CC})	high-level	0	—	0.01	mA
CMOS clock input loading current (0 V ≤ V_{IL} ≤ 0.8 V)	low-level	−0.01	—	0	mA

Outputs

The three classes of outputs that can support TTL input voltage levels are:

Class 1: This class is capable of driving one TTL load or eight PNP Schottky TTL loads; has current allowance for an external holding resistor employed in 3-state buffers. The minimum holding resistor value is 2.7 KΩ.

Class 2: This class is capable of driving 450 pF and does not have current allowance for a holding resistor.

Class 3: The signal in this class is an open drain output used for wired-logic operations, allowing more than one device to drive a node without conflict. An external resistor is required to pull this signal high. The minimum pull-up resistor value is 510 Ω.

The following lists the outputs assigned to each class:

Class 1		Class 2	Class 3
$\overline{BM0}$—$\overline{BM3}$	$\overline{IE}$	ADOUT0—ADOUT9	$\overline{BLKACS}$
$\overline{BYPASAK}$/$\overline{BREQ}$	$\overline{OE}$	$\overline{CAS0}$—$\overline{CAS3}$	$\overline{DTACK}$
DATA00—DATA07	$\overline{WZ}$	$\overline{G}$	$\overline{FAULT}$
$\overline{EDCWE}$		$\overline{RAS0}$—$\overline{RAS3}$	$\overline{SRDY}$
$\overline{ENABLEB}$		$\overline{WE}$	

Table 20. DC Output Parameters

Outputs		Min	Nom	Max	Unit
Output sink current (I_{OL}) ($V_{OL} \leq 0.4$ V)	Class 1	—	—	5.5	mA
	Class 2	—	—	30	mA
	Class 3	—	—	12.0	mA
Output source current (I_{OH}) ($V_{OH} \geq 2.4$ V)	Class 1	—	—	–5.5	mA
	Class 2	—	—	–50	mA
	Class 3	—	—	–10.0	μA
Output logic levels	high-level	2.4	—	—	V
	low-level	—	—	0.4	V

Operating Conditions

Table 21. DC Operating Conditions

Parameter		Symbol	Min	Nom	Max	Unit
Supply voltage		V_{CC}	4.75	5.00	5.25	V
Input load capacitance	TTL inputs	C_{IN}	—	—	12	pF
	CMOS clocks		—	—	7	pF
Total output load capacitance	Class 1	C_L	—	—	150	pF
	Class 2		—	—	450	pF
	Class 3		—	—	130	pF
Ambient temperature at the DRAM controller pins		T_A	0	—	70	°C
Humidity range		—	5%	—	95%	—
Power dissipation	10-MHz	PD	—	—	0.73	W
	14-MHz		—	—	1.0	W
	18-MHz		—	—	1.3	W
Operating frequency		F	—	—	18	MHz

Derating Curves

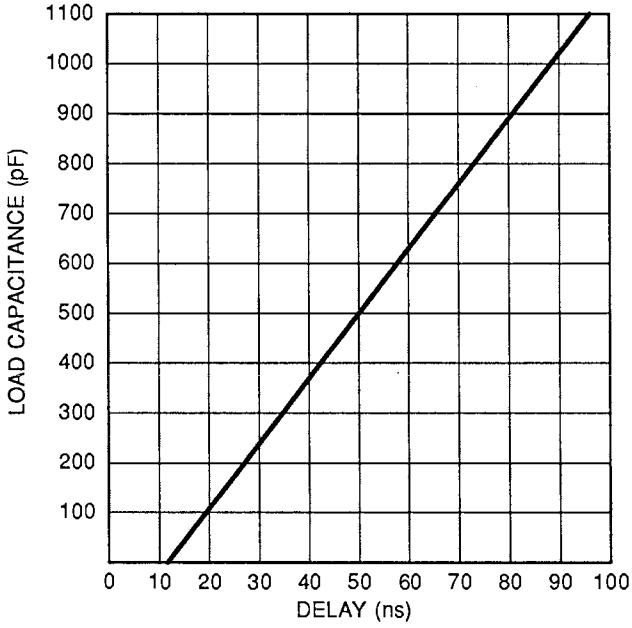


Figure 48. For Class 1 and 3 Output Buffer Capacitive Load vs. Delay – Worst Case at 105 °C For Output Transition From Low to High Level

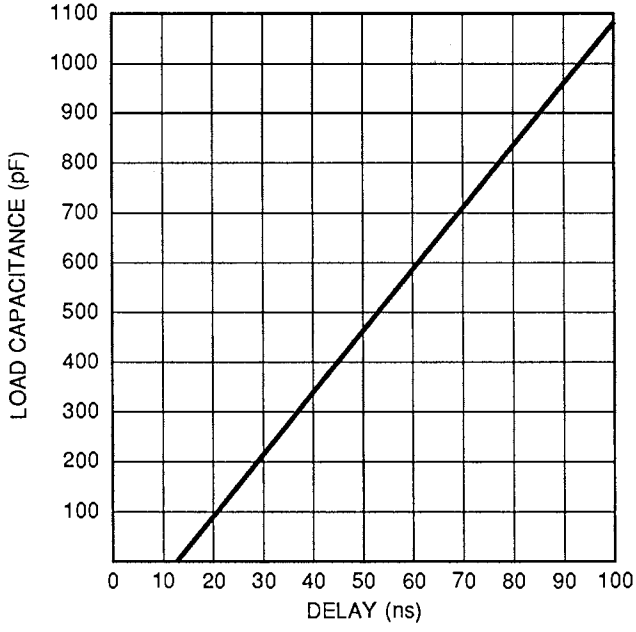


Figure 49. For Class 1 and 3 Output Buffer Capacitive Load vs. Delay – Worst Case at 105 °C For Output Transition From High to Low Level

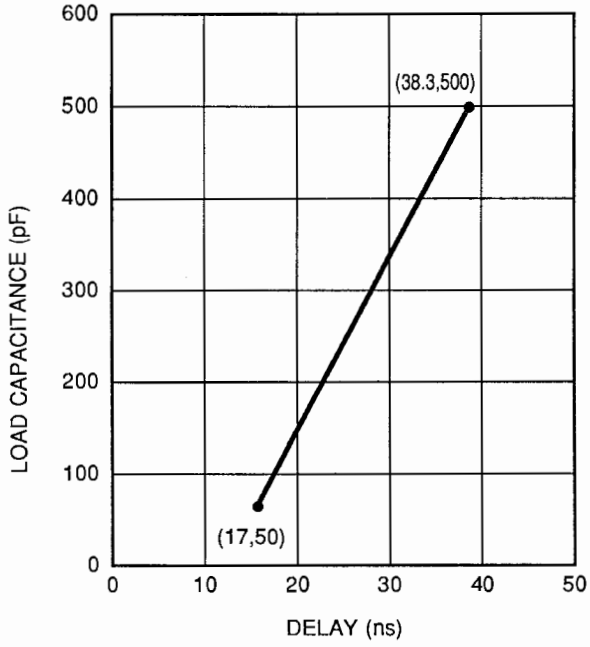


Figure 50. For Class 2 Output Buffer Capacitive Load vs. Delay – Worst Case at 105 °C For Output Transition From Low to High Level

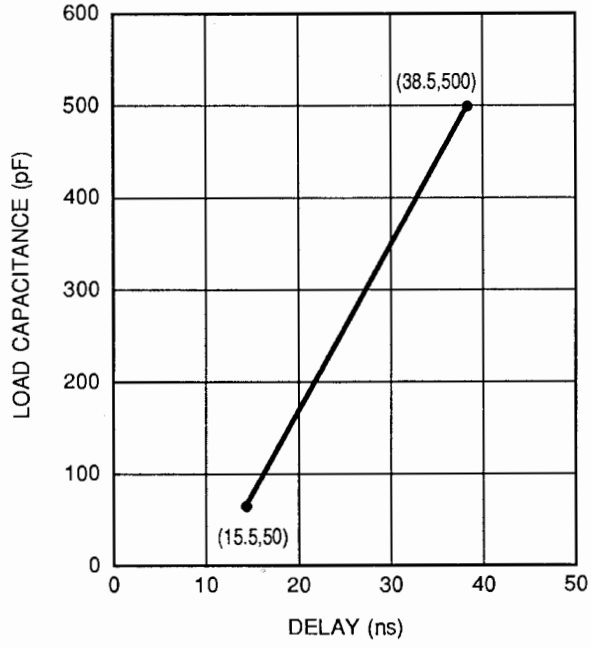
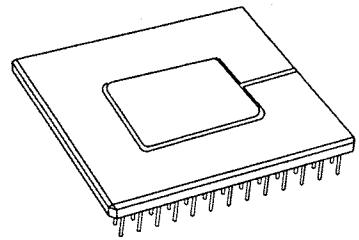


Figure 51. For Class 2 Output Buffer Capacitive Load vs. Delay – Worst Case at 105 °C For Output Transition From High to Low Level

WE® 32104 DMA Controller

Description

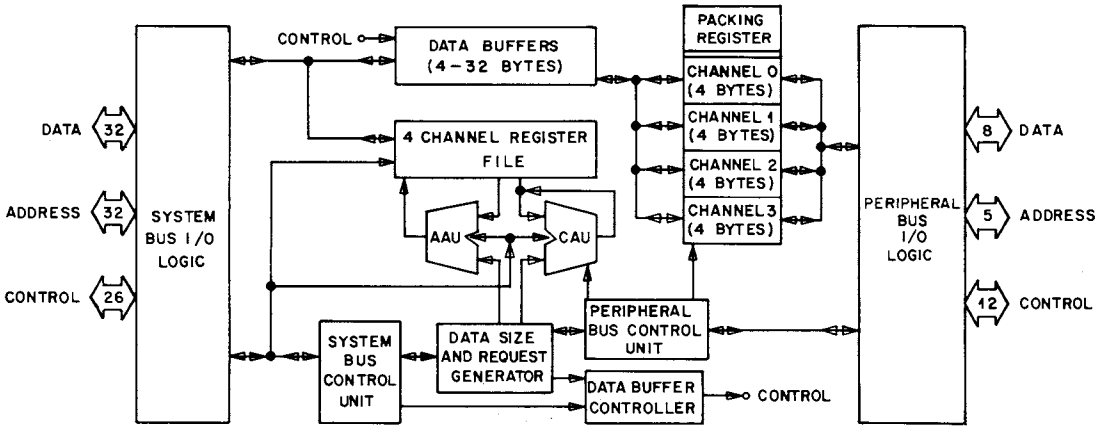
The *WE* 32104 DMA Controller (DMAC) is a memory-mapped peripheral device that performs memory-to-memory, memory fill, memory-to-peripheral, and peripheral-to-memory data transfers quickly and efficiently. The DMAC contains specialized hardware that permits data transfers at a rate much faster than a typical CPU. When used with the *WE* 32100 Microprocessor, the DMAC permits the full 32-bit width of the system bus to be utilized without external interfacing logic. In addition, a peripheral bus is provided to decouple 8-bit input/output devices from the system bus. The DMAC is implemented in CMOS technology, and is available in a 133-pin square, ceramic pin grid array (PGA) package. It is available in 10-, 14-, and 18-MHz frequency versions and requires a single 5 V supply.



Features

- Full 32-bit address and data buses
- 8-bit peripheral bus for decoupling I/O devices from the system bus
- Double- and quad-word bus cycles available for high system throughput
- Internal data buffers to support peripherals that transfer bursts of data
- Four independent DMA channels
- Fixed priority level for each DMA channel
- Two programmable interrupt vectors per channel
- Memory-to-memory transfers at rates up to 14.4 Mbytes/s at 18 MHz
- Memory-to-peripheral transfers at rates up to 7.8 Mbytes/s at 18 MHz (with burst mode)
- Memory fill operations available for writing an arbitrary constant to a block of memory

Description 313
Features 313
User Information 315
Architectural Summary 315
Register Organization 316
Per Channel Control Registers 316
Global DMAC Register 322
Data Buffer Registers 322
Operation 322
Channel Initialization 322
Data Transfer 324
Modes of Operation 324
Bus Requests 324
Operand Transfers 325
Data Packing and Unpacking 326
Address Alignment for Memory-to-Memory Transfers 326
Source and Destination Address Control .. 326
Peripheral Bus Transfers 326
Termination of Channel Operations 327
Interrupts 327
Bus Exceptions 328
Fault 328
Retry 328
Relinquish and Retry 328
Reset 328
Interfacing 329
Pin Descriptions 330
Numerical Order 331
Functional Groups 334
Characteristics 339
Timing Characteristics 339
Timing Diagrams 345
Electrical Characteristics 361
Inputs 361
Outputs 361
Operating Conditions 362



AAU - Address arithmetic unit.
CAU - count arithmetic unit.

Figure 1. WE® 32104 DMA Controller Block Diagram

User Information

The terms *asserted* and *negated* are used when describing signals. *Asserted* means that the signal is driven to its active state either by the DMAC (outputs) or an external device (inputs). *Negated* means that the signal is driven to its inactive state.

Direct memory access (DMA) is a mechanism for servicing I/O device data transfer requests. This mechanism is driven by the I/O device to provide direct access to system memory. Memory accesses can be interleaved with CPU execution via cycle sharing of the system bus, or the DMAC can obtain control of the system bus as in burst mode. When a transfer is requested of the DMA mechanism, it must:

1. Gain control of the system buses (address, data, and control)
2. Select desired memory location(s) and/or I/O device
3. Provide read and write control signals to perform the data transfer
4. Update memory address and transfer count for block transfers
5. Disable the memory and the I/O device and relinquish control of the system buses.

The *WE* 32104 DMA Controller provides this mechanism, while taking advantage of the full 32-bit data and address of the *WE* 32100 Microprocessor. The DMAC contains specialized hardware that permits transfers at a much faster rate than the microprocessor typically can.

Architectural Summary

The seven functional elements of the DMAC shown on Figure 1 are:

System Bus Interface – provides address, data, and control signals needed to interface the DMAC to the *WE* 32100 Microprocessor.

Data Buffers – each of the four channels has a 32-byte data buffer. The lowest address data buffer is used as the memory fill data register (MFDR). This register contains the data written to consecutive locations during memory fill operations.

Register File – the register file contains eight per channel control registers and one global register.

- The eight control registers are:
  - Source address register (SAR) – contains the address for source operands used during transfers out of memory.
  - Destination address register (DAR) – contains the address for destination operands used during transfers into memory.
  - Transfer count register (TCR) – contains the number of bytes to be transferred by a channel.
  - Base address register (BAR) – contains the memory address for the start of the next request block (used during chained operations only).
  - Mode register (MR) – configures the channel for the transfer to be performed.
  - Device control register (DCR) – contains information about the device serviced by the channel.
  - Interrupt vector register (IVR) – contains the interrupt vector used when a channel's operation terminates.
  - Status and control register (SCR) – contains status information about the channel's operation.
- The global DMAC register is:
  - Mask register (MASKR) – contains information used to disable particular channel activity.

Address Arithmetic Unit (AAU) – calculates addresses of source and destination using information from per channel registers and data size and request generator.

Count Arithmetic Unit (CAU) – calculates number of bytes to be transferred using information from per channel registers and data size and request generator.

Packing Registers – used to pack bytes into larger operands when transferring from the peripheral bus to the system bus and to unpack large system bus operands to bytes when transferring to the peripheral bus.

Peripheral Bus – used to communicate with 8-bit I/O devices.

Register Organization

The DMAC has four independent channels that allow it to service four unrelated transfer requests simultaneously. Each channel has a set of registers that configures and controls its operation. In addition, there is one mask register that is shared among the four channels. The DMAC registers are mapped into the processor's address space. When the DMAC is in peripheral mode, registers within the DMAC and registers within devices connected to the peripheral bus are accessed by the CPU. This provides the CPU with an access path to the peripheral bus.

The DMAC requires a 2-Kbyte address space, which is partitioned into four sections (see Table 1). Sections are provided for the DMAC per channel registers, global registers, and internal data buffers, and for a programmed input/output path to device controllers.

Per Channel Control Registers

There are eight DMAC control registers allocated on a per channel basis, as shown on Figure 2. These registers can be accessed by the CPU and are addressed in section one of the DMAC address space (shown in Table 2). A description for each register follows.

Table 1. DMAC Address Space

Access Type	$\overline{CS}$	Address Space (ADDR02—ADDR10)								
		Section		Channel		Address				
		10	09	08	07	06	05	04	03	02
No operation	1	x	x	x	x	x	x	x	x	x
Access to the peripheral bus	0	0	0	0	0	x	x	x	x	x
	0	0	0	0	1	x	x	x	x	x
	0	0	0	1	0	x	x	x	x	x
	0	0	0	1	1	x	x	x	x	x
DMAC internal registers (per channel)	0	0	1	0	0	x	x	x	x	x
	0	0	1	0	1	x	x	x	x	x
	0	0	1	1	0	x	x	x	x	x
	0	0	1	1	1	x	x	x	x	x
DMAC internal registers (global)	0	1	0	x	x	x	x	x	x	x
DMAC internal data buffers (per channel)	0	1	1	0	0	x	x	x	x	x
	0	1	1	0	1	x	x	x	x	x
	0	1	1	1	0	x	x	x	x	x
	0	1	1	1	1	x	x	x	x	x

x = don't care.

31 16	15 0
	Status and control register (SCR)
	Mode register (MR)
	Device control register (DCR)
	Transfer count register (TCR)
	Interrupt vector register (IVR)
Source address register (SAR)	
Destination address register (DAR)	
Base address register (BAR)	

Figure 2. Programmer's Per Channel Control Register Set

Table 2. DMAC Register Address Map (Sections 1 and 2)

Address Space (ADDR00—ADDR10)											
Section		Channel		Address							
10	09	08	07	06	05	04	03	02	01	00	Register
0	1	x	x	x	0	0	0	0	x	x	Source address
0	1	x	x	x	0	0	0	1	x	x	Destination address
0	1	x	x	x	0	0	1	1	x	x	Base address
0	1	x	x	x	0	1	0	x	x	x	Transfer count
0	1	x	x	x	0	1	1	x	x	x	Interrupt vector
0	1	x	x	x	1	0	0	0	x	x	Status and control
0	1	x	x	x	1	0	0	1	x	x	Mode
0	1	x	x	x	1	0	1	0	x	x	Device control
1	0	x	x	x	0	1	0	0	x	x	Mask

x = don't care.

Source Address Register (SAR). This 32-bit register contains the address of source operands used in memory-to-peripheral and memory-to-memory transfers. It is incremented by 0, 1, 2, 4, 8, or 16, depending on the size of the operand just transferred and the contents of the mode register (specifies whether source addresses are incremented or held constant). The SAR must be loaded with a source address for each transfer unless the source address for the transfer is contiguous with the last address of the previous transfer.

Note: The source address of memory-to-peripheral transfers must be word aligned (i.e., addresses divisible by 4). Memory-to-peripheral transfers should not use request chaining if there is multiple channel activity.

Destination Address Register (DAR). This 32-bit register contains the address of destination operands used in peripheral-to-memory and memory-to-memory transfers. It is incremented by 0, 1, 2, 4, 8, or 16, depending on the size of the operand just transferred and the contents of the mode register that specifies whether destination addresses are incremented or held constant. The DAR must be loaded with a destination address for each new transfer unless the starting address is contiguous with the last address of the previous transfer.

Transfer Count Register (TCR). This 16-bit register initially contains the number of bytes to be transferred. This nonzero value is decremented each time an operand is transferred and should contain 0 after an operation has completed without error. Because this is a 16-bit register, the maximum data transfer that can be performed in a single request is approximately 64 Kbytes (i.e., 64K – 1).

Base Address Register (BAR). This 32-bit register contains the memory address for the start of the next request block. This register is only used for chained operations. The BAR is usually loaded at the beginning of a chained operation and again each time a new request block is read from memory.

Interrupt Vector Register (IVR). This 16-bit register contains an 8-bit interrupt vector that is placed on the system bus upon termination of an operation. Bits 8 through 15 of the IVR are unused. This register is used only when the mode register specifies that interrupts are enabled. The contents of this register are placed on the lower 8 bits of the system data bus after the CPU acknowledges an interrupt request from the DMAC with an interrupt acknowledge. When a channel terminates normally, bit 0 of the IVR is set before the vector is placed on the data bus. When a channel terminates because of an error condition, bit 0 is cleared before the vector is placed on the data bus. Figure 3 shows the IVR format.

15 8	7 0
Unused	Interrupt Vector

Figure 3. Interrupt Vector Register Format

Mode Register (MR). The mode register is used to configure a channel for the transfer to be performed. If the channel is shared for different transfer types, the MR must be updated when switching between types. Table 3 describes the MR format.

Device Control Register (DCR). The device control register contains information that describes the device being serviced by the channel. It also describes how the peripheral bus is to be used. The DCR is usually loaded only when the system is set up. Table 4 describes the format of the DCR.

Status and Control Register (SCR). The status and control register contains information about the channel's current status. The SCR is also used to issue commands to the channel. This register is used to begin a channel's operation and is referenced when a channel terminates operation. Table 5 shows the SCR format.

Table 3. Mode Register Format

Bit	15	14 13	12	11	10	9	8	7 5	4 3	2	1 0
Field	BR	TT	—	SAC	—	DAC	—	DS	—	IE	—

Bit	Field	Contents	Description
0—1	—	—	Unused.
2	IE	Interrupt enable	When set, the CPU is interrupted upon completion of the channel's transfer by the interrupt request signal, $\overline{\text{INTRQ}}$. When cleared, $\overline{\text{INTRQ}}$ is disabled.
3—4	—	—	Unused.
5—7	DS	Data size	This field selects the largest operand that can be transferred in a single bus cycle. It overrides any automatic data size generated by the DMAC for cases where the addressed location cannot handle certain data sizes. Memory fill operations use this field to select the size of the operand to be used from the memory fill data register (MFDR). This field is interpreted as: **Bit 7 6 5 — Operand Size** 1 1 1 — Byte 1 1 0 — Half-word 1 0 0 — Word 1 0 1 — Double-word[1] 0 x x — Quad-word[1,2] where x = don't care.
8	—	—	Unused.
9	DAC	Destination address control	If cleared, the destination address is incremented; if set, the destination address is held constant.
10	—	—	Unused.
11	SAC	Source address control	If cleared, the source address is incremented; if set, the source address is held constant.
12	—	—	Unused.
13—14	TT	Transfer type	This field identifies the type of transfer and is interpreted as: **Bit 14 13 — Transfer Type** 0 0 — Memory-to-memory 0 1 — Peripheral-to-memory 1 0 — Memory-to-peripheral 1 1 — Memory fill
15	BR	Burst mode	When set, the channel operates in burst mode. When cleared, the channel operates in cycle steal mode.

[1] Memory systems must support block accesses to use this data size. $\overline{\text{BLKACS}}$ must be asserted.

[2] If using chained mode peripheral-to-memory transfers, do not use this data size.

Table 4. Device Control Register

Bit	15 14	13	12 11	10	9 7	6 2	1 0
Field	CS	BL	—	S/$\overline{A}$	SF	PA	—

Bit	Field	Contents	Description
0—1	—	—	Unused.
2—6	PA	Peripheral address	This field contains the address of the peripheral device data port. The contents of this field are placed on the peripheral bus when the channel wants to access the peripheral device data port.
7—9	SF	Stretch field	This field specifies the number of wait states that are inserted before the DMAC latches/removes data from the peripheral bus during read/write cycles. This field is ignored if the S/$\overline{A}$ field (bit 10) is cleared. The number of wait states to be inserted is interpreted as: **Bit 9 8 7** — **Wait States** 0 0 0 — 0 0 0 1 — 1 0 1 0 — 2 0 1 1 — 4 1 0 0 — 8 1 0 1 — 16 1 1 0 — 32 1 1 1 — 64
10	S/$\overline{A}$	Synchronous/ asynchronous	This bit determines how transfers are timed on the peripheral bus. If this bit is set, the SF field is used to determine the number of wait states to be used during transfers. If this bit is cleared, the peripheral bus data transfer acknowledge signal, $\overline{PDTACK}$, is used to terminate the bus cycle. The $\overline{PDTACK}$ signal is generated by the peripheral device.
11—12	—	—	Unused.
13	BL	Burst length	This bit determines how many bytes are transferred for each peripheral bus device request. If cleared, only one byte can be transferred. If set, up to 64 Kbytes may be transferred per request.
14—15	CS	Chip select code	This field contains the chip select code for the device that this channel is servicing. The chip select code is placed on the peripheral bus during a data transfer and decoded to enable one of four devices on the peripheral bus.

Table 5. Status and Control Register

Bit	15 13	12	11	10	9	8	7 6	5 1	0
Field	—	CH	STR	HB	SA	ACT	—	ERR	NT

Bit	Field	Contents	Description
0	NT	Normal termination	This bit is set when a channel terminates operation without an error. The CPU checks this field to verify that the channel terminated normally. If this bit is not set, the ERR field (bits 1—5) must be examined.
1—5	ERR	Error code	This field is loaded with an error code when the DMAC detects that an error has occurred on the channel. This field is cleared when the channel start bit (bit 11) is set. The error codes are interpreted as: **Bit 5 4 3 2 1** — **Error Type** 0 0 0 0 0 — No error 0 0 0 0 1 — System bus error due to fault 0 0 0 1 0 — System bus error due to $\overline{CS}$ being asserted when DMAC is bus master 0 0 1 0 0 — Software abort 1 0 0 0 0 — Peripheral bus time out
6—7	—	—	Unused.
8	ACT	Channel active	This bit is set by the channel when the channel becomes active. When the transfer completes, the ACT bit is cleared. If the interrupt option is not selected in the mode register, this bit can be polled by the CPU to determine when a transfer has completed. A channel can be active even if its mask register bit or halt bit is set.
9	SA	Software abort	This bit is set by the CPU to terminate the current block of a peripheral-to-memory or memory-to-memory transfer, causing the DMA controller to proceed to the next block.
10	HB	Channel halt	This bit is set by the CPU to halt the operation of a channel. When the bit is set, the channel does not service requests on the system bus; however, requests on the peripheral bus are still honored. When the bit is cleared, the channel resumes operation.
11	STR	Channel start	This bit is set by the CPU to start a channel's operation. When the channel detects that this bit is set, it clears the STR bit, sets the ACT bit, and executes the transfer request.
12	CH	Chained requests	When this bit is set by the CPU, the channel chains requests, using the base address register as a pointer to the next request block in memory. When this bit is cleared, the channel does not use chaining.
13—15	—	—	Unused.

Global DMAC Register

The global DMAC register is shared by all four DMAC channels. Its address map is shown in Table 2.

Mask Register (MASKR). The mask register is used to specify which channels are to ignore and/or inhibit bus requests. The DMAC ignores bus requests for each channel (CH0—CH3) that has its mask bit set on the peripheral bus and inhibits requests on the system bus. Figure 4 shows the MASKR format. When TCR is zero and the MASKR is cleared, setting the start bit terminates the channel normally. However, with the TCR=0, if the MASKR and the start bit are both set, the channel goes active and clearing the MASKR does not make that channel inactive.

9 4	3	2	1	0
Reserved	CH3	CH2	CH1	CH0

Figure 4. Mask Register Format

Data Buffer Registers

Each channel has eight 32-bit data buffer registers. These buffers are accessible by the CPU and are addressed in section 3 of the address map (see Table 6). The lowest addressed data buffer register is also used as the memory fill data register for each channel.

Memory Fill Data Register (MFDR). This 32-bit register holds the data that is written to consecutive locations during memory fill operations. The register is written with the appropriate data during initialization of the memory fill operation and is used as the source for each transfer. The mode register determines the access type for the MFDR in the following ways:

If the DS field of the mode register is set for byte transfers, data for the first access is taken from bits 24—31 of the MFDR, data for the second from bits 16—23, data for the third from bits 8—15, and data for the fourth from bits 0—7. The next access uses data from bits 24—31 and the pattern repeats.

If the DS field of the MR is set for half-word transfers, data for the first access is taken from bits 16—31 of the MFDR and data for the second access from bits 0—15. The next access uses data from bits 16—31 and the pattern repeats.

If the DS field of the MR is set for word transfers, bits 0—31 of the MFDR are used for each access.

If the DS field of the MR is set for double-word transfers, the contents of the MFDR is used twice for each access.

If the DS field of the MR is set for quad-word transfers, the contents of the MFDR is used four times for each access.

Operation

The operation of a DMAC channel takes place in three phases: initialization, data transfer, and operation termination.

During initialization, the CPU sets up the operation to be performed. The starting address, word count, configuration, and control registers are loaded by the CPU and the channel operation is started.

During data transfer, the DMAC receives requests for data transfers from device controllers or generates internal requests and performs operand transfers. Address and transfer count registers are updated as operands are transferred. When the transfer count is decremented to zero, the channel enters the operation termination phase.

During operation termination, the channel updates status registers and generates an interrupt request to the CPU if configured to do so. If a chained operation is in progress, the channel determines if a request block is waiting to be processed. If a request block is waiting, the requested information is loaded and the channel returns to the data transfer phase.

Channel Initialization

To initialize a DMA channel, certain DMAC registers must be written with the appropriate

Table 6. DMAC Data Buffer Addresses

	Address Space (ADDR02—ADDR10)								
	Section		Channel		Address				
Access Type	10	09	08	07	06	05	04	03	02
DMAC internal	1	1	0	0	x	x	0	0	0*
data buffer	1	1	0	0	x	x	0	0	1
channel 1	1	1	0	0	x	x	0	1	0
	1	1	0	0	x	x	0	1	1
	1	1	0	0	x	x	1	0	0
	1	1	0	0	x	x	1	0	1
	1	1	0	0	x	x	1	1	0
	1	1	0	0	x	x	1	1	1
DMAC internal	1	1	0	1	x	x	0	0	0*
data buffer	1	1	0	1	x	x	0	0	1
channel 2	1	1	0	1	x	x	0	1	0
	1	1	0	1	x	x	0	1	1
	1	1	0	1	x	x	1	0	0
	1	1	0	1	x	x	1	0	1
	1	1	0	1	x	x	1	1	0
	1	1	0	1	x	x	1	1	1
DMAC internal	1	1	1	0	x	x	0	0	0*
data buffer	1	1	1	0	x	x	0	0	1
channel 3	1	1	1	0	x	x	0	1	0
	1	1	1	0	x	x	0	1	1
	1	1	1	0	x	x	1	0	0
	1	1	1	0	x	x	1	0	1
	1	1	1	0	x	x	1	1	0
	1	1	1	0	x	x	1	1	1
DMAC internal	1	1	1	1	x	x	0	0	0*
data buffer	1	1	1	1	x	x	0	0	1
channel 4	1	1	1	1	x	x	0	1	0
	1	1	1	1	x	x	0	1	1
	1	1	1	1	x	x	1	0	0
	1	1	1	1	x	x	1	0	1
	1	1	1	1	x	x	1	1	0
	1	1	1	1	x	x	1	1	1

x = don't care.

* Also used as MFDR for this channel.

information. The specific registers needed and the contents of these depends on the type of operation to be performed. Some registers need to be written only when the system is initialized. Others need to be written for each transfer the channel is to perform. For specific information on each register and when it needs to be updated see Register Organization.

If chained operation is selected during initialization, the CPU does not load the source address register, the destination address register, or the transfer count register. Instead, a request block must be written to memory. The request block consists of four 32-bit words: a source address, destination address, transfer count, and base address. The request block must begin on a word boundary. The base address register is written with the beginning address of the request block. During operation, the base address register is used to fetch the first request block from memory. The source address, destination address, and transfer count of the request block are used as in operations without chaining. When the transfer count is completed, the base address of the request block is used to fetch the next request block. If the base address is all 0s, the channel operation terminates. See Figure 5 for a representation of chaining.

When the chosen set of channel registers have been configured, the STR bit of the status and control register must be set. When the DMA senses that this bit is set, the STR bit is cleared and the ACT bit is set indicating that the channel is active. The channel enters the data transfer phase.

Data Transfer

Operands are read from and written to memory and peripherals during the data transfer phase.

Modes of Operation. A DMAC channel can be configured to work in one of two modes: burst or cycle steal. The distinction between the modes is the manner in which the system bus is requested.

The DMAC has an internal register that holds pending system bus requests for each of the four channels. When a channel wants access to the system bus, it sets a request pending bit in this internal register. The DMAC uses this register to determine when to activate its $\overline{\text{BUSRQ}}$ signal to the CPU. The DMAC asserts the $\overline{\text{BUSRQ}}$ signal whenever any channel has a request pending and the $\overline{\text{BRACK}}$ signal from the CPU is not active. Once $\overline{\text{BUSRQ}}$ has been asserted, it remains active as long as at least one channel has a request pending.

When a channel is operating in burst mode, its system bus request remains active as long as the channel is ready. Therefore, the $\overline{\text{BUSRQ}}$ signal remains asserted for the same period.

In cycle steal mode, a channel requests the bus only when it has data to be written or it is ready to read new data. When the channel's request is serviced, it performs one bus cycle. The DMAC negates the $\overline{\text{BUSRQ}}$ signal at the end of the bus cycle. Once $\overline{\text{BUSRQ}}$ is negated, it is not asserted again until a new request is posted internally and the $\overline{\text{BRACK}}$ signal from the CPU is negated.

In the cycle steal mode, DMAC holds the bus for only one cycle. Consequently, only the highest priority channel is served when several channels are operating in cycle steal mode.

When a cycle steal channel and a burst mode channel are operating simultaneously, the burst mode channel locks out the cycle steal channel, overriding the channel priority.

During operation, a burst mode channel can be changed to cycle steal mode if the $\overline{\text{IPEND}}$ signal is asserted (see Pin Descriptions).

Bus Requests. Peripheral bus requests are generated when a peripheral device has data to transmit or when it is available to receive data. Each of the four DMA channels has an associated peripheral bus request line ($\overline{\text{PBR0}}$—$\overline{\text{PBR3}}$) that the peripheral device uses to request the bus.

For peripheral-to-memory transfers, the peripheral device requests the bus from the DMAC each time the device has data ready to send. The DMA channel reads the data and stores it in the internal data buffer (provided space is available). When 16 bytes have been stored in the buffer, the channel requests the system bus and transfers the data to memory.

For memory-to-peripheral transfers, the DMAC fills its internal data buffer by reading the system bus. Each time the peripheral device issues a peripheral bus request, the DMAC writes another byte to the device (provided the data buffer is not empty).

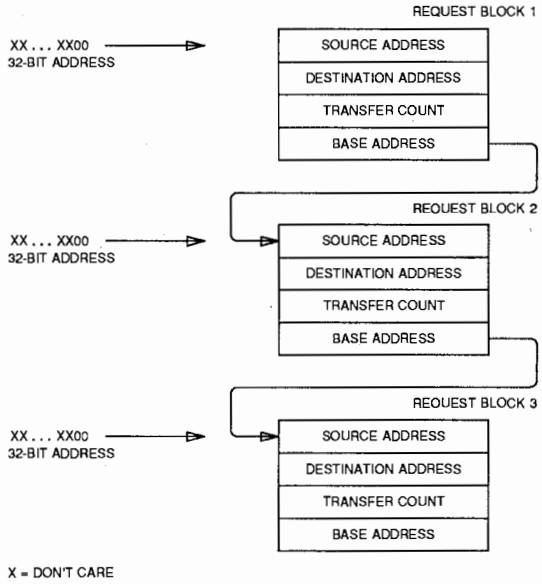


Figure 5. Linked List Chaining

The DMAC channel can be configured to respond to the peripheral bus requests in two ways: one byte transferred per request or up to 64 Kbytes transferred per request. The burst length (BL) bit of the device control register controls this operation. When the BL bit is cleared, the channel is configured for byte-at-a-time operation; each high-to-low transition of the channel's PBR pin signals that another byte is to be transferred. If the BL bit is set, each high-to-low transition of the channel's PBR pin signals the start of another burst transfer. This transfer is terminated by the negation of the PBR signal, assertion of the $\overline{\text{PDONE}}$ signal, or when the transfer count register reaches zero.

Note: For burst peripheral-to-memory transfers, if the transfer does not end on a word boundary, a minimum of one wait-state on the peripheral side is required.

The DMAC is capable of servicing both the peripheral and system buses at the same time. Transfers on each bus do not have to be for the same channel. The channel number determines a channel's priority when two or more channels request a bus at the same time (e.g., channel 0 has the highest priority and channel 3 has the lowest). The same scheme is used to prioritize requests for the system bus and the peripheral bus. Multi-channel, memory-to-peripheral transfers should not use chaining.

Operand Transfers. A single byte is the normal operand size on the peripheral bus. However, the peripheral device may operate in burst mode when an unlimited number of bytes are transferred per bus request. Burst mode is selected by setting the BL bit of the channel's device control register. The length of a burst transfer is determined by the assertion and negation of the peripheral bus request signals ($\overline{\text{PBR0}}$—$\overline{\text{PBR3}}$).

On the system bus, operands may be a single byte, halfword, word, double-word, or quad-word. The data size signals (DSIZE0—DSIZE2) define the operand size. For double- and quad-word transfers, the DMAC places a single address on the system bus for the duration of the transfer. Data and handshake signals are exchanged with memory. The source or destination address register and transfer count register are updated by the end of the bus cycle.

Data Packing and Unpacking. The DMAC provides packing and unpacking of data when transferring to and from the peripheral bus. Each channel has 32 bytes of buffer space, which are used to pack bytes of data coming from the peripheral bus into larger operands before writing them to memory. Under normal circumstances, 16 bytes of data are read from the peripheral bus and stored in the buffer.

A system bus request is then posted and any data read from the peripheral bus while waiting for the system bus request to be granted is stored in the remaining 16 bytes of buffer memory. When the request is granted, a quad-word transfer can be performed, moving 16 bytes from the buffer to the system bus.

For memory to peripheral transfers, larger system bus operands are unpacked into bytes that are stored in the buffer and transferred to the peripheral bus.

Address Alignment for Memory-to-Memory Transfers. The most efficient operand transfers take place in the DMAC when the source address and the destination address are aligned on quad-word boundaries. When this alignment is present, the channel can use a quad-word read cycle to fetch the operand. The operand is stored in the internal buffer and a quad-word write cycle can be used to return the operand to memory. The DMAC can still take advantage of the quad-word read and write cycles when the addresses are not aligned if they differ by a multiple of four.

When the difference between the source address and the destination address is not divisible by four but is even, read cycles are a quad-word and write cycles are a halfword. When the difference between the source address and the destination address is odd, read cycles are a quad-word and write cycles are a byte.

With these options, the DMAC provides total flexibility in address alignment for memory-to-memory transfers. With careful coding to insure proper alignment, operations can take advantage of the quad-word bus cycles.

Source and Destination Address Control. Source and destination addresses placed on the system bus can be independently configured so that the addresses are incremented or held constant following an operand transfer. The source address control (SAC) and destination address control (DAC) bits of the mode register are used to select these options.

Holding an address constant permits that address to be used as the source or destination location for all operands in a transfer. This supports 16- or 32-bit peripheral devices on the system bus. Since the data port of a device is used for every access to the device, the corresponding address need not be changed.

Certain requirements must be met when the hold feature is used for source addresses. The data size field of the mode register must be set to halfword or greater and the source address must be on a word boundary.

Peripheral Bus Transfers. Each channel can be configured for synchronous or asynchronous peripheral bus cycles. When the S/$\overline{A}$ bit of the channel's device control register is set, the channel is configured for synchronous operation. The DMAC drives the peripheral bus address (PERADD0—PERADD4), chip select code (PCS0—PCS3), strobe ($\overline{PDS}$), and peripheral bus data (PERD0—PERD7) for a write cycle. The DMAC waits for the number of cycles as specified in the stretch field of the DCR and then terminates the transaction.

When the S/$\overline{A}$ bit of the channel's DCR is cleared, the channel is configured for asynchronous operation. Operation during asynchronous cycles is the same as synchronous cycles except that the transaction does not terminate until the $\overline{PDTACK}$ signal is asserted. The DMAC "times out" an asynchronous peripheral bus cycle if the $\overline{PDTACK}$ is not asserted within 64 cycles. When a bus cycle is timed out, the DMAC negates $\overline{PDS}$ and $\overline{PCS}$ ends the bus cycle and the operation of the channel responsible for the bus cycle is terminated. The normal termination bit of the channel's status and control register is cleared to indicate an error has occurred and the appropriate error code for a peripheral bus time-out is stored in the error field.

A channel can also be configured to transfer bursts of data for each peripheral bus request. When the burst length bit of the DCR is set, the channel sends or receives data until the request is negated or the transfer terminates. In synchronous or asynchronous mode, the DMAC

generates a single address for the duration of the burst, transfers each data byte, and then either waits for the PDTACK signal to be asserted or generates wait states internally. Once the DMAC has an acknowledgement (PDTACK asserted), the DMAC negates PDS (indicates to the peripheral device to remove the data and DTACK from the bus). After one half cycle, PDS can be asserted and the process repeats.

Termination of Channel Operations

When the DMAC detects that a channel's transfer count register is 0 and the internal buffers are emptied, the channel's operation terminates normally. The channel's status and control register (SCR) is checked to see if the CH bit is set. If chaining is not specified, the DMAC updates the SCR by clearing the ACT bit to indicate that the channel is no longer active, and by setting the NT bit to indicate normal termination. If the IE bit of the mode register is set, an interrupt request is generated.

A channel operation can also terminate as a result of the PDONE signal being asserted during a peripheral bus read. The bytes remaining in the data buffer are written to memory and the channel terminates as usual. An external transfer count register is required to determine the exact number of bytes transferred.

Peripheral transfers in which PDONE is asserted require a minimum of one wait state. For synchronous transfers, this implies that the stretch field must be at least one. For asynchronous transfers, PDTACK must remain high so that at least one wait state is asserted.

When PDONE is asserted during a peripheral data transfer on the third byte of a data word in the DMAC's data RAM, an extra byte of data is written to the system memory at the end of the data transfer. When PDONE is asserted during a peripheral data transfer on the fourth byte of a data word in the DMAC's data RAM, the SCR's ACT bit is not cleared and 0—14 bytes of data are not written into the system memory. The amount of data depends on DSIZE.

In order for PDONE to be recognized, it must be asserted in the cycle prior to PDTACK assertion.

The PDONE signal can be used to terminate channels performing PTM transfers, using either a "byte-at-a-time" or "burst" mode peripheral bus transfers. When byte-at-a-time transfers are used for a channel to be terminated with PDONE, only one active channel can be configured for a PTM operation. Other types of transfers, excluding peripheral-to-memory and memory-to-peripheral, execute on the other three channels. For channels using burst mode peripheral data transfers, any combination of channel configurations is valid. Note that peripheral bus bursts are normally terminated by the negation of the peripheral bus request (PBR) input, but that PDONE will also halt a burst when a channel operation is terminated in this manner.

When the CPU responds to an interrupt from the DMAC or detects that the ACT bit is cleared, the status and control register should be read to determine if the operation terminated normally (NT bit). If an error occurred, the NT bit is cleared and the error field should be read to determine the type of error that occurred.

Interrupts

The DMAC can be configured to interrupt the CPU upon completion of a channel's operation by setting the IE bit of the channel's mode register. The DMAC supplies an interrupt vector and expects an interrupt acknowledge cycle when interrupts are enabled.

Upon termination of a channel's operation, the DMAC checks the IE bit. If it is set, an interrupt request is issued by asserting the INTRQ signal. When an interrupt acknowledge is received (INTACK asserted), the DMAC checks the NT bit of the status and control register. If the NT bit is set, bit 0 of the interrupt vector register (IVR) is set, providing the normal interrupt vector. If the NT bit is cleared, bit 0 of the IVR is cleared, providing the error interrupt vector.

Each DMAC channel has an interrupt pending bit that is set when the channel wishes to interrupt the CPU. If more than one channel has its interrupt enable bit set, the DMAC uses a priority algorithm to determine which vector to supply when an interrupt acknowledge is received. The DMAC INTRQ signal remains asserted until all pending interrupts are acknowledged. A fault, retry, or relinquish and retry request must not occur while the CPU is acknowledging a DMAC interrupt request to prevent the DMAC's internal interrupt request for each channel from being cleared before the CPU reads the interrupt vector. If INTACK is

asserted for the DMAC after the internal interrupt register has been cleared, the DMAC returns the normal interrupt vector for channel 3.

Bus Exceptions

The DMAC has four types of bus exceptions: fault, retry, relinquish and retry, and reset. Initiation of any of these bus exceptions is caused by the assertion of its associated signal. The assertion of one of the bus exception signals forces the DMAC to terminate the current bus cycle. The bus exception must arrive before or at the same time as $\overline{\text{DTACK}}$ or $\overline{\text{SRDY}}$ for the cycle to be recognized as terminating abnormally. While a bus exception is present, the DMAC will not issue any system bus cycles, but will monitor peripheral bus requests. When the bus exception is removed, the DMAC resumes system bus transactions.

Fault. The assertion of $\overline{\text{FAULT}}$ during a bus cycle in which the DMAC is the bus master causes the cycle to be preempted. At this time the operation in progress is aborted, and the error field in the status register is updated to indicate that the operation is terminated and that the error type was a system bus fault. If the DMAC is configured to issue an interrupt upon operation completion, the NT bit of the status register is not set, causing the error interrupt vector to be issued (i.e., abnormal termination). The interrupt routine then examines the error field of the status register to determine the type of error that occurred. The CPU also has access to other DMAC channel registers in order to locate the source of the problem and to reinitialize the operation at the point where the error occurred.

Retry. The assertion of $\overline{\text{RETRY}}$ causes the DMAC to terminate the present bus cycle and retry it when the $\overline{\text{RETRY}}$ signal is negated. During the assertion of $\overline{\text{RETRY}}$, the address and data buses are three-stated. The DMAC is guaranteed to get control of the bus on the negation of $\overline{\text{RETRY}}$. If the retry request occurred during a double- or quad-word cycle, the DMAC retrys the entire cycle, not just the remaining portion.

Relinquish and Retry. The assertion of $\overline{\text{RRREQ}}$ causes the DMAC to three-state all of its bus master control signals, preempting the bus cycle in progress (output states are shown in Table 7). Following the negation of $\overline{\text{RRREQ}}$, the bus cycle is retried. If this exception occurs during a double- or quad-word cycle, the DMAC retrys the entire cycle, not just the remaining portion.

Reset. The reset exception causes the DMAC to return to the reset state in which:

- The mask register is set to block all requests (i.e., 0xFF)
- The STR, ERR, and ACT bits for all SCR registers are cleared
- The NT bits for all SCR registers are set
- All other register values are undefined
- Outputs are set as shown in Table 7.

Table 7. Output States

Signal	DMAC Not Bus Master	$\overline{\text{RRRACK}}$ Asserted	After Reset
ADDR00—ADDR31	Z	Z	Z
DATA00—DATA31	Z	Z	Z
$\overline{\text{AS}}$	Z1	Z1	Z1
$\overline{\text{DS}}$	Z1	Z1	Z1
$\overline{\text{CYCLEI}}$	Z1	Z1	Z1
$\overline{\text{DRDY}}$	Z1	Z1	Z1
DSIZE0—DSIZE2	Z	Z	Z
R/$\overline{\text{W}}$	Z	Z	Z
$\overline{\text{RRRACK}}$	Z*	logic 0	Z*
$\overline{\text{DTACK}}$	Z*	Z*	Z*
$\overline{\text{INTRQ}}$	X	X	logic 1
$\overline{\text{BUSRQ}}$	X	logic 0	Z*
$\overline{\text{FAULT}}$	Z*	0Z*	logic 1

Notes:
Z = High-impedance.
Z1 = Actively pulled to logic 1; then made high impedance (held at logic 1 with passive holding resistor).
Z* = High impedance, open drain.
0Z* = 0 if $\overline{\text{CS}}$ = 0, otherwise Z*.
X = Don't care

Interfacing

The DMAC provides a convenient interface with both the system bus and the I/O device controllers. It is designed to interface to the *WE* 32100 Microprocessor with a minimal amount of external logic.

Figure 6 shows a basic configuration with the DMAC and the microprocessor. The only external logic required in this configuration is a decoder for chip select and an interrupt controller.

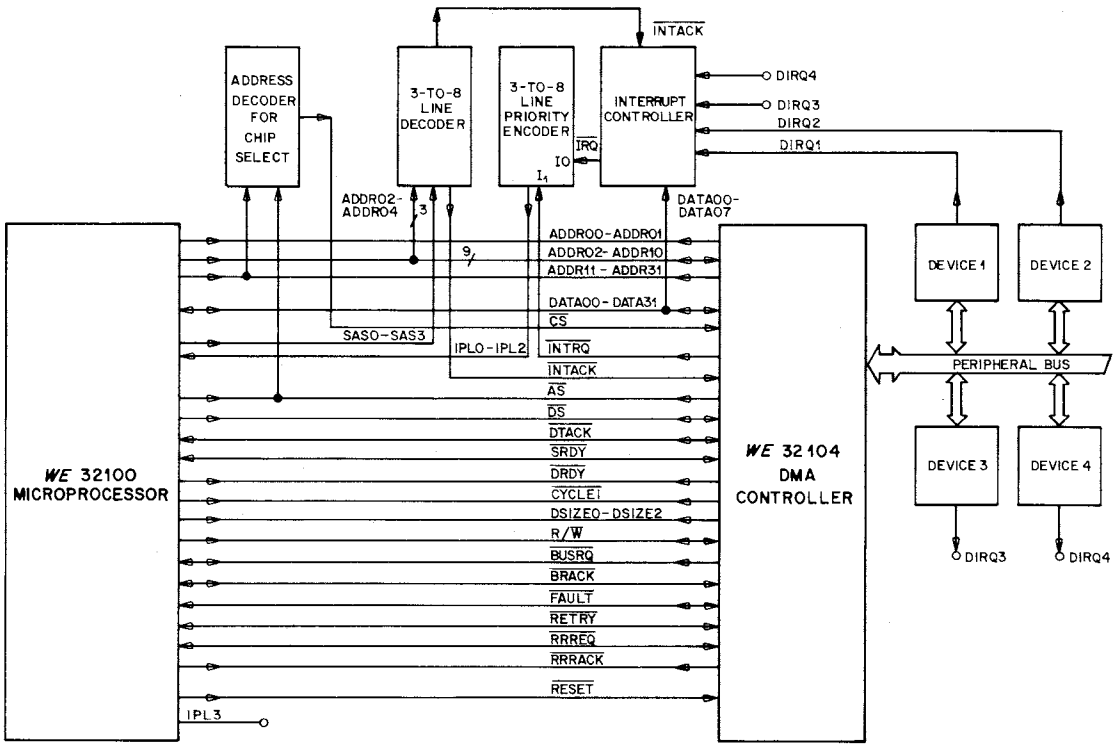


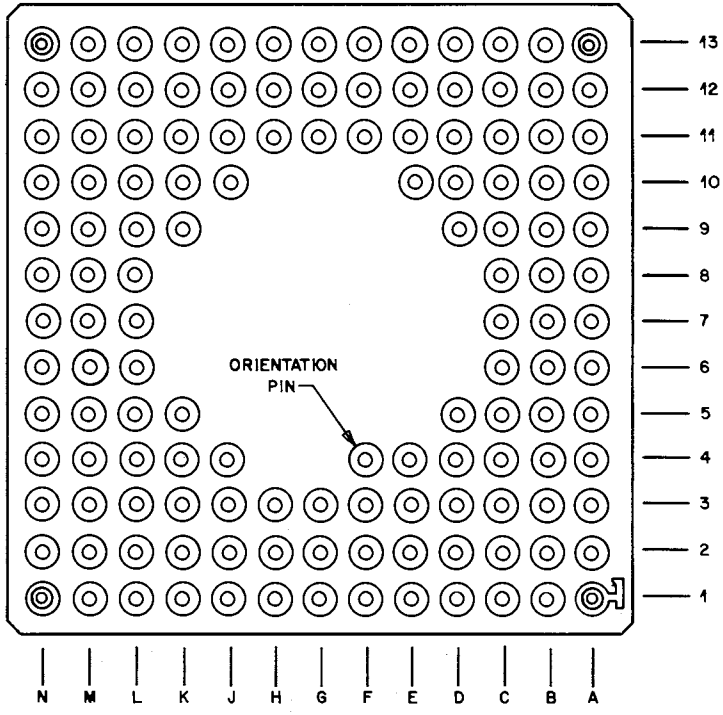
Figure 6. DMAC – Microprocessor Configuration

Pin Descriptions

The *WE* 32104 DMA Controller is available in a 133-pin square, ceramic PGA package. Figure 7 and Tables 8—16 describe the pin assignments.

The term *asserted* in the pin function descriptions means that the signal is driven to its active state either by the DMA controller (outputs) or an external device (inputs). The term *negated* means that the signal is driven to its inactive state. A bar over a signal name (e.g., $\overline{AS}$) indicates that the signal is active low, logic 0. The 0 bit is the least significant bit for signals that occupy two or more pins (e.g., $\overline{PBR0}$—$\overline{PBR3}$). In the signal type column, I indicates an input, O an output, and I/O a bidirectional signal.

Table 8 list signals in the numerical order of the pins for the 133-pin square PGA package. Tables 9 through 16 describe signals by functional group.



Bottom View

Figure 7. WE® 32104 DMA Controller 133-Pin Square, Ceramic PGA Package

Numerical Order

Table 8. Pin Descriptions – 133-Pin Ceramic PGA Package

Pin	Name	Type	Description
A1	$\overline{\text{PBR1}}$	I	Peripheral bus request 1
A2	$\overline{\text{PBR3}}$	I	Peripheral bus request 3
A3	$\overline{\text{BLKACS}}$	I	Block access
A4	DSIZE0	O	Data size 0
A5	ADDR31	O	Address 31
A6	DATA30	I/O	Data 30
A7	DATA29	I/O	Data 29
A8	DATA27	I/O	Data 27
A9	DATA26	I/O	Data 26
A10	ADDR26	O	Address 26
A11	ADDR24	O	Address 24
A12	ADDR23	O	Address 23
A13	DATA21	I/O	Data 21
B1	PERADD4	O	Peripheral address bus 4
B2	$\overline{\text{PBR0}}$	I	Peripheral bus request 0
B3	PR/$\overline{\text{W}}$	O	Peripheral bus read/write
B4	DATA31	I/O	Data 31
B5	DSIZE2	O	Data size 2
B6	ADDR29	O	Address 29
B7	DATA28	I/O	Data 28
B8	ADDR28	O	Address 28
B9	ADDR25	O	Address 25
B10	ADDR27	O	Address 27
B11	DATA22	I/O	Data 22
B12	DATA25	I/O	Data 25
B13	ADDR21	O	Address 21
C1	PERADD1	O	Peripheral address bus 1
C2	PERADD0	O	Peripheral address bus 0
C3	$\overline{\text{INTRQ}}$	O	Interrupt request
C4	DSIZE1	O	Data size 1
C5	$\overline{\text{PDS}}$	O	Peripheral data bus strobe
C6	ADDR30	O	Address 30
C7	+5V	—	Power
C8	GND	—	Ground
C9	+5V	—	Power
C10	DATA24	I/O	Data 24
C11	DATA23	I/O	Data 23
C12	ADDR22	O	Address 22
C13	DATA19	I/O	Data 19
D1	PERD7	I/O	Peripheral data bus 7
D2	PERADD3	O	Peripheral address bus 3
D3	$\overline{\text{PBR2}}$	I	Peripheral bus request 2
D4	+5V	—	Power
D5	$\overline{\text{BUSRQ}}$	O	Bus request

Table 8. Pin Descriptions – 133-Pin Ceramic PGA Package (Continued)

Pin	Name	Type	Description
D9	GND	—	Ground
D10	ADDR20	O	Address 20
D11	ADDR19	O	Address 19
D12	ADDR18	O	Address 18
D13	DATA18	I/O	Data 18
E1	PERD6	I/O	Peripheral data bus 6
E2	PERD5	I/O	Peripheral data bus 5
E3	PERADD2	O	Peripheral address bus 2
E4	$\overline{\text{PDTACK}}$	I	Peripheral bus data transfer acknowledge 2
E10	DATA20	I/O	Data 20
E11	+5V	—	Power
E12	DATA17	I/O	Data 17
E13	ADDR17	O	Address 17
F1	PERD4	I/O	Peripheral data bus 4
F2	PERD3	I/O	Peripheral data bus 3
F3	GND	—	Ground
F4	—	—	Device socket orientation pin
F11	GND	—	Ground
F12	DATA16	I/O	Data 16
F13	ADDR16	O	Address 16
G1	PERD2	I/O	Peripheral data bus 2
G2	PERD0	I/O	Peripheral data bus 0
G3	+5V	—	Power
G11	DATA13	I/O	Data 13
G12	DATA15	I/O	Data 15
G13	ADDR15	O	Address 15
H1	$\overline{\text{PDONE}}$	I	Peripheral bus done
H2	PERD1	I/O	Peripheral data bus 1
H3	GND	—	Ground
H11	GND	—	Ground
H12	ADDR12	O	Address 12
H13	DATA14	I/O	Data 14
J1	$\overline{\text{HIGHZ}}$	I	High impedance
J2	$\overline{\text{DRDY}}$	O	Data ready
J3	+5V	—	Power
J4	$\overline{\text{AS}}$	O	Address strobe
J10	DATA08	I/O	Data 08
J11	+5V	—	Power
J12	DATA12	I/O	Data 12
J13	ADDR14	O	Address 14
K1	$\overline{\text{RRREQ}}$	I	Relinquish and retry request
K2	$\overline{\text{RETRY}}$	I	Retry
K3	R/$\overline{\text{W}}$	I/O	Read/write
K4	$\overline{\text{PCS0}}$	O	Peripheral chip select 0
K5	$\overline{\text{PCS1}}$	O	Peripheral chip select 1

Table 8. Pin Descriptions – 133-Pin Ceramic PGA Package (Continued)

Pin	Name	Type	Description
K9	DATA07	I/O	Data 07
K10	GND	—	Ground
K11	DATA09	I/O	Data 09
K12	ADDR11	O	Address 11
K13	ADDR13	O	Address 13
L1	$\overline{DS}$	I/O	Data strobe
L2	$\overline{CYCLEI}$	O	Cycle initiate
L3	$\overline{PCS3}$	O	Peripheral chip select 3
L4	$\overline{RESET}$	I	Reset
L5	+5V	—	Power
L6	GND	—	Ground
L7	+5V	—	Power
L8	GND	—	Ground
L9	DATA04	I/O	Data 04
L10	DATA05	I/O	Data 05
L11	DATA06	I/O	Data 06
L12	ADDR10	I/O	Address 10
L13	DATA11	I/O	Data 11
M1	$\overline{FAULT}$	I/O	Fault
M2	$\overline{SRDY}$	I	Synchronous ready
M3	$\overline{INTACK}$	I	Interrupt acknowledge
M4	DATA00	I/O	Data 00
M5	$\overline{IPEND}$	I	Interrupt pending
M6	CLK34	I	Input clock 34
M7	CLK23	I	Input clock 23
M8	DATA01	I/O	Data 01
M9	ADDR04	I/O	Address 04
M10	ADDR03	I/O	Address 03
M11	ADDR07	I/O	Address 07
M12	ADDR08	I/O	Address 08
M13	DATA10	I/O	Data 10
N1	$\overline{DTACK}$	I/O	Data transfer acknowledge
N2	$\overline{RRRACK}$	O	Relinquish and retry acknowledge
N3	$\overline{PCS2}$	O	Peripheral chip select 2
N4	$\overline{CS}$	I	Chip select
N5	$\overline{BRACK}$	I	Bus request acknowledge
N6	ADDR00	O	Address 00
N7	ADDR01	O	Address 01
N8	ADDR02	I/O	Address 02
N9	DATA02	I/O	Data 02
N10	DATA03	I/O	Data 03
N11	ADDR05	I/O	Address 05
N12	ADDR06	I/O	Address 06
N13	ADDR09	I/O	Address 09

Functional Groups

Table 9. Address and Data

Name	Pin(s)	Type	Function
ADDR00, ADDR01, ADDR11—ADDR31	N6, N7, K12, H12, K13, J13, G13, F13, E13, D12, D11, D10, B13, C12, A12, A11, B9, A10, B10, B8, B6, C6, A5	O	**Address.** Along with ADDR02—ADDR10, are used to convey 32-bit addresses while the DMAC is the bus master. These signals are outputs when the DMAC is bus master and are 3-stated otherwise.
ADDR02—ADDR10	N8, M10, M9, N11, N12, M11, M12, N13, L12	I/O	**Address.** These signals are inputs when the DMAC is accessed in peripheral mode, translated as 9-bit addresses. When the DMAC is the bus master, they are outputs and are used with the remainder of the address bus to convey a 32-bit address.
DATA00—DATA31	M4, M8, N9, N10, L9, L10, L11, K9, J10, K11, M13, L13, J12, G11, H13, G12, F12, E12, D13, C13, E10, A13, B11, C11, C10, B12, A9, A8, B7, A7, A6, B4	I/O	**Data.** These signals provide a bidirectional bus to transmit data to and from the controller.

Table 10. Interface and Control Signals

Name	Pin	Type	Function
$\overline{AS}$	J4	O	**Address Strobe.** Assertion (logic 0) indicates a valid address on the address bus; negation (logic 1) indicates termination of a bus cycle.
$\overline{BLKACS}$	A3	I	**Block Access.** Asserted when the memory system is capable of performing the operand transfer specified by the DSIZE0—DSIZE2 output signals. If not asserted, the memory system addressed cannot perform the transfer requested, and only the first word is transferred.
$\overline{CS}$	N4	I	**Chip Select.** Used to select the DMAC for peripheral mode access to the DMAC internal registers, or to the device registers on the peripheral bus. When asserted the DMAC is in peripheral mode and responds as a slave to read and write requests. If $\overline{CS}$ becomes valid during a bus cycle in which the DMAC is the bus master, the bus cycle is halted and the DMAC records a system bus error in the channel's error register.

Table 10. Interface and Control Signals (Continued)

Name	Pin(s)	Type	Function
$\overline{\text{CYCLEI}}$	L2	O	**Cycle Initiate.** Asserted at the beginning of a DMAC-owned bus cycle and negated at the end of state 3. When the DMAC does not own the bus, this signal is 3-stated.
$\overline{\text{DRDY}}$	J2	O	**Data Ready.** Assertion indicates to slave devices that the DMAC has not detected any bus exceptions (i.e., $\overline{\text{FAULT}}$, $\overline{\text{RETRY}}$, $\overline{\text{RRREQ}}$). The trailing edge of this signal marks the end of a bus cycle that has no bus exceptions.
$\overline{\text{DS}}$	L1	I/O	**Data Strobe.** Is an output when the DMAC is the bus master. During read operations, it indicates to slave devices that data may be placed on the data bus. During write operations, it indicates to slave devices that the DMAC has placed valid data on the data bus. In addition, $\overline{\text{DS}}$ is used during interrupt acknowledge cycles to clear an internal interrupt pending register. When DMAC is the bus slave, $\overline{\text{DS}}$ becomes an input. During read cycles it signals to the DMAC that data may be placed on the data bus. During write cycles, it indicates to the DMAC that valid data is on the data bus.
DSIZE0—DSIZE2	A4, C4 B5	O	**Data Size.** Indicates the size of the transaction when the DMAC is the bus master. The pins are encoded as follows: **DSIZE 2 1 0** — **Size of Transaction** 0 0 0 — Unassigned 0 0 1 — Unassigned 0 1 0 — Unassigned 0 1 1 — Quad-word 1 0 0 — Word 1 0 1 — Double-word 1 1 0 — Halfword 1 1 1 — Byte
$\overline{\text{DTACK}}$	N1	I/O	**Data Transfer Acknowledge.** An input when DMAC is bus master. Assertion during a read operation indicates that the slave has placed data on the data bus and that the DMAC will latch it. Assertion during a write operation indicates that the slave device has received the data and the DMAC can now negate $\overline{\text{AS}}$ to terminate the bus cycle. An output when the DMAC is a bus slave. Assertion during read operations indicates that the data requested has been put on the data bus. Assertion during write operations indicates that the DMAC has latched the data present on the data bus. For both read and write operations, the DMAC must negate $\overline{\text{DTACK}}$ when $\overline{\text{DS}}$ is negated by the current bus master.

Table 10. Interface and Control Signals (Continued)

Name	Pin	Type	Function
R/$\overline{W}$	K3	I/O	**Read/Write.** This signal is an output when the DMAC is the bus master and an input when the DMAC is a bus slave. Indicates if the bus cycle is a read or a write. Logic 1 indicates read; logic 0 indicates write.
$\overline{SRDY}$	M2	I	**Synchronous Ready.** Signals the DMAC to begin terminating the current read or write access. Used only when the DMAC is bus master. **Note:** Since the signal is synchronous, it is used only when the DMAC shares a common clock with the slave device.

Table 11. Interrupt Signals

Name	Pin	Type	Function
$\overline{INTACK}$	M3	I	**Interrupt Acknowledge.** When the DMAC receives this signal from an external source and $\overline{DS}$ has been asserted, it places the interrupt vector on the data bus.
$\overline{INTRQ}$	C3	O	**Interrupt Request.** Asserted when the DMAC wants to interrupt the CPU. It is an input to an external interrupt controller that supplies the DMAC's interrupt priority level to the CPU.
$\overline{IPEND}$	M5	I	**Interrupt Pending.** When asserted, all burst mode transfers change to cycle steal transfers. This allows the CPU control of the bus for handling interrupts.

Table 12. Bus Arbitration Signals

Name	Pin	Type	Function
$\overline{BRACK}$	N5	I	**Bus Request Acknowledge.** Informs the DMAC that the system bus is available. It is asserted by the bus arbiter (CPU) after the appropriate bus signals have been 3-stated.
$\overline{BUSRQ}$	D5	O	**Bus Request.** Asserted when the DMAC wants to use the system bus, and remains asserted until the end of the last DMAC bus cycle. The DMAC must wait until $\overline{BRACK}$ is negated before reasserting $\overline{BUSRQ}$.

Table 13. Bus Exception Pins

Name	Pin	Type	Function
$\overline{\text{FAULT}}$	M1	I/O	**Fault.** When the DMAC is bus master, this signal notifies the DMAC that the current transfer cannot be completed and must be aborted. The DMAC terminates the channel operation in progress when the fault occurred and an internal error condition is set. Since this is an asynchronous signal, it is double-latched. As an output, $\overline{\text{FAULT}}$ is asserted when $\overline{\text{CS}}$ is asserted while $\overline{\text{RRRACK}}$ is asserted. This indicates that the DMAC is being selected while it is servicing a relinquish and retry request. Once the $\overline{\text{CS}}$ is negated, the DMAC negates $\overline{\text{FAULT}}$ and continues to wait for the negation of $\overline{\text{RRREQ}}$.
$\overline{\text{RESET}}$	L4	I	**Reset.** An asynchronous signal used to reset the DMAC. If the DMAC is bus master when the reset signal is received, it relinquishes the bus. Assertion of this signal puts the DMAC in its initialized state (See Bus Exceptions).
$\overline{\text{RETRY}}$	K2	I	**Retry.** If asserted while the DMAC is bus master, this signal causes the current bus cycle to be terminated. When the bus cycle is terminated, $\overline{\text{AS}}$ and $\overline{\text{DS}}$ are negated. When $\overline{\text{RETRY}}$ is negated, the bus cycle will be tried again. Only the address and data buses are 3-stated.
$\overline{\text{RRRACK}}$	N2	O	**Relinquish and Retry Acknowledge.** Asserted in response to a relinquish and retry exception when the DMAC has 3-stated its bus pins. Negated upon retry of the bus cycle terminated by the relinquish and retry bus exception.
$\overline{\text{RRREQ}}$	K1	I	**Relinquish and Retry Request.** Causes the DMAC to 3-state all bus master signals. When negated, the DMAC retries the previous bus cycle. DMAC pins 3-stated are $\overline{\text{AS}}$, $\overline{\text{DS}}$, R/$\overline{\text{W}}$, ADDR00—ADDR31, DATA00—DATA31, $\overline{\text{DRDY}}$ and DSIZE0—DSIZE2.

Table 14. Peripheral Bus Signals

Name	Pin(s)	Type	Function
$\overline{\text{PBR0}}$—$\overline{\text{PBR3}}$	B2, A1, D3, A2	I	**Peripheral Bus Requests.** Asynchronous inputs used by the device controllers to issue bus requests to the DMAC. Each bus request line is associated with a DMA channel (e.g., $\overline{\text{PBR0}}$ with channel 0), and an individual device may own one or more request lines. Assertion of this signal indicates that a bus request has been made. The request may be for a single byte or for a series of bytes, depending on the BL bit in the device control register.
$\overline{\text{PCS0}}$—$\overline{\text{PCS3}}$	K4, K5, N3, L3	O	**Peripheral Chip Selects.** When asserted, selects the peripheral device that will be the slave during the peripheral bus cycle. The device is not able to decode the $\overline{\text{PERADD}}$ lines until after the appropriate $\overline{\text{PCS}}$ signal is asserted. When negated, the peripheral bus cycle is terminated and the slave device is disabled.

Table 14. Peripheral Bus Signals (Continued)

Name	Pin(s)	Type	Function
$\overline{PDS}$	C5	O	**Peripheral Bus Data Strobe.** Asserted during a read operation on the peripheral bus when the DMAC is ready to accept data. Negated after the DMAC has received $\overline{PDTACK}$, or when the synchronous wait state timer has expired. For write operations, it is asserted when the DMAC has placed valid data on the data bus, and negated when the slave device issues $\overline{PDTACK}$ or when the synchronous timer expires.
$\overline{PDTACK}$	E4	I	**Peripheral Bus Data Transfer Acknowledge.** Asynchronous signal that notifies the DMAC to begin terminating a read or write access on the peripheral bus. Used by devices that are capable of issuing a data transfer acknowledge (those not capable use the synchronous timer function).
PERADD0—PERADD4	C2, C1, E3, D2, B1	O	**Peripheral Address Bus.** Used to select which register to access in the chip-selected device.
PERD0—PERD7	G2, H2, G1, F2, F1, E2, E1, D1	I/O	**Peripheral Data Bus.** Eight-bit bus used to transfer data between the DMAC and the 8-bit peripheral devices it communicates with.
PR/$\overline{W}$	B3	O	**Peripheral Bus Read/Write.** Determines if the current peripheral bus cycle will be a read or a write. logic 1 indicates a read, logic 0 indicates a write.
$\overline{PDONE}$	H1	I	**Peripheral Bus Done.** Indicates when the last byte of a transfer from peripheral-to-memory has been read. This signal is sampled in the cycle prior to $\overline{PDTACK}$. If $\overline{PDONE}$ is asserted, it is assumed that the last byte is being read for the channel currently utilizing the peripheral bus. This pin must not be asserted in a memory-to-peripheral transfer.

Table 15. Clocks

Name	Pin	Type	Function
CLK23	M7	I	**Clock 23.** A 1X clock input to the DMAC. Leads CLK34 by 90°.
CLK34	M6	I	**Clock 34.** A 1X clock input to the DMAC. Lags CLK23 by 90°.

Table 16. Miscellaneous Signals

Name	Pin	Type	Function
$\overline{HIGHZ}$	J1	I	**High Impedance.** Assertion causes the DMAC to 3-state all its outputs.

Characteristics

Timing Characteristics

All TTL timing specifications are referenced to and from 0.8 V for a low voltage and 2.0 V for a high voltage. CMOS clock references are to and from Vcc/2. All min and max values are in ns.

Table 17. Timing Characteristics

Num	Symbol	Description	Fig(s)	10 MHz		14 MHz		18 MHz	
				Min	Max	Min	Max	Min	Max
1	tDATVC34H	Data set-up time	9	13	—	7	—	5	—
2A	tDSBHDATZ	Data hold time	9	0	—	0	—	0	—
2B	tC34HDATZ	Data hold time	9	30	—	24	—	20	—
3	tSRYLC34L	Synchronous ready set-up time	9	15	—	13	—	6	—
4	tC34LSRYH	Synchronous ready hold time	9	20	—	14	—	11	—
5	tDTALC34H	Data transfer acknowledge set-up time*	9	15	—	10	—	8	—
6	tDSBHDTAH	Data transfer acknowledge assertion time	9	0	—	0	—	0	—
7	tC34LDSZV	Data size assertion time	9	—	62	—	44	—	34
8	tC34LADDV	Address assertion time	9	—	42	—	38	—	36
9	tC34HASBL	Address strobe assertion time	9	—	50	—	35	—	27
10	tC34HASBH	Address strobe negation time	9	—	56	—	33	—	26
11	tC34HASBX	Address strobe negation time	9	-23	—	-17	—	-14	—
12	tC34HDSBL	Data strobe assertion time	9,10	—	50	—	35	—	27
12A	tDSBLDSBH	Data strobe assertion width time	10	0.9Tc	—	0.9Tc	—	0.9Tc	—
13	tC34HDSBH	Data strobe negation time	9	—	58	—	38	—	31
14	tDRYXDRYL	Data ready assertion time	9	—	56	—	35	—	27
15	tC34HDRYH	Data ready negation time	9	—	56	—	33	—	26
16	tC34LCYCL	Cycle initiate assertion time	9	—	60	—	40	—	30
17	tC34LCYCH	Cycle initiate negation time	9	—	62	—	35	—	29
18	tC34LDATV	Data assertion time	10	—	57	—	40	—	33
19	tC34LRWL	Read/write strobe assertion time	10,15	—	62	—	38	—	29
19A	tRWLASBL	Read/write strobe set-up time	10	25	—	25	—	19	—
19B	tRWLDSBL	Read/write strobe set-up time	10	125	—	95	—	77	—

* These are the asynchronous signals; set-up times are specified for testing and informational purposes. The set-up times guarantee recognition at the next clock edge.

Table 17. Timing Characteristics (Continued)

Num	Symbol	Description	Fig(s)	10 MHz		14 MHz		18 MHz	
				Min	Max	Min	Max	Min	Max
20	tBASLC34L	Block access set-up time	11	17	—	10	—	8	—
21	tC34LBASX	Block access hold time	11	20	—	16	—	12	—
22	tDSBHBASX	Block access assertion block	11	0	—	0	—	0	—
23	tC34LPRAV	Peripheral address assertion time	18	—	70	—	55	—	45
23A	tPRAVPCSL	Peripheral address set-up time	18	20	—	12	—	6	—
25	tC34HPCSL	Peripheral chip select assertion time	18	—	56	—	35	—	27
25A	tPCSLPDSL	Peripheral chip select set-up time	18	-2	—	-3	—	-5	—
26	tC34HPCSH	Peripheral chip select negation time	18	—	56	—	33	—	26
27	tC34HPCSH	Peripheral chip select negation time	18	-10	—	-10	—	-10	—
28	tC34HPDSL	Peripheral data strobe assertion time	18,19	—	56	—	35	—	27
29	tC34HPDSH	Peripheral data strobe negation time	18,20	—	56	—	33	—	26
30	tPDTLC34H	Peripheral data transfer acknowledge set-up time*	18	17	—	10	—	8	—
31	tIPDLC23L	Interrupt pending set-up time*	15	15	—	10	—	8	—
32	tC34HPDTH	Peripheral data transfer acknowledge hold time	18	0	—	0	—	0	—
33	tPRDVC34H	Peripheral data set-up time	18	15	—	10	—	8	—
34	tC34HPRDZ	Peripheral data hold time	18	26	—	22	—	17	—
35	tPDSHPRDZ	Peripheral data hold time	18	0	—	0	—	0	—
36	tBRQHIPDH	Interrupt pending hold time	15	0	—	0	—	0	—
37	tC34LPRWL	Peripheral read/write strobe assertion time	19	—	62	—	44	—	34
38	tC34LPRDV	Peripheral data assertion time	19	—	54	—	40	—	31
38A	tPRDVPDSL	Peripheral data set-up time	19	25	—	18	—	14	—
38B	tPRDHPDSH	Peripheral data hold time	—	20	—	14	—	11	—
39	tPBRHC34H	Peripheral bus request set-up time*	20	17	—	10	—	8	—

* These are the asynchronous signals; set-up times are specified for testing and informational purposes. The set-up times guarantee recognition at the next clock edge.

Table 17. Timing Characteristics (Continued)

Num	Symbol	Description	Fig(s)	10 MHz		14 MHz		18 MHz	
				Min	Max	Min	Max	Min	Max
40	tFATLC34H	Asynchronous fault set-up time*	13	17	—	10	—	8	—
41	tDTALFATL	Delayed fault time	12	—	45	—	30	—	23
42A	tDSBHFATH	Delayed fault hold time	12	0	—	0	—	0	—
42B	tDSBHFATH	Asynchronous fault hold time	13	0	—	0	—	0	—
43	tRTYLC34H	Asynchronous retry set-up time*	13	17	—	10	—	8	—
44	tDTALRTYL	Delayed retry time	12,13	—	45	—	30	—	20
45A	tDSBHRTYH	Delayed retry hold time	12	0	—	0	—	0	—
45B	tDSBHRTYH	Asynchronous retry hold time	13	0	—	0	—	0	—
46	tRRRLC34H	Asynchronous relinquish and retry request set-up time*	13	17	—	10	—	8	—
47	tDTALRRRL	Delayed relinquish and retry time	12	—	45	—	30	—	20
48A	tASBHRRRH	Delayed relinquish and retry request hold time	12	0	—	0	—	0	—
48B	tDSBHRRRH	Asynchronous relinquish and retry request hold time	13	0	—	0	—	0	—
49	tFATLC34L	Synchronous fault set-up time	—	17	—	10	—	8	—
50	tC34LFATH	Synchronous fault hold time	13	20	—	16	—	12	—
51	tRTYLC34L	Synchronous retry set-up time	13	17	—	10	—	8	—
52	tC34LRTYH	Synchronous retry hold time	13	20	—	16	—	12	—
53	tRRRLC34L	Synchronous relinquish and retry request set-up time	13,14	17	—	10	—	8	—
54	tC34LRRRH	Synchronous relinquish and retry request hold time	13,15	20	—	16	—	12	—
55	tBRAHC34L	Bus request acknowledge set-up time*	14	17	—	10	—	8	—
56	tC34LBRQL	Bus request assertion time	14	—	56	—	35	—	27
57	tC34LBRQH	Bus request high impedance time	15	—	56	—	33	—	26
58	tBRQHBRAH	Bus request acknowledge assertion time	15	0	—	0	—	0	—
59	tC34LADDZ	Address 3-state time	—	—	72	—	50	—	39
60	tC34LDSZZ	Data size 3-state time	15	—	62	—	50	—	39

* These are the asynchronous signals; set-up times are specified for testing and informational purposes. The set-up times guarantee recognition at the next clock edge.

Table 17. Timing Characteristics (Continued)

Num	Symbol	Description	Fig(s)	10 MHz		14 MHz		18 MHz	
				Min	Max	Min	Max	Min	Max
61	tC34LDATZ	Data 3-state time	15	—	72	—	50	—	39
62	tC34LASBZ	Address strobe 3-state time	15	—	60	—	40	—	31
63	tC34LDSBZ	Data strobe 3-state time	15	—	60	—	40	—	31
64	tCSLADDV	Address delay peripheral mode time	16,17	—	105	—	75	—	58
65	tC34LDRYZ	Data ready 3-state time	15	—	60	—	40	—	31
66	tC34LCYCZ	Cycle initiate 3-state time	15	—	60	—	40	—	31
67	tC34LRWZ	Read/write strobe 3-state time	15	—	60	—	40	—	31
70	tCSLC23H	Peripheral mode chip select set-up time*	16,17	17	—	10	—	8	—
71	tCSLDSBL	Peripheral mode data strobe delay time	16,17	—	95	—	65	—	50
72	tDTALDSBH	Peripheral mode data strobe hold time	16	0	—	0	—	0	—
73	tCSLRWH	Peripheral mode read/write strobe delay time	16,17	—	95	—	70	—	54
74	tDTALRWX	Peripheral mode read/write strobe hold time	16,17	0	—	0	—	0	—
75	tCSLDATV	Peripheral mode data delay time	17	—	125	—	90	—	70
76	tDTALDATX	Peripheral mode data hold time	17	0	—	0	—	0	—
77	tDTALADDX	Peripheral mode address hold time	16	0	—	0	—	0	—
78	tC34HDATV	Peripheral mode & interrupt acknowledge data assertion time	16	—	100	—	63	—	56
79	tDSBHDATZ	Peripheral mode data hold time	16	0	—	0	—	0	—
80	tDSBHDATZ	Peripheral mode & interrupt acknowledge data 3-state time	16	—	172	—	120	—	93
81	tC34LDTAL	Data transfer acknowledge assertion time	16	—	56	—	35	—	27
83	tDSBHRWH	Data transfer acknowledge high impedance	16	—	56	—	35	—	27

* These are the asynchronous signals; set-up times are specified for testing and informational purposes. The set-up times guarantee recognition at the next clock edge.

Table 17. Timing Characteristics (Continued)

Num	Symbol	Description	Fig(s)	10 MHz		14 MHz		18 MHz	
				Min	Max	Min	Max	Min	Max
85	tIACLC34H	Interrupt acknowledge set-up time*	22	15	—	10	—	8	—
86	tDSBHC34L	Interrupt acknowledge hold time	22	0	—	0	—	0	—
87	tDTALCSX	Peripheral mode chip select hold time	16	0	—	0	—	0	—
88	tRRALC34L	Relinquish and retry acknowledge assertion time	—	—	56	—	35	—	27
89	tRRAZC34L	Relinquish and retry acknowledge high impedance	—	—	62	—	33	—	26
92	tRSTLC23L	Reset set-up time*	—	17	—	10	—	8	—
93	tADDVASBL	Address set-up time	9	25	—	18	—	12	—
94	tASBHADDZ	Address hold time	9	20	—	18	—	7	—
95	tDATVDSBL	Data set-up time	10	20	—	15	—	12	—
96	tDSBHDATZ	Data hold time	10	25	—	18	—	11	—
97	tRESET	Reset valid to guarantee reset	—	3Tc	—	3Tc	—	3Tc	—
98	tOUTZHIGL	All outputs 3-stated	—	—	100	—	70	—	58
99	tC34LINRL	Interrupt request assertion time	15	—	65	—	45	—	35
100	tC34LINRH	Interrupt request negation time	22	—	75	—	45	—	35
101A	tPDSHPBRH	Peripheral bus request hold time (burst mode)	20	0	—	0	—	0	—
101B	tPCSLPBRL	Peripheral bus request hold time (non-burst mode)	—	0	—	0	—	0	—
102	tPDTHPBRH	Peripheral bus request hold time with respect to PDTACK negation	20	0	—	0	—	0	—
103	tPDNLC34H	Peripheral done set-up time	21	—	15	—	10	—	8

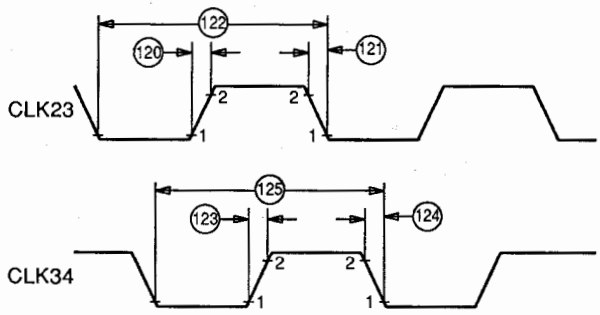
* These are the asynchronous signals; set-up times are specified for testing and informational purposes. The set-up times guarantee recognition at the next clock edge.

Table 17. Timing Characteristics (Continued)

Num	Symbol	Description	Fig(s)	10 MHz		14 MHz		18 MHz	
				Min	Max	Min	Max	Min	Max
104	tPDNLPDTL	Peripheral done set-up time relative to PDTACK	21	1.25Tc	—	1.25Tc	—	1.25Tc	—
105	tC34HPDNH	Peripheral done hold time	21	Tc	—	Tc	—	Tc	—
106	tC34HFATL	Fault assertion time	23	—	56	—	56	—	44
107	tC34HFATZ	Fault high impedance	23	—	60	—	60	—	47
120	tC23L1C23H2	Clock 23 rise time	8	—	4	—	4	—	4
121	tC23H2C23L1	Clock 23 fall time	8	—	4	—	4	—	4
122	tC23L1C23L1	Clock 23 period	8	100	—	71.4	—	56	—
123	tC34L1C34H2	Clock 34 rise time	8	—	4	—	4	—	4
124	tC34H2C34L1	Clock 34 fall time	8	—	4	—	4	—	4
125	tC34L1C34L1	Clock 34 period	8	100	—	71.4	—	56	—
126	tC23J	Clock 23 jitter	—	—	0.5	—	0.5	—	0.5
127	tC23E	Clock 23 duty cycle error	—	—	3	—	2	—	2
128	tC34J	Clock 34 jitter	—	—	0.5	—	0.5	—	0.5
129	tC34E	Clock 34 duty cycle error	—	—	3	—	2	—	2
130	tSKEW	Clock skew	—	—	3	—	2	—	2

Timing Diagrams

These timing diagrams represent a subset of all possible transactions and are intended only for the purpose of displaying and clarifying timing relationships.

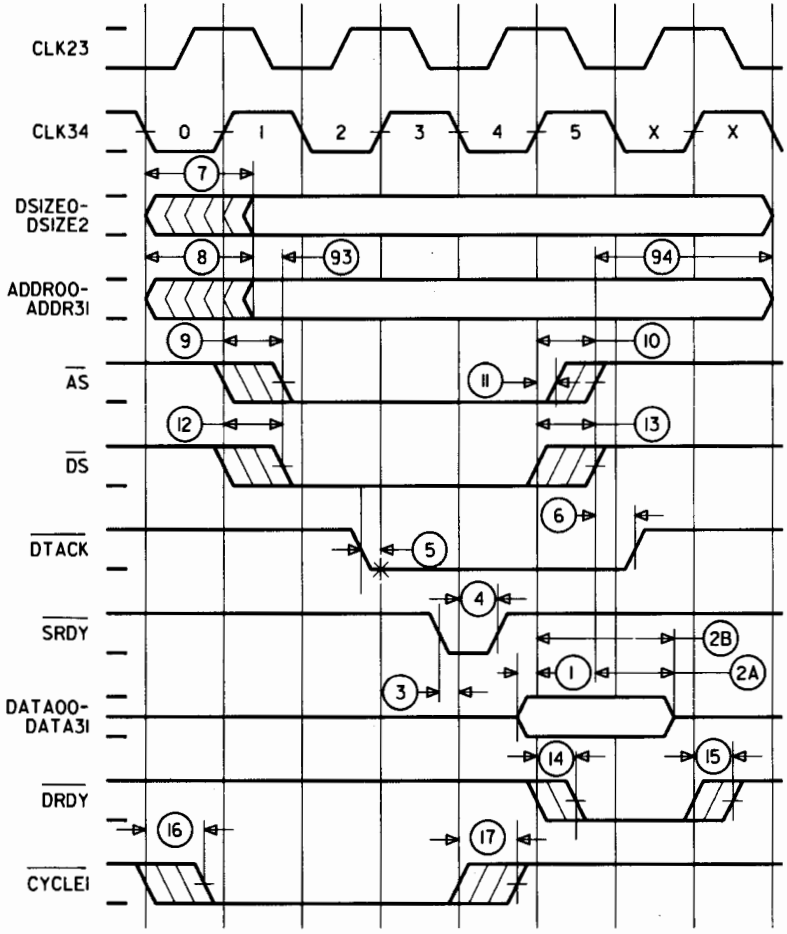


Notes:
Duty Cycle Error – The duty cycle of each clock input may deviate from 50% but should not exceed timing specification numbers 127 and 129 for CLK23 and CLK34, respectively.

Skew – CLK23 nominally leads CLK34 by 90 ° (1/4 clock period). This phase lead should never exceed timing specification 130.

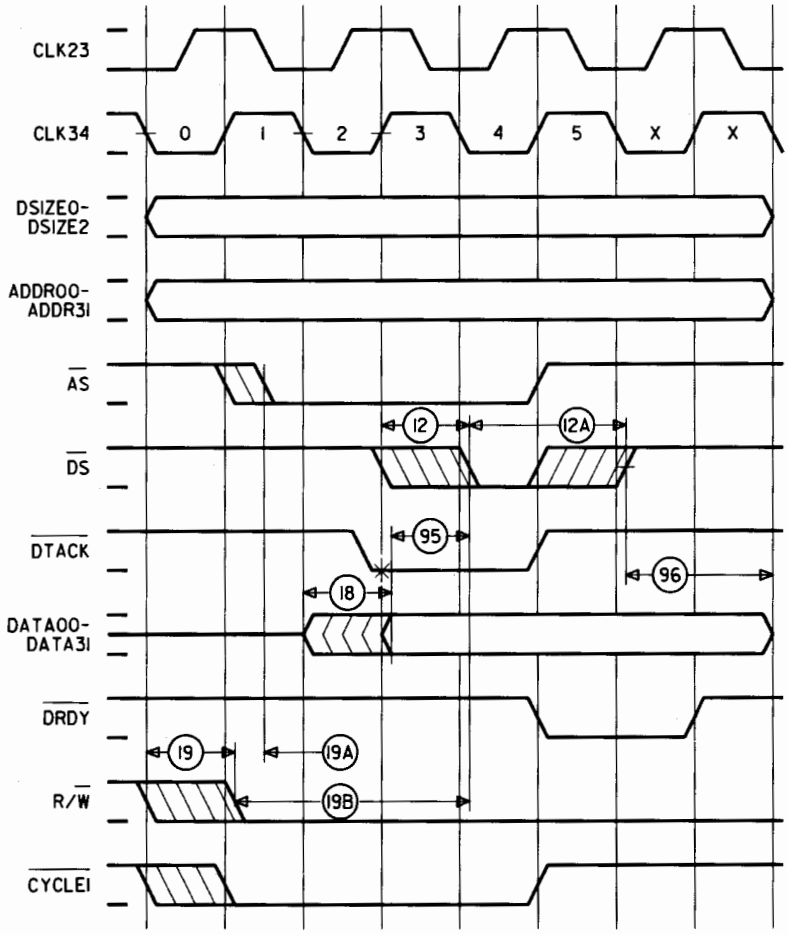
Jitter – The period of each clock input may deviate from its nominal value but should not exceed timing specification numbers 126 or 128 for CLK23 and CLK34, respectively.

Figure 8. Clock Inputs



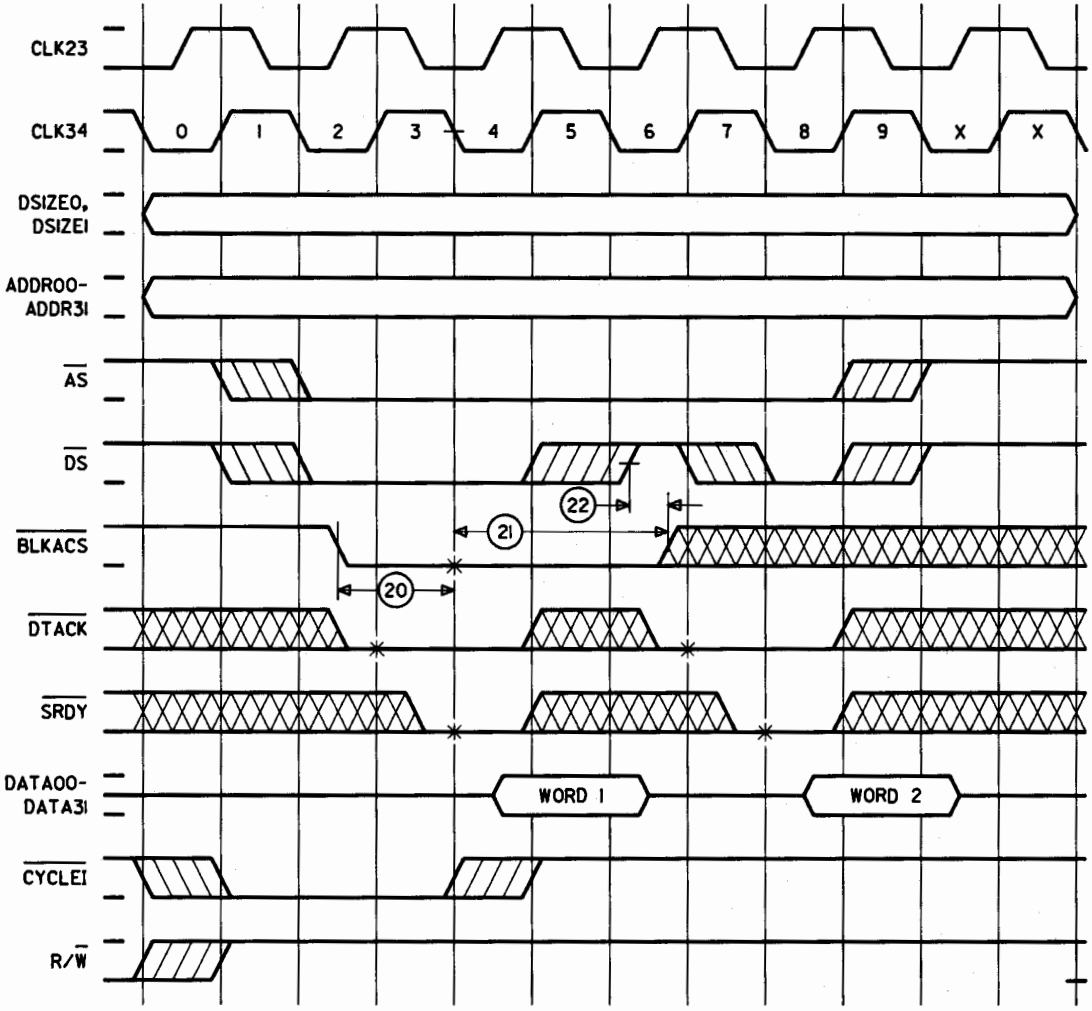
Note: System bus. No wait states.
* Indicates when signal is sampled.

Figure 9. DMA Controller Read



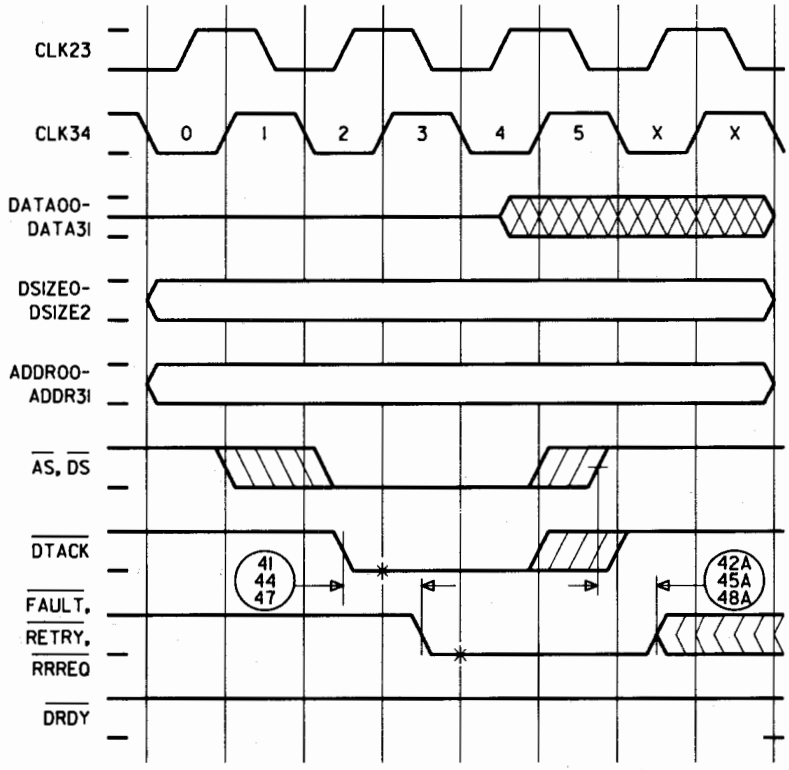
Note: System bus. No wait states.
* Indicates when signal is sampled.

Figure 10. DMA Controller Write



* Indicates when signal is sampled.

Figure 11. Double-Word Transaction



* Indicates when signal is sampled.

Figure 12. Bus Exception After $\overline{\text{DTACK}}$, $\overline{\text{FAULT}}$, $\overline{\text{RETRY}}$, $\overline{\text{RRREQ}}$

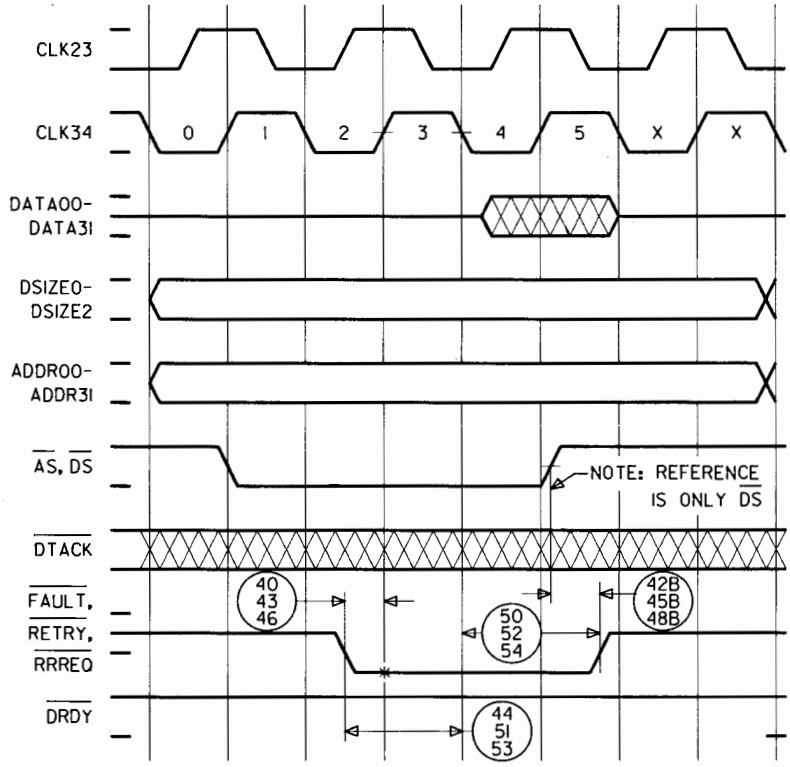
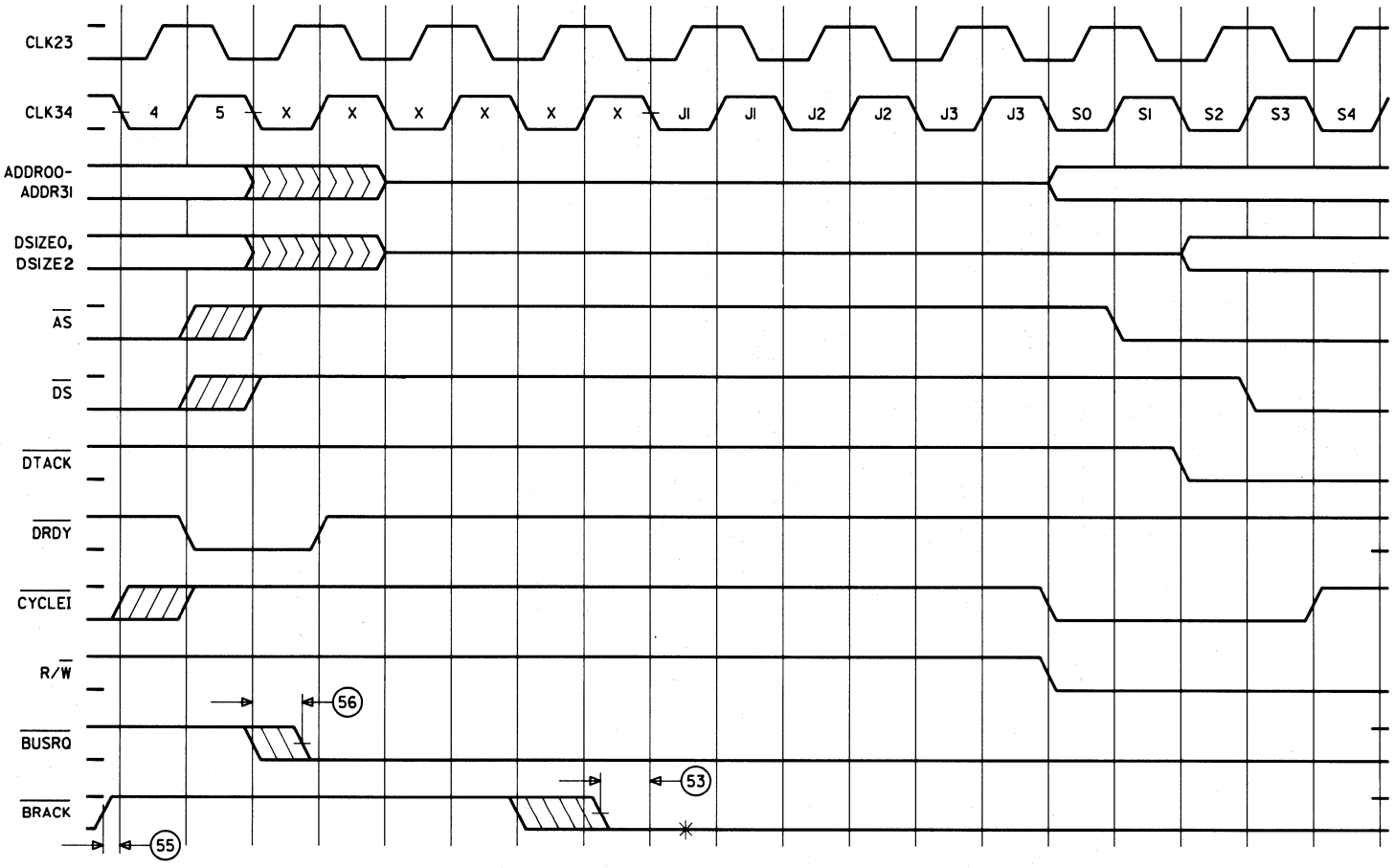
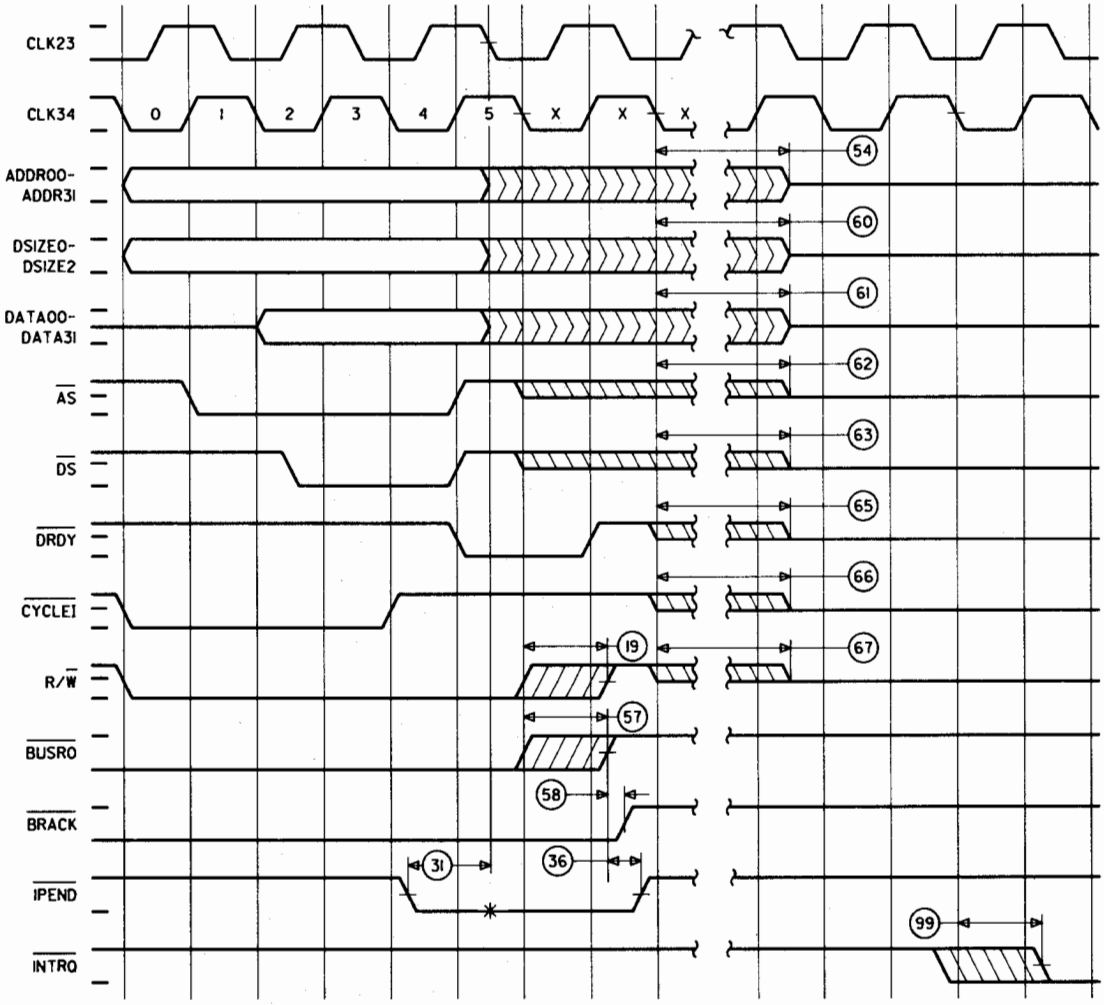


Figure 13. Bus Exception Timing Before $\overline{\text{DTACK}}$, $\overline{\text{FAULT}}$, $\overline{\text{RETRY}}$, or $\overline{\text{RRREQ}}$



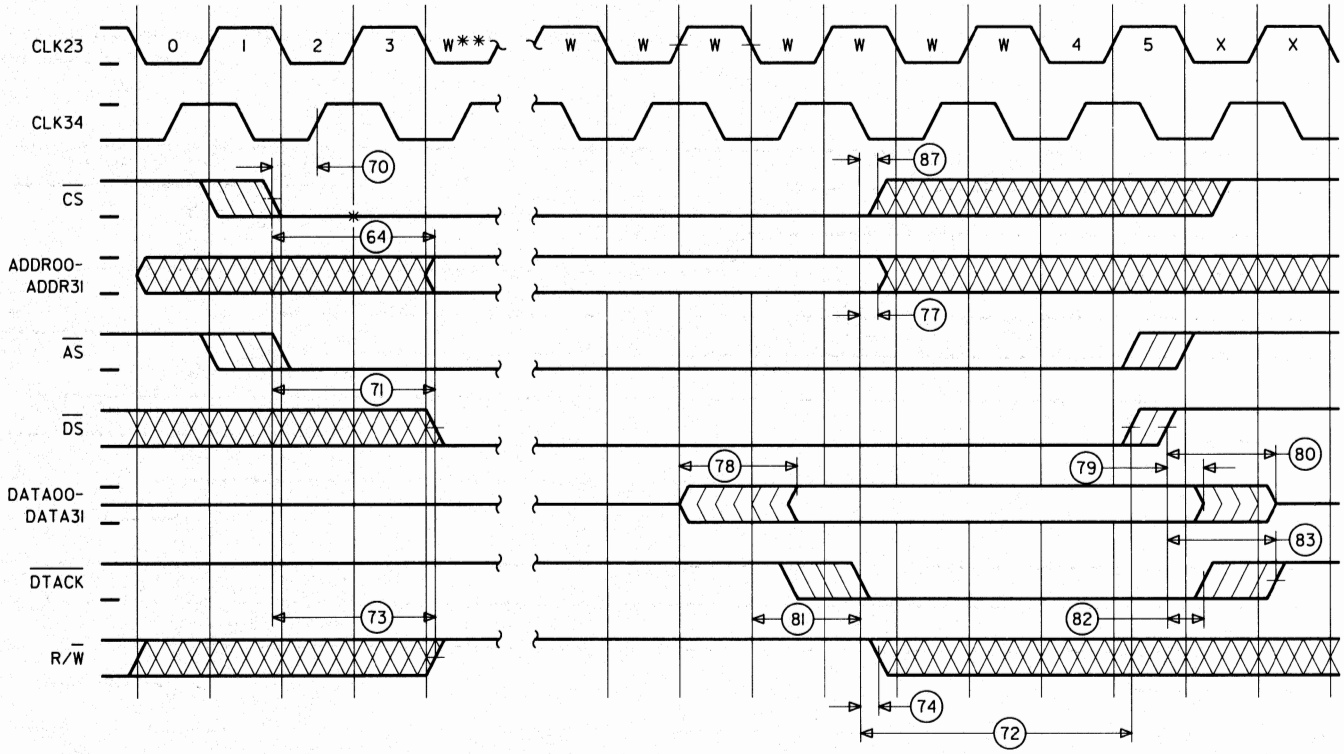
* Indicates when signal is sampled.

Figure 14. Bus Arbitration



* Indicates when signal is sampled.

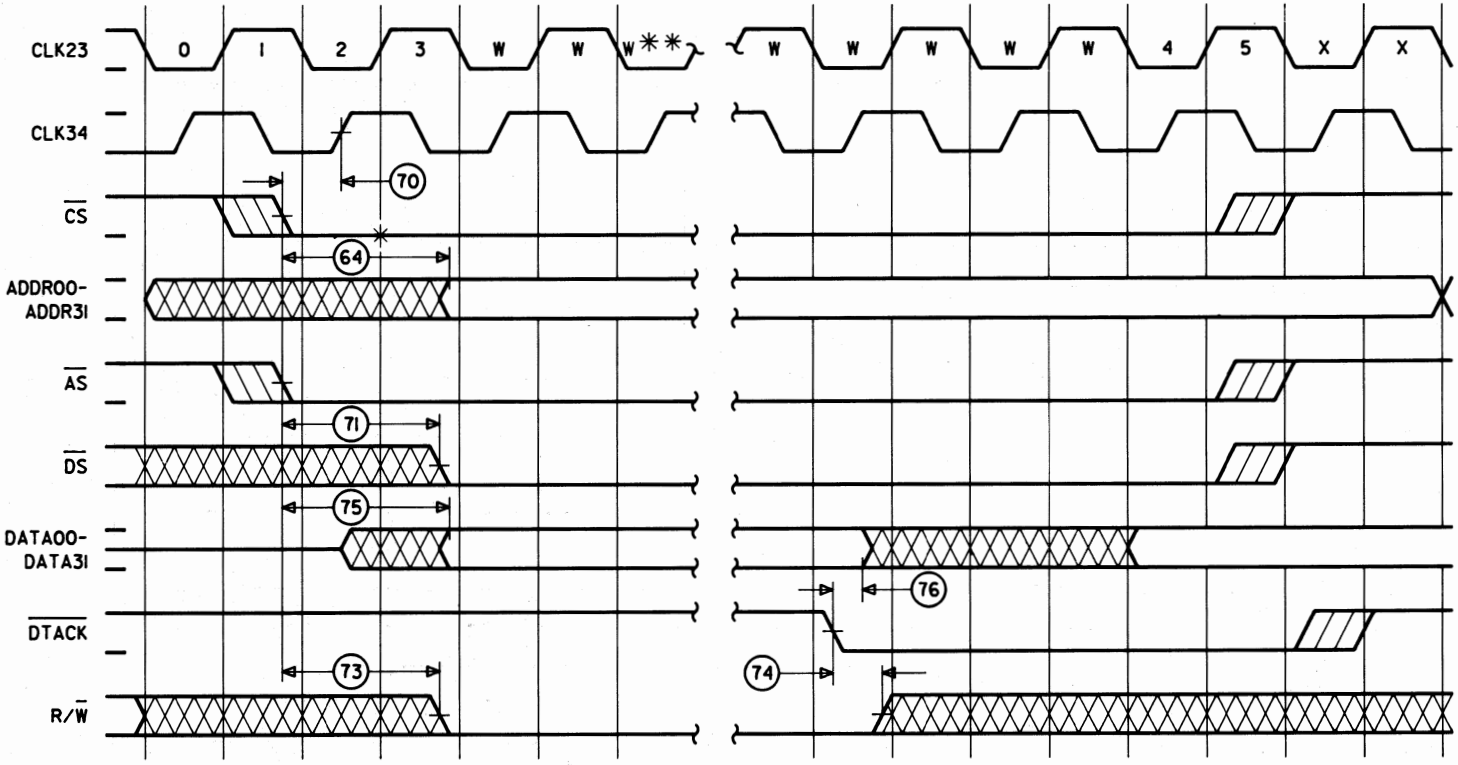
Figure 15. Bus Arbitration (Releasing the System Bus)



* Indicates when signal is sampled.

** Number of wait states depends on whether read is from DMAC register or peripheral device.

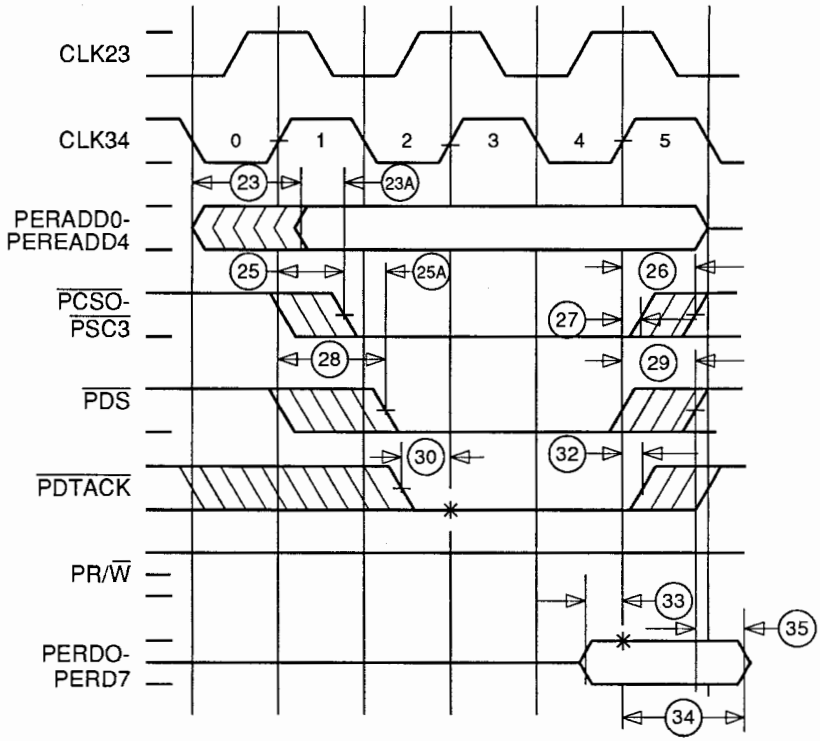
Figure 16. Peripheral Mode Read of DMAC Register or Peripheral Device



* Indicates when signal is sampled.

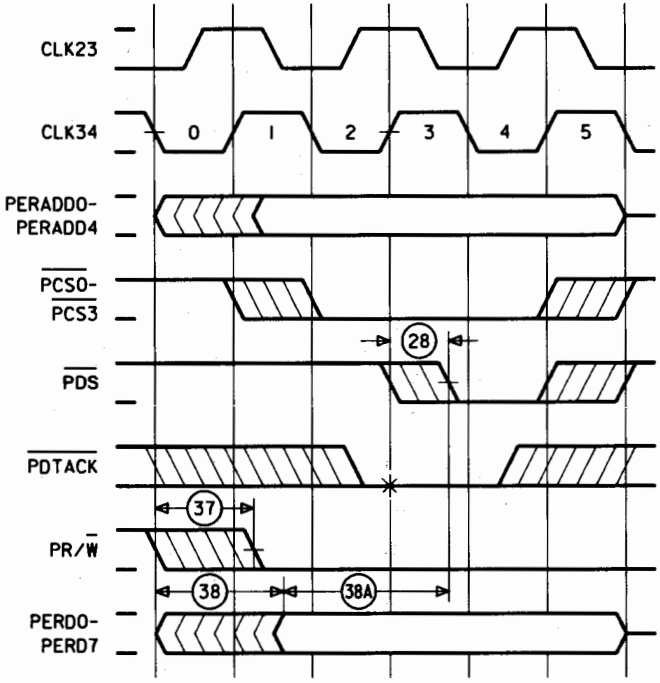
** Number of wait states depends on whether write is to DMAC register or peripheral device.

Figure 17. Peripheral Mode Write to DMAC Register or Peripheral Device



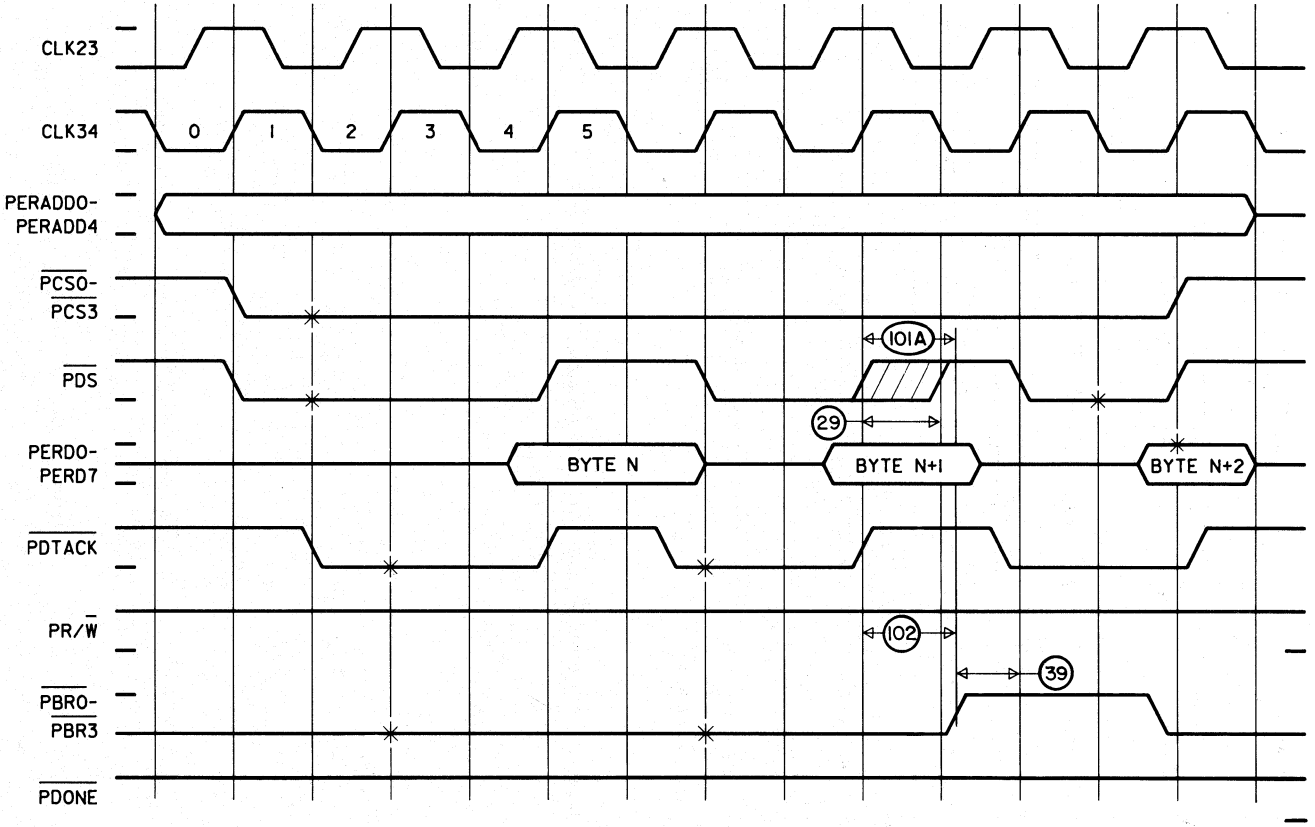
Note: Single byte. Zero wait states.
* Indicates when signal is sampled.

Figure 18. Perpheral Bus Read



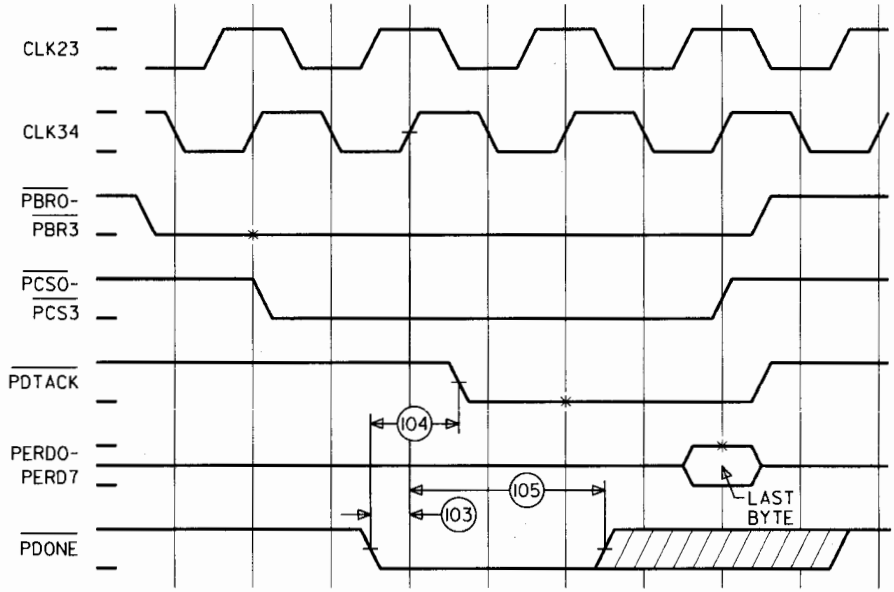
Note: Single byte. Zero wait states.
* Indicates when signal is sampled.

Figure 19. Peripheral Bus Write



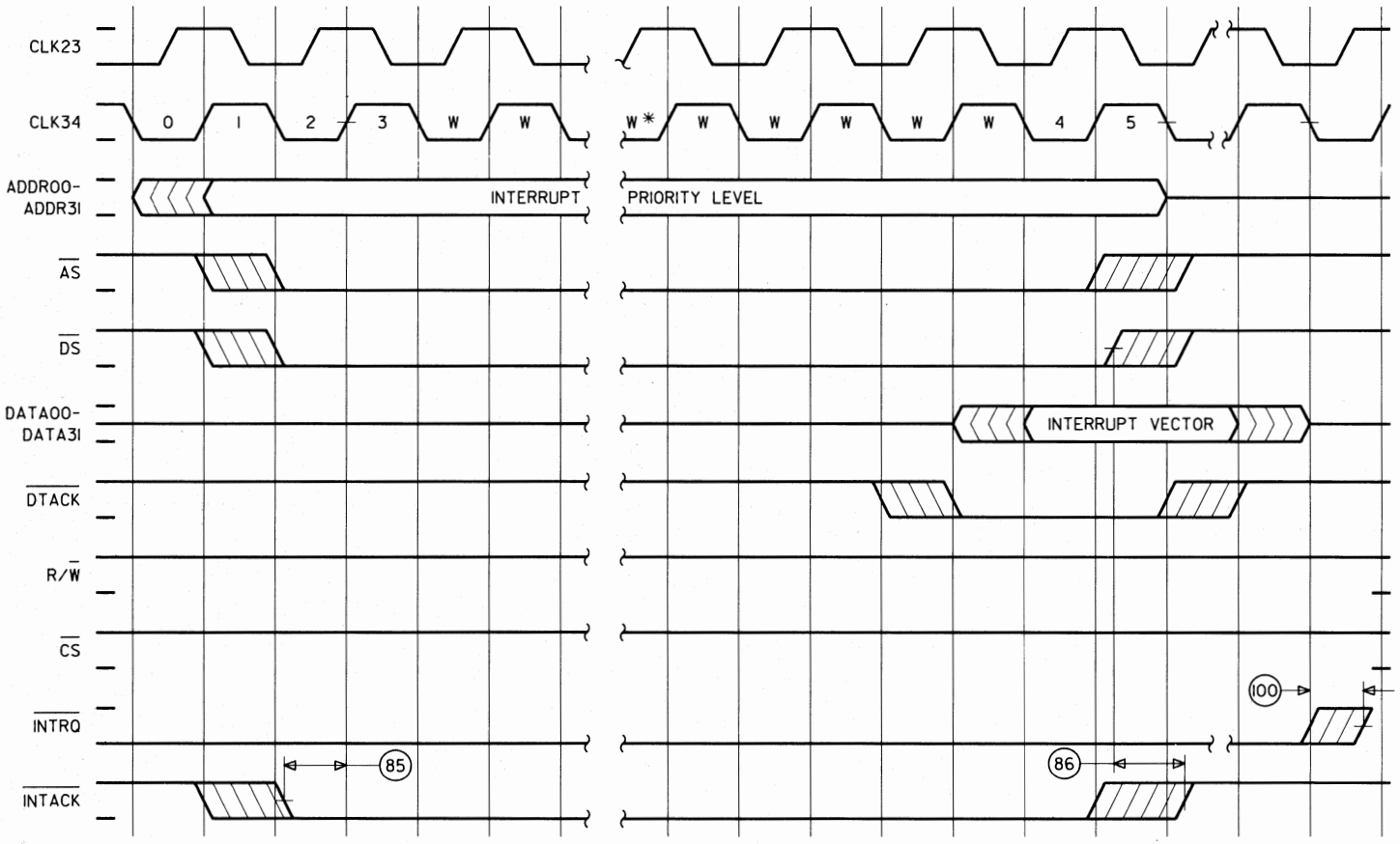
Note: Zero wait states. Terminated by removal of PBR.
* Indicates when signal is sampled.

Figure 20. Asynchronous Peripheral Bus Burst



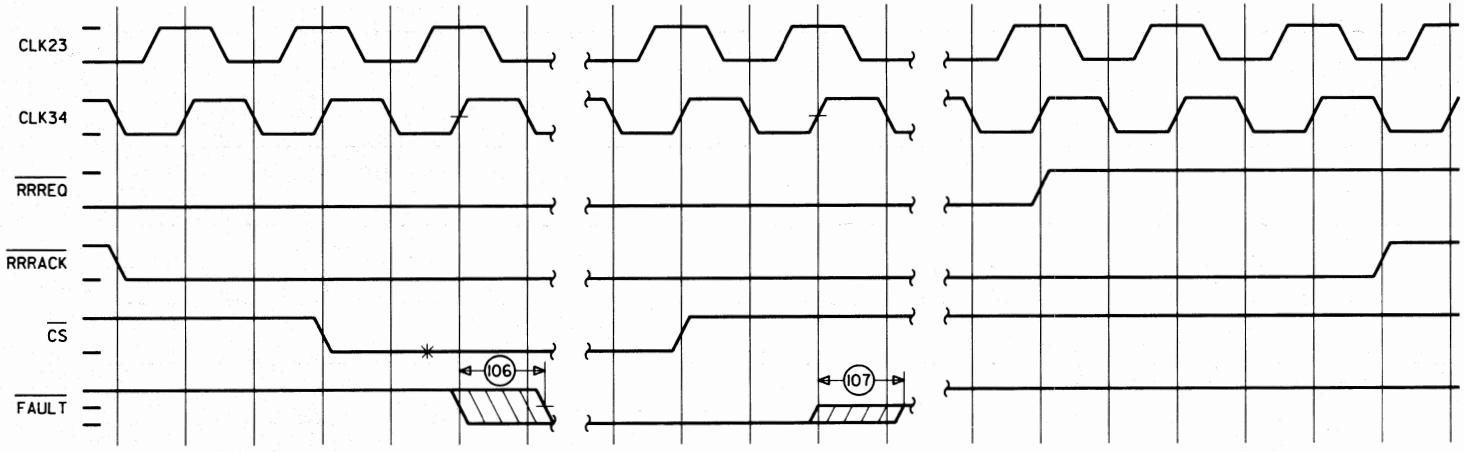
* Indicates when signal is sampled.

Figure 21. Channel Termination Using Peripheral Done (BL = 0)



Note: The number of wait states between the assertion of $\overline{\text{INTACK}}$ and $\overline{\text{DTACK}}$ is between 4 and 9.

Figure 22. Interrupt Acknowledge



* Indicates when signal is sampled.

Figure 23. $\overline{CS}$ Asserted During Relinquish and Retry

Electrical Characteristics

Inputs

All inputs, except the two CMOS clocks, are TTL compatible. The TTL inputs and clock inputs are specified separately below.

Table 18. DC Input Parameters

Inputs		Min	Nom	Max	Unit
TTL input voltage	high-level	2	—	V_{CC}+0.5	V
	low-level	–0.5	—	0.8	V
CMOS clocks input voltage	high-level	V_{CC}–1.3	—	V_{CC}+0.5	V
	low-level	0	—	0.8	V
TTL input loading current 2.0 V ≤ V_{IH} ≤ V_{CC}	high-level	0	—	0.01	mA
TTL input loading current: 0 V ≤ V_{IL} ≤ 0.8 V	low-level	–0.01	—	0	mA
CMOS clocks input loading current V_{CC} – 1.5 V ≤ V_{IH} ≤ V_{CC}	high-level	0	—	0.01	mA
CMOS clocks input loading current 0 ≤ V_{IL} ≤ 1.0	low-level	–0.01	—	0	mA

Outputs

The list below contains descriptions of the outputs assigned to classes 1, 2 and 3.

Class 1: Capable of driving 1 TTL load or 8 PNP Schottky TTL loads. Has current allowance for external holding resistor employed in 3-state buffers. The minimum holding resistor value is 2.7 kΩ.

Class 2: Same as class 1 except does not have current allowance for pull up resistor.

Class 3: Open drain device used for wired-logic operations, allowing more than one device to drive a node without conflict. An external resistor is required to pull the signal high. The minimum holding resistor value is 680 Ω.

The following lists the outputs assigned to classes 1, 2 and 3:

Class 1		Class 2	Class 3
$\overline{AS}$	$\overline{PDS}$	ADDR00—ADDR31	$\overline{BUSRQ}$
$\overline{CYCLE1}$	PR/$\overline{W}$	DATA00—DATA31	$\overline{DTACK}$
$\overline{DRDY}$	R/$\overline{W}$	$\overline{INTRQ}$	$\overline{RRRACK}$
$\overline{DS}$		$\overline{PCS0}$—$\overline{PCS3}$	$\overline{FAULT}$
DSIZE0—DSIZE2		PERADD0—PERADD4	
		PERD0—PERD7	

Table 19. DC Output Parameters

Outputs		Min	Nom	Max	Unit
Output sink current (I_{OL}) ($V_{OL} \leq 0.4$ V)	Class 1	—	—	5.5	mA
	Class 2	—	—	3.5	mA
	Class 3	—	—	12	mA
Output source current (I_{OH}) ($V_{OH} \geq 2.4$ V)	Class 1	—	—	5.5	mA
	Class 2	—	—	3.5	mA
	Class 3	—	—	10	μA
Output logic levels	high-level	2.4	—	—	V
	low-level	—	—	0.4	V

Operating Conditions

Table 20. DC Operating Conditions

Parameter		Symbol	Min	Nom	Max	Unit
Supply voltage		V_{CC}	4.75	5.00	5.25	V
Input load capacitance	TTL inputs	C_{IN}	—	—	12	pF
	CMOS clocks		—	—	7	pF
Total output load capacitance	Class 1	C_L	—	—	130	pF
	Class 2		—	—	130	pF
	Class 3		—	—	130	pF
Ambient temperature at the microprocessor pins		T_A	0	—	70	°C
Humidity range		—	5%	—	95%	—
Power dissipation	at 10 MHz	PD	—	—	1.38	W
	at 14 MHz		—	—	1.9	W
	at 18 MHz		—	—	2.5	W
Operating frequency		F	—	—	18.0	MHz

Table 21. Capacitive Derating Factor (dt/dC)

Output	10 MHz	14 MHz	18 MHz
Class 1	0.1	.08	.07
Class 2	.06	.05	.04
Class 3	0.1	.08	.07

Note: All values are nominal in units of ns/pF ($25pF \leq C_L \leq 225pF$).

WE® 32204 DMA Controller

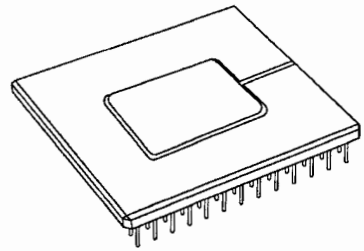
Description

Direct memory access (DMA) is a mechanism for servicing I/O device data transfer requests. It is driven by an I/O device to provide direct access to system memory. The *WE* 32204 Direct Memory Access Controller (DMAC) provides DMA capabilities while taking advantage of the full 32-bit data and address of the *WE* 32200 Microprocessor. The DMAC contains specialized hardware that permits transfers at a much faster rate than possible under microprocessor control.

The DMAC is a memory-mapped peripheral device that performs memory-to-memory, memory-to-peripheral, and peripheral-to-memory data transfers quickly and efficiently. When used with the *WE* 32200 Microprocessor, the DMAC permits the full 32-bit width of the system bus to be used without external interfacing logic. In addition, a peripheral bus is provided to couple 8-bit input/output devices to the system bus. The DMAC is implemented in CMOS technology; is available in a 133-pin square, hermetic, ceramic pin grid array; and requires a single 5 V supply. The DMAC is available in 24 MHz and higher versions.

Features

- Full 32-bit address and data buses
- 8-bit peripheral bus for coupling I/O devices to the system bus
- Double- and quad-word bus cycles available for high system throughput
- Internal data buffers to support burst data peripherals
- Four independent prioritized DMA channels
- Two programmable interrupt vectors per channel



- Memory-to-memory transfers at rates of up to 19.2 Mbytes/s at 24 MHz
- Memory to peripheral transfers at rates of up to 10.4 Mbytes/s at 24 MHz (with burst mode)
- Memory fill operations available for writing an arbitrary constant to a block of memory

User Information

Architectural Summary

The DMAC has four independent channels, which allow it to serve four unrelated transfer requests simultaneously. Each channel has a set of registers that configures and controls its operation. In addition, there is one mask register that is shared between the four channels. When the DMAC is in peripheral mode, registers within the DMAC and registers within devices connected to the peripheral bus are accessed by the CPU. This procedure provides the CPU an access path to the peripheral bus.

The seven functional elements of the DMAC are:

System Bus Interface — provides the address, data, and control signals needed to interface the DMAC to the *WE* 32200 Microprocessor.

Data Buffers — each of the four channels has a 32-byte data buffer. The lowest address data buffer is used as the memory fill data register (MFDR). This register contains the data that is written to consecutive locations during memory fill operations.

Register File — contains eight control registers for each DMA channel and one global register. The control registers consist of source address, destination address, transfer count, base address, mode, device control, interrupt vector, and status and control registers. The global register is a mask register used to disable particular channel activity.

Address Arithmethic Unit (AAU) — calculates the addresses of source and destination using information from the per channel registers and the data size and request generator.

Count Arithmetic Unit (CAU) — calculates the number of bytes to be transferred using information from the per channel registers and the data size and request generator.

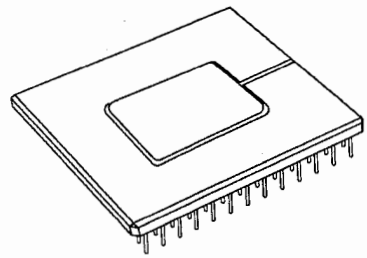
Packing Registers — used to pack bytes into larger operands when transferring from the peripheral bus to the system bus. Also, it is used to unpack large system bus operands to bytes when transferring to the peripheral bus.

Peripheral Bus — used to communicate with 8-bit I/O devices.

WE ® 32106 Math Acceleration Unit

Description

The *WE* 32106 Math Acceleration Unit (MAU) provides floating-point capability for the *WE* 32100 Microprocessor and is fully compatible with the IEEE Standard for Binary Floating-Point Arithmetic (ANSI/IEEE Std. 754-1985). It provides single (32-bit), double (64-bit), and double-extended (80-bit) precision for add, subtract, multiply, divide, remainder, square root, and compare operations. The operand, result, and status and command information transfers take place over a 32-bit bidirectional data bus that provides the interface to the host microprocessor. The MAU is implemented in CMOS technology. It is available in 10-, 14-, and 18-MHz version; requires a single 5 V supply; and is available in a 125-pin square, ceramic pin grid array (PGA) package.



Features

- Compatible with ANSI/IEEE Standard 754-1985 for Binary Floating-Point Arithmetic
- Single (32-bit), double (64-bit), and double-extended (80-bit) precision capability
- Add, subtract, multiply, divide, remainder, negate, absolute value, and square root functions
- Compare, move, and rounding to integral value functions
- Coprocessor and peripheral mode interfaces available
- Symmetric integer, decimal, and floating-point conversions
- 32-bit I/O interface

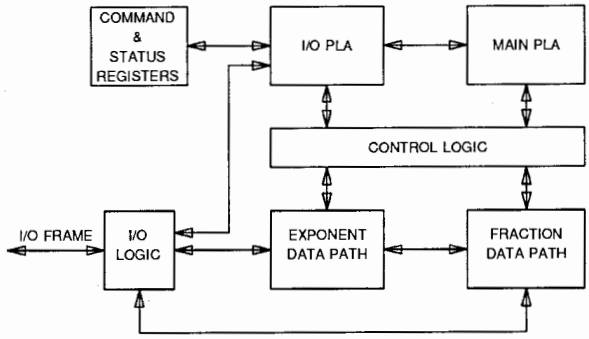


Figure 1. WE® 32106 Math Acceleration Unit Block Diagram

Description 365
Features 365
User Information 366
Architectural Summary 366
Protocols 366
Coprocessor Mode 366
Peripheral Mode 367
Data Types and Formats 367
Single-Precision 367
Double-Precision 367
Double-Extended-Precision 367
Decimal 367
Integer 371
Registers 371
Auxiliary Status Register 371
Operand Registers 371
Command Register 371
Data Register 371
Instruction Set Summary by Mnemonic and Opcode 375
Instruction Set Summary by Functional Groupings 378
Arithmetic Group 378
Data Transfer Group 378
Logical Group 378
Conversion Group 378
Miscellaneous Group 378
Rounding 378
Exceptions 378
Interrupts 379
Restart Capability 379
Peripheral Mode Accesses 379
Operating System Considerations 379
Peripheral Mode Protocol 379
Coprocessor Mode Protocol 380
Context Saving and Restoring 380
Peripheral Mode 380
Coprocessor Mode 381
System Configuration Restrictions 381
Peripheral Mode Protocol 381
Coprocessor Mode Protocol 381
MAU-CPU Interconnections 381
Definitions 381
Pin Descriptions 383
Numerical Order 384
Functional Group 385
Characteristics 386
Timing Characteristics 386
Timing Diagrams 389
Electrical Characteristics 393
Inputs 393
Outputs 393
Operating Conditions 394

User Information

Architectural Summary

The MAU has two major subsystems: the fraction datapath and the exponent datapath. The data width of each datapath was chosen to facilitate calculations in the widest data format. All operands are converted, upon entry to the MAU from memory, to the double-extended precision format. This causes no loss of precision and saves internal hardware that would be necessary to explicitly support the narrower formats in all internal operations.

Protocols

The MAU interacts with a system via two different protocols: coprocessor and peripheral mode protocol. The coprocessor mode protocol allows high system throughput when the MAU is used with the *WE* 32100 Microprocessor. The peripheral mode protocol allows use of the MAU with any CPU that can perform data read and write bus transactions.

Coprocessor Mode

In coprocessor mode protocol, the CPU initiates an MAU transaction by performing a coprocessor broadcast access. This access includes a word in the MAU command register (CR) format (see Registers). The MAU checks the ID field of this word against the MAU ID value of 0. If they match, the word is stored in the MAU CR.

If any of the operand specifiers in the command word indicate that an operand is to be obtained from memory, the MAU waits until the proper number of coprocessor data fetch bus transactions occur. The words are stored in internal registers.

The MAU performs the operation, generating a result, new condition codes and, possibly, an exception. The $\overline{\text{DONE}}$ pin is asserted and the MAU waits until a coprocessor status fetch access is seen. If an exception is present, the MAU faults the access and returns to the quiescent state. If there is no exception, a word

containing the current auxiliary status register (ASR) value is returned. If the result is to be placed in memory, the MAU waits until the proper number of coprocessor data write bus transactions occur, putting a word of the result on the bus as each transaction occurs. The MAU then returns to the quiescent state.

Peripheral Mode

In the peripheral mode protocol, the MAU registers appear as memory-mapped locations and are accessed via normal read and write operations. The CPU (or any other bus master) starts an MAU transaction by writing a word into the MAU command register. When the write access is completed, the MAU clears the result available (RA) bit in the ASR and negates the $\overline{\text{DONE}}$ pin. The command is then executed by using the operands available in registers F0 through F3 or by reading values via the data register (DR).

If any of the operands are to come from memory, the bus master must write the operands into the DR one word at a time after the CR has been written. These operands are latched into internal registers. If the result is to be placed in memory, the bus master must read the result from the DR one word at a time and transfer the words to memory. The MAU then updates the ASR contents to reflect the new condition codes, exception bits, and any other bits resulting from this operation. The RA bit in the ASR is set (1) and the MAU enters the quiescent state.

Data Types and Formats

The *WE* 32106 Math Acceleration Unit (MAU) supports single, double, double-extended, decimal, and integer data formats.

The integer format provides a binary representation of signed integers in the 32-bit 2's complement format.

Single-, double-, and double-extended-precision formats are available for representing floating-point numbers. The single-precision format provides an 8-bit exponent and an exponent bias, which allows the reciprocal of all normalized numbers to be represented without overflow.

With double-precision, the exponent range is sufficient to ensure that the product of eight 32-bit terms cannot overflow the 64-bit format.

Double-extended-precision provides a format with a range and precision that is greater than double-precision format but requires less than twice as many bits for representation. Having greater precision, double-extended-precision numbers lessen the chance that a final result will have been contaminated by excessive roundoff error; with greater range, the chance of an intermediate overflow aborting a computation whose result would have been representable in a basic format is lessened.

The decimal format provides a method for representing 18-digit signed decimal numbers in BCD (i.e., 18 BCD digits plus sign nibble).

Single-Precision

Single-precision operands are 32 bits long. The single-precision format is shown and described in Table 1. Table 2 shows how certain special-case values are represented when using the single-precision format.

Double-Precision

The double-precision operands are 64 bits long. The double-precision format is presented in Table 3. Table 4 shows how certain special-case values are represented when using the double-precision format.

Double-Extended-Precision

Double-extended-precision operands are 80 bits long. Table 5 describes the double-extended-precision format. Table 6 shows how certain special-case values are represented when using the double-extended-precision format.

Decimal

Decimal operands are 76 bits long. Table 7 shows the format for a decimal operand and describes each bit position. When doing float-to-decimal and decimal-to-float conversion operations, software support is needed to provide the range required by ANSI/IEEE Standard 754-1985 for these operations. Table 8 shows how certain special-case values are represented in decimal format.

Table 1. Single-Precision Format

Bit	31	30 23	22 0
Field	Sign	Exponent	Fraction

Bit	Field	Description
0—22	Fraction	**Fraction.** A 23-bit string that encodes the significant bits of the number. For normalized numbers, an implicit bit that has a value of 1 resides immediately to the left of the binary point. The implicit bit and fraction can represent values in the range 1 through $2-2^{-23}$.
23—30	Exponent	**Exponent.** These 8 bits represent an exponent biased by 127.
31	Sign	**Sign Bit.** A 1 represents a negative value; 0 represents a positive value.

Table 2. Single-Precision Special-Case Values

			Fraction Value	
Value Name	Sign Value	Exponent Value	Bit 22	Bits 0—21
Nontrapping NaN	x	max	1	x
Trapping NaN	x	max	0	nonzero
Positive infinity	0	max	min	
Negative infinity	1	max	min	
Positive zero	0	min	min	
Negative zero	1	min	min	
Denormalized number	x	min	nonzero	
Normalized number	x	notmm	x	

Note: x = don't care.

Table 3. Double-Precision Format

Bit	63	62 52	51 0
Field	Sign	Exponent	Fraction

Bit	Field	Description
0—51	Fraction	**Fraction.** A 52-bit string that encodes the significant bits of the number. For normalized numbers, an implicit bit that has a value of 1 resides immediately to the left of the binary point. The implicit bit and the fraction can represent values in the range 1 through $2-2^{-52}$.
52—62	Exponent	**Exponent.** These 11 bits represent an exponent biased by 1023.
63	Sign	**Sign Bit.** A 1 represents a negative value; 0 represents a positive value.

Table 4. Double-Precision Special-Case Values

Value Name	Sign Value	Exponent Value	Fraction Value	
			Bit 51	Bits 0—50
Nontrapping NaN	x	max	1	x
Trapping NaN	x	max	0	nonzero
Positive infinity	0	max	min	
Negative infinity	1	max	min	
Positive zero	0	min	min	
Negative zero	1	min	min	
Denormalized number	x	min	nonzero	
Normalized number	x	notmm	x	

Note: x = don't care.

Table 5. Double-Extended-Precision Format

Bit	95 80	79	78 64	63	62 0
Field	Unused	Sign	Exponent	J	Fraction

Bit	Field	Description
0—62	Fraction	**Fraction.** A 63-bit string that encodes the significant bits of the number.
63	J	**Explicit Bit.** This bit resides immediately to the left of the binary point. The fraction and explicit bit, together, can represent numbers in the range 0 through $2-2^{-63}$.
64—78	Exponent	**Exponent.** These 15 bits represent an exponent biased by 16383.
79	Sign	**Sign Bit.** A 1 represents a negative value; 0 represents a positive value.
80—95	Unused	**Unused.** These bits are ignored and overwritten with 0s.

Table 6. Double-Extended-Precision Special-Case Values

Value Name	Sign Value	Exponent Value	J Value	Fraction Value	
				Bit 62	Bits 0—61
Nontrapping NaN	x	max	x	1	x
Trapping NaN	x	max	x	0	nonzero
Positive infinity	0	max	0	min	
Negative infinity	1	max	0	min	
Positive zero	0	min	0	min	
Negative zero	1	min	0	min	
Denormalized number	x	min	0	nonzero	
Unnormalized number	x	notmm	0	x	
Normalized number	x	notmax	1	x	

Note: x = don't care.

Table 7. Decimal Format

Bit	95 76	75 72	71 68	67 64	63 60	59 56	55 52	51 48	47 44	43 40
Field	UNUSED	D17	D16	D15	D14	D13	D12	D11	D10	D9

Bit	39 36	35 32	31 28	27 24	23 20	19 16	15 12	11 8	7 4	3 0
Field	D8	D7	D6	D5	D4	D3	D2	D1	D0	Sign

Note: D in the field position indicates a digit (e.g., D5 is digit 5). Each digit is 4-bits wide.

Bit	Field	Description
0—3	Sign	**Sign Bits.** These 4 bits represent a positive or negative sign and also indicate if the remaining fields represent a number, an infinity, or NaN. These are interpreted as:

Bit 3,2,1,0	Meaning	Bit 3,2,1,0	Meaning
0000	Positive infinity	1000	Trapping NaN
0001	Negative infinity	1001	Trapping NaN
0010	Positive NaN	1010	Positive number
0011	Negative NaN	1011	Negative number
0100	Trapping NaN	1100	Positive number
0101	Trapping NaN	1101	Negative number
0110	Trapping NaN	1110	Positive number
0111	Trapping NaN	1111	Positive number

4—75	D0—D17	**Digits.** These 72 bits make up 18 decimal digits (each digit is 4-bits wide). The leftmost digit (D17) is the most significant.
76—95	Unused	**Unused.** These bits are ignored and overwritten with 0s.

Table 8. Decimal Format Special-Case Values

Value Name	Digit Values	Sign Value
Nontrapping NaN	x	0010 or 0011
Nontrapping NaN	1010,1011,1100,1101, 1110 or 1111	1010, 1011, 1100, 1101, 1110 or 1111
Trapping NaN	x	0100, 0101, 0110, 0111, 1000 or 1001
Positive infinity	x	0000
Negative infinity	x	0001
Number	0000 through 1001	1010, 1011, 1100, 1101 1110 or 1111

Note: x = don't care.

Integer

An integer number is represented as a 2's complement 32-bit operand. This representation is shown on Figure 2.

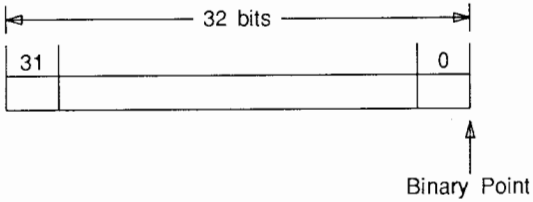


Figure 2. Integer Format

Registers

Auxiliary Status Register

The auxiliary status register (ASR) controls the remainder operation (partial remainder bit), signals the state of an operation (result available bit), disables and records exceptions (mask and sticky bits), controls rounding of results (round control bits), and records condition codes (negative and zero bits).

The negative, zero, inexact, and integer overflow bits in the ASR match the positions of the negative, zero, overflow, and carry bits (the condition codes) in the PSW register of the *WE* 32100 Microprocessor. This allows the bits to be copied into the PSW as part of the coprocessor status access and to be easily tested by CPU software.

Operand Registers

Each of the four operand registers (F0—F3) is 80-bits and contains one operand in extended format (see Table 10). The registers are shown as 96 bits long because they are read and written as three 32-bit words through the data register. In peripheral mode, bits 80 through 95 are ignored during writes and returned as 0s during read operations. Registers F0 through F3 are unchanged on reset; they are indeterminate on power-up.

Command Register

The command register (CR) accepts command words that are used to initiate an MAU transaction. The command register format is shown in Table 11.

Data Register

The data register (DR) is used to read and write operands via peripheral mode accesses. The DR appears as three 32-bit words in the peripheral mode address space. When an exception occurs, the information supplied to the trap handler is stored in the DR. This information is summarized in Table 12. An extract result on fault (EROF) instruction can be executed to retrieve this information from the DR.

Table 9. Auxiliary Status Register

Bit	20	19	18	17	16	15	14	13	12	11	10	9	8	7	6	5	4 0
Field	Z	IO	PS	CSC	UO	Unused	IM	OM	UM	QM	PM	IS	OS	US	QS	PR	Unused

Bit	Field	Contents	Description
0—4	Unused	Unused	These bits appear as 0 when read.
5	PR	Partial remainder	Set when result of a remainder operation is a partial remainder; cleared when result is a full remainder. This bit is cleared on reset.
6	QS	Divide by zero sticky	Set if the divisor is normalized zero and dividend is a finite nonzero number.
7	US	Underflow sticky	Set if exponent of a rounded result of an arithmetic operation is too small to be represented in the exponent field of the destination format.
8	OS	Overflow sticky	Set if exponent of a rounded result of an arithmetic operation is too large for the exponent field of the destination format.
9	IS	Invalid operation sticky	Set if a result cannot be stored in a destination legally or if illegal operands are given to some operation.
10	PM	Inexact mask	If this bit is set by the user, an exception occurs when bit 18 is set. If this bit is cleared, there are no inexact exceptions.
11	QM	Divide by zero mask	If this bit is set by the user, an exception occurs when bit 6 is set. If this bit is cleared, there are no divide by zero exceptions.
12	UM	Underflow mask	If this bit is set by the user, an exception occurs when bit 7 is set. If this bit is cleared, there are no underflow exceptions.
13	OM	Overflow mask	If this bit is set by the user, an exception occurs when bit 8 is set. If this bit is cleared, there are no overflow exceptions.
14	IM	Invalid operation mask	If this bit is set by the user, an exception occurs when bit 9 is set. If this bit is cleared, there are no invalid operation exceptions.
15	Unused	Unused	This bit appears as 0 when read.
16	UO	Unordered	Set when a compare operation results in an unordered indication; otherwise, it is cleared.
17	CSC	Context switch control	Set on every MAU instruction execution. This bit is cleared on reset.
18	PS	Inexact sticky	Set if the result cannot be specified in the destination format.
19	IO	Integer overflow	Set when a convert float-to-integer operation causes an overflow.
20	Z	Zero	Set if result of last operation is zero. If result is nonzero, bit is cleared.

Table 9. Auxiliary Status Register (Continued)

Bit	31	30 26	25	24	23 22	21
Field	RA	Unused	ECP	NTNC	RC	N

Bit	Field	Contents	Description
21	N	Negative	Set if result of the last operation is negative. If result is positive, bit is cleared. Note that it is possible for an operation to result in both bits 20 and 21 being set since negative zero is a representable number.
22,23	RC	Round control	Represents the round control mode. The code is interpreted as: **Bit 23,22 — Description** 00 — Round to nearest representable number 01 — Round towards plus infinity 10 — Round towards minus infinity 11 — Round towards zero (truncation)
24	NTNC	Nontrapping NaN control	This bit is tested when an invalid operation exception occurs and bit 14 is cleared. If this bit is set, an exception occurs and bit 9 is set. It is expected that software will write a virtual program counter value into the fraction portion of the nontrapping NaN generated.
25	ECP	Exception condition present	Set if any one of the floating-point exception conditions except inexact is present.
26—30	Unused	Unused	These bits appear as 0 when read.
31	RA	Result available	Cleared at beginning of an operation and set when result of an operation is available. During the quiescent state, this bit has a value of 1.

Table 10. Operand Registers

Bit	95 80	79	78 64	63	62 0
Field	Unused	Sign	Exponent	J	Fraction

Bit	Field	Contents	Description
0—62	Fraction	Fraction	These bits represent the fractional part of the number.
63	J	Explicit bit	The J bit resides to the left of the binary point in the 2^0 position. Together, the J bit and fraction field can represent values in the range 0 through $2-2^{-63}$.
64—78	Exponent	Exponent	The exponent field contains an exponent that is biased by 16383.
79	Sign	Sign	A 1 represents a negative value; 0 represents a positive value.
80—95	Unused	—	—

Table 11. Command Register

Bit	31 24	23 15	14 10	9 7	6 4	3 0
Field	ID	Unused	Opcode	OP1	OP2	OP3

Bit	Field	Contents	Description
0—3	Op3	Operand specifier	Specifies whether destination operand is an MAU register, a memory-based operand of a given size, or nonexistent (no operand). Even though the register destinations are specified as single, double, or double-extended, the result is stored in the registers in double-extended precision. The precision designations are used for rounding and checking for underflow and overflow. The value of this field is interpreted as:

Bit 3,2,1,0	Operand Register	Register Precision	Bit 3,2,1,0	Operand Register	Register Precision
0000	F0	Single	1000	F0	Double-extended
0001	F1	Single	1001	F1	Double-extended
0010	F2	Single	1010	F2	Double-extended
0011	F3	Single	1011	F3	Double-extended
0100	F0	Double	1100	DR	Memory-based single-word
0101	F1	Double	1101	DR	Memory-based double-word
0110	F2	Double	1110	DR	Memory-based triple-word
0111	F3	Double	1111	None	No operand

Bit	Field	Contents	Description
4—6	Op2	Operand specifier	Specifies whether second source operand is an MAU register, a memory-based operand of a given size, or nonexistent (no operand). The value of this field is interpreted as:

Value Bit 6,5,4	Operand Location
000	Register F0
001	Register F1
010	Register F2
011	Register F3
100	Memory-based single-word
101	Memory-based double-word
110	Memory-based triple-word
111	No operand

Bit	Field	Contents	Description
7—9	Op1	Operand specifier	Specifies whether first source operand is an MAU register, a memory-based operand of a given size, or nonexistent (no operand). The value of this field is interpreted as:

Value Bit 9,8,7	Operand Location
000	Register F0
001	Register F1
010	Register F2
011	Register F3
100	Memory-based single-word
101	Memory-based double-word
110	Memory-based triple-word
111	No operand

Table 11. Command Register (Continued)

Bit	Field	Contents	Description
10—14	Opcode	Operation code	Specifies operation to be performed.
15—23	Unused	—	These bits are returned as 0 when read.
24—31	ID	Processor ID number	Specifies identification number of the processor that should react to the command word. The MAU ID is 0.

Table 12. Data Register Trap Handler Information

Exception Type	Information Supplied
Invalid operation	If either of the source operands is a trapping NaN, the DR contains the NaN, converted to double-extended precision (80 bits) if necessary. If both source operands are trapping NaNs or infinities of different signs, the second operand (Op2) in double-extended precision (80 bits) is stored in the DR.
Overflow or underflow	The significand, along with the 17-bit internal result exponent and result sign, is stored in the DR. The most significant bit of the 17-bit exponent behaves like a sign bit in 2s complement notation. Two additional bits (bits 79 and 80) in the exponent ensure that no significant exponent bits are lost from an internal representation when an overflow or underflow condition occurs. The exponent value is biased by 16383. The format is:
Divide by zero	The dividend (Op2) converted to double-extended precision, if necessary, is stored in the DR.
Inexact	The rounded result converted to double-extended precision, if necessary, is stored in the DR.*

Bit	81	80 64	63	62 0
Field	Sign	Exponent	J	Fraction
			SIGNIFICAND	

* If operating in the peripheral mode and an inexact exception occurs, the DR contains the result in the original operation's destination precision, instead of always in double-extended precision.

Instruction Set Summary by Mnemonic and Opcode

The *WE* 32106 Math Acceleration Unit (MAU) instruction set provides arithmetic, logical, data transfer, and conversion functions. Table 13 lists the instruction set mnemonics alphabetically; Table 14 lists the instruction set opcodes numerically.

The *WE* 32100 Microprocessor allows up to two memory-based operands for each coprocessor protocol instruction. The MAU allows up to three memory-based operands so that it can be used with a CPU that supports three memory-based operand operations.

Unassigned opcodes cause behavior identical to that of the no operation (NOP) opcode.

Table 13. Instruction Set Summary by Mnemonic

Mnemonic	Opcode	Instruction	Operation	Condition Codes Affected
ABS	0x0C	Absolute value	Clears sign bit of operand 1 and copies operand 1 into operand 3. Operand 1 is unaffected.	N,Z,IO,PR,UO
ADD	0x02	Add	Adds operand 1 to operand 2 and stores result in operand 3.	N,Z,IO,PR,UO
CMP	0x0A	Compare	Compares operand 1 to operand 2 and sets condition codes. Operand 1 and operand 2 are not modified.	N,Z,IO,PR,UO
CMPE	0x0B	Compare with exceptions	Compares operand 1 to operand 2 and sets condition codes. Operand 1 and operand 2 are not modified.	N,Z,IO,PR,UO
CMPES	0x1B	Compare with exceptions and flags swapped	Compares operand 1 to operand 2 and sets condition codes. Operand 1 and operand 2 are not modified.	N,Z,IO,PR,UO
CMPS	0x1A	Compare with flags swapped	Compares operand 1 to operand 2 and sets condition codes. Operand 1 and operand 2 are not modified.	N,Z,IO,PR,UO
DIV	0x04	Divide	Divides operand 2 by operand 1 and stores result in operand 3.	N,Z,IO,PR,UO
DTOF	0x11	Convert decimal to floating-point	Converts operand 1 (in decimal format) to floating-point format and then copies it into operand 3. Operand 1 is unaffected.	N,Z,IO,PR,UO
EROF	0x14	Extract result on fault	Places value of data register in operand 3. **Note:** Execute this instruction only when a fault condition occurs: if this instruction is executed under normal conditions, the value returned is indeterminate.	None
FTOD	0x12	Convert floating-point to decimal	Rounds operand 1 to an integral value and then converts it into decimal format and copies it into operand 3. Operand 1 is unaffected.	N,Z,IO,PR,UO
FTOI	0x0F	Convert floating-point to integer	Converts operand 1 from floating-point to 2's complement integer and copies it into operand 3 (single-word only).	N,Z,IO,PR,UO
ITOF	0x10	Convert integer to floating-point	Converts operand 1 from 2's complement integer format to floating-point and copies it into operand 3.	N,Z,IO,PR,UO
LDR	0x18	Load data register	Places operand 1 in data register.	Undefined

Table 13. Instruction Set Summary by Mnemonic (Continued)

Mnemonic	Opcode	Instruction	Operation	Condition Codes Affected
MOVE	0x07	Move	Copies operand 1 into operand 3. Operand 1 is unaffected.	N,Z,IO,PR,UO
MUL	0x06	Multiply	Multiplies operand 1 by operand 2 and stores result in operand 3.	N,Z,IO,PR,UO
NEG	0x17	Negate	Complements sign of operand 1 and copies operand 1 into operand 3. Operand 1 is unaffected.	N,Z,IO,PR,UO
NOP	0x13	No operation	Causes any operation in progress to terminate without affecting any of the MAU internal registers.	None
RDASR	0x08	Move from ASR	Copies ASR into operand 3. The ASR is unaffected.	None
REM	0x05	Remainder	Divides operand 2 by operand 1 and stores remainder in operand 3.	N,Z,IO,PR,UO
RTOI	0x0E	Round to integral value	Rounds operand 1 to an integral value in floating-point format and then copies it into operand 3. Operand 1 is unaffected.	N,Z,IO,PR,UO
SQRT	0x0D	Square root	Computes square root of operand 1 and stores in operand 3. Operand 1 is unaffected.	N,Z,IO,PR,UO
SUB	0x03	Subtract	Subtracts operand 1 from operand 2 and stores result in operand 3.	N,Z,IO,PR,UO
WRASR	0x09	Move to ASR	Copies operand 1 into ASR. Operand 1 is unaffected.	N,Z,IO,PR,UO

Table 14. Instruction Set Summary by Opcode

Opcode	Mnemonic	Instruction
0x02	ADD	Add
0x03	SUB	Subtract
0x04	DIV	Divide
0x05	REM	Remainder
0x06	MUL	Multiply
0x07	MOVE	Move
0x08	RDASR	Move from ASR
0x09	WRASR	Move to ASR
0x0A	CMP	Compare
0x0B	CMPE	Compare with exceptions
0x0C	ABS	Absolute value
0x0D	SQRT	Square root
0x0E	RTOI	Round to integral value

Opcode	Mnemonic	Instruction
0x0F	FTOI	Convert floating-point to integer
0x10	ITOF	Convert integer to floating-point
0x11	DTOF	Convert decimal floating-point
0x12	FTOD	Convert floating-point to decimal
0x13	NOP	No operation
0x14	EROF	Extract result on fault
0x17	NEG	Negate
0x18	LDR	Load data register
0x1A	CMPS	Compare with flags swapped
0x1B	CMPES	Compare with exceptions and flags swapped

Instruction Set Summary by Functional Groupings

Arithmetic Group

These instructions perform arithmetic operations on data.

Table 15. Arithmetic Group

Instruction	Mnemonic	Opcode
Absolute Value	ABS	0x0C
Add	ADD	0x02
Subtract	SUB	0x03
Multiply	MUL	0x06
Divide	DIV	0x04
Remainder	REM	0x05
Square root	SQRT	0x0D
Negate	NEG	0x17
Round to integral value	RTOI	0x0E

Data Transfer Group

These instructions transfer data to and from registers and memory.

Table 16. Data Transfer Group

Instruction	Mnemonic	Opcode
Move	MOVE	0x07
Move from ASR	RDASR	0x08
Move to ASR	WRASR	0x09
Load DR	LDR	0x18

Logical Group

These instructions perform logical operations on data.

Table 17. Logical Group

Instruction	Mnemonic	Opcode
Compare	CMP	0x0A
Compare with exceptions	CMPE	0x0B
Compare with flags swapped	CMPS	0x1A
Compare with exceptions and flags swapped	CMPES	0x1B

Conversion Group

These instructions perform symmetric decimal, floating-point, and integer conversions.

Table 18. Conversion Group

Instruction	Mnemonic	Opcode
Decimal to floating-point	DTOF	0x11
Floating-point to decimal	FTOD	0x12
Floating-point to integer	FTOI	0x0F
Integer to floating-point	ITOF	0x10

Miscellaneous Group

These instructions do not fall into any of the previous categories.

Table 19. Miscellaneous Group

Instruction	Mnemonic	Opcode
Extract result on fault	EROF	0x14
No operation	NOP	0x13

Rounding

The MAU performs rounding at the end of every arithmetic operation; rounding is not a separate operation triggered via an opcode in the command word.

Rounding is controlled by bits 22 and 23 of the ASR. By changing the value of these bits, results can be rounded to the nearest representable number, towards positive infinity, towards minus infinity, or towards zero (truncation). See Table 9 for a description of the round control bits (bits 22 and 23).

Exceptions

When peripheral mode protocol is used, the MAU neither signals nor reacts to any exceptions on the system bus. If an operation results in an exception, the exception is signaled to the CPU by setting the appropriate ASR bits.

With coprocessor protocol, if an exception occurs as the result of an operation, the MAU indicates an exception during the coprocessor status fetch access after the ASR contents have been updated to reflect the result of the operation. If a coprocessor data fetch or a coprocessor data write access is not

completed successfully, the MAU waits until the access is completed.

If a coprocessor broadcast access occurs during an MAU transaction, the MAU aborts the current transaction. If the ID associated with the access is not the MAU ID, the MAU enters the quiescent state. If the ID is equal to the MAU ID, the MAU starts the transaction specified by the access.

The bus master that initiates a coprocessor broadcast access also detects when no coprocessor reacts to the access.

For each exception type, there is a mask bit and a sticky bit in the ASR. If the condition associated with the exception type is satisfied, the MAU sets the sticky bit for that exception type. If the mask bit is 1, an exception occurs. If the mask bit is 0, no exception occurs.

The MAU never changes the mask bits or clears the sticky bits. These actions, if necessary, must be done with software writes to the ASR.

The exceptions follow an order of precedence so that the highest priority exception is taken if more than one occurs. The ordering from highest priority to lowest priority is:

- Invalid operation (highest priority)
- Divide by zero, overflow, and underflow
- Inexact (lowest priority).

Interrupts

Once a transaction has been initiated via a peripheral mode write to the CR, the MAU ignores any interrupt acknowledge or autovector interrupt acknowledge bus cycle. This continues until the next MAU transaction is started.

In coprocessor protocol use, once an operation has been initiated via a coprocessor broadcast access, the MAU responds to any interrupt acknowledge or autovector interrupt acknowledge bus cycle by aborting the transaction and returning to the quiescent state.

Restart Capability

During peripheral mode accesses, reads and writes always terminate normally, with no fault indication. However, the CPU may read the DR successfully and then encounter a fault when writing to memory. In such cases, the CPU may then restart the instruction and thus cause another read to the DR.

When coprocessor protocol is used, the only time operations cannot be restarted is when a double-word or triple-word result is to be stored in memory at a location that is identical to or that overlaps that of a source operand. If one or more words of the result are stored successfully and a subsequent store is faulted, the MAU aborts the operation, leaving one of the source operands corrupted. This can be prevented by avoiding operations in which the result operand overlaps one of the source operands. It can also be prevented by ensuring that multiple-word result operands cannot be faulted on the second or third words.

Peripheral Mode Accesses

In peripheral mode, objects in the MAU can be accessed as memory-mapped addresses. A peripheral mode access occurs when there is a data read or data write access and the MAU chip select input is being asserted. The MAU latches the address associated with the access and interprets it as shown in Table 20. If the access is a read, the contents of the selected MAU object are returned as data. If the access is a write, the data associated with the access are stored in the MAU object selected.

The ASR and DR are readable and writable in peripheral mode. The CR can be written in peripheral mode, but not read. The operand registers are not directly accessible in peripheral mode.

All peripheral mode accesses are treated as word accesses by the MAU. If the *WE* 32100 CPU is used to perform byte or halfword reads from the MAU in peripheral mode, these accesses are completed correctly because the CPU ignores the extra bytes returned by the MAU. If the CPU is used to perform byte or halfword writes to the MAU, these accesses terminated normally, but a full word of data is written into the MAU. The specified byte or halfword is written correctly, but the rest of the data written may take on any value.

Operating System Considerations

Peripheral Mode Protocol

When it uses peripheral mode protocol, the MAU does not abort its operation when an

Table 20. Peripheral Mode Address Fields

Bit	31 — 5	4 — 2	1 — 0
Field	Unused	Object	Unused

Bit	Field	Description
0—1	Unused	Ignored by the MAU.
2—4	Object	Selects which object is to be accessed as per object value specified. (Bit 2 is the least significant bit.)
5—31	Unused	Ignored by the MAU.

Object Bit 4 3 2	Object Addressed
0 0 0	ASR
0 0 1	ASR
0 1 0	ASR
0 1 1	ASR
1 0 0	DR bits 95—64
1 0 1	DR bits 63—32
1 1 0	DR bits 31—0
1 1 1	CR

interrupt acknowledge cycle occurs. The peripheral mode protocol assumes that the CPU is executing a stream of discrete instructions to implement the protocol, and thus is not able to back up the first instruction in the event of an interrupt. The CPU is free to allow or disallow interrupts during MAU operations; however, MAU context saving is simplified if the interrupts can be disabled.

The state of the MAU must be saved and restored later if an interrupt causes control of the MAU to be given to another process. If the MAU is in the middle of an operation and the RA bit of the ASR is cleared, the MAU's state cannot be saved and restored without explicit software provisions. If the first CPU instruction of the sequence can be found, the context of the partially executed transaction can be saved so that execution can resume there. The context to be saved and restored consists of the ASR, DR, and F0—F3. It may be necessary to run the context save and restore sequences with interrupts disabled.

If an interrupt occurs and the RA bit of the ASR has been set, the MAU is placed in the quiescent state. The transaction may or may not have completed (i.e., some reads from the DR may not be complete). The context to be saved and restored consists of the ASR, DR, and F0—F3. (See Context Saving and Restoring.)

Coprocessor Mode Protocol

If an interrupt occurs during an MAU transaction before the $\overline{\text{DONE}}$ signal goes to 0, the MAU aborts the operation. In such cases, it is possible that the ASR has been changed to reflect exceptions or new condition codes; however, when the CPU returns from the interrupt, it can restart the transaction from the beginning, and the spurious ASR value does not affect the repeated operation.

The context of the MAU must be saved and restored later if the interrupt causes control of the MAU to be given to another process. The context to be saved and restored consists of the ASR, DR, and F0—F3.

Context Saving and Restoring

Peripheral Mode

The MAU context can be saved by:

- Reading the ASR via peripheral mode and saving its value.
- Clearing the ASR via peripheral mode.
- Reading the three DR words via peripheral mode and saving their value.

- Reading registers F0—F3 via MOVE instructions and saving their values in memory locations. The values appear in the DR and should be moved to memory via peripheral mode accesses.

The MAU context can be restored by:

- Performing four MOVE operations to restore registers F0—F3. The values must be written to the DR via peripheral mode accesses.
- Performing an MAU MOVE operation when the source is a memory location (the value must be written into the DR via a peripheral mode access) and the destination is a memory location (the value will appear in the DR). This restores the DR.
- Writing the saved ASR value into the ASR via peripheral mode.

Coprocessor Mode

The MAU context can be saved by:

- Reading DR with an extract result on fault (EROF) instruction and saving its value.
- Reading the ASR via peripheral mode or reading the ASR with the RDASR instruction and saving its value.
- Clearing the ASR.
- Performing four MAU MOVE operations to save registers F0—F3.

The MAU context can be restored by:

- Performing four MAU MOVE operations to restore registers F0—F3.
- Writing the saved ASR value into the ASR via peripheral mode or the WRASR instruction.
- Performing a load DR operation using the LDR instruction or, in peripheral mode, using the MOVE instruction to restore DR.

System Configuration Restrictions

Peripheral Mode Protocol

If non-addressed devices assert bus exceptions, incorrect data may be written into the MAU during writes.

Coprocessor Mode Protocol

The MAU ID number is fixed, so it is not possible to have more than one MAU on the same bus in coprocessor mode.

It is not possible to have more than one coprocessor protocol transaction in progress at any one time. If an MAU and several other coprocessors exist on one bus, only one transaction can be in progress at any one time.

MAU-CPU Interconnections

Several MAU signals are interfaced directly to the CPU. These signals are shown on Figures 3 and 4.

Definitions

Biased exponent. The sum of the exponent and a constant (bias) chosen to make the biased exponent's range nonnegative.

Binary floating-point number. A bit string characterized by three components: a sign, a signed exponent, and a significand. Its numerical value, if any, is the signed product of its significand and two raised to the power of its exponent.

Denormalized. The exponent is the format's minimum, the explicit or implicit bit is zero, and the number is not normal zero. To denormalize a double-extended precision binary floating-point number means to shift its significand to the right while incrementing its exponent to zero.

Exponent. That component of a binary floating-point number that normally signifies the power to which two is raised in determining the value of the represented number. Occasionally, the exponent is called the signed or unbiased exponent.

Format's maximum (max). The value of a field in which all bits in the field are 1.

Format's minimum (min). The value of a field in which all bits in the field are 0.

Fraction. The field of the significand that lies to the right of its implied binary point.

NaN. Not a number; a symbolic entity encoded in floating-point format. Operations that have no mathematical interpretation, such as zero

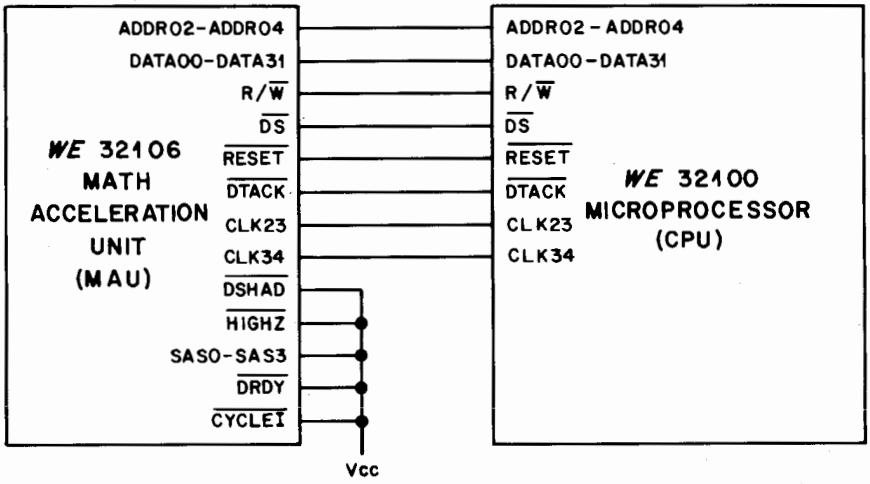
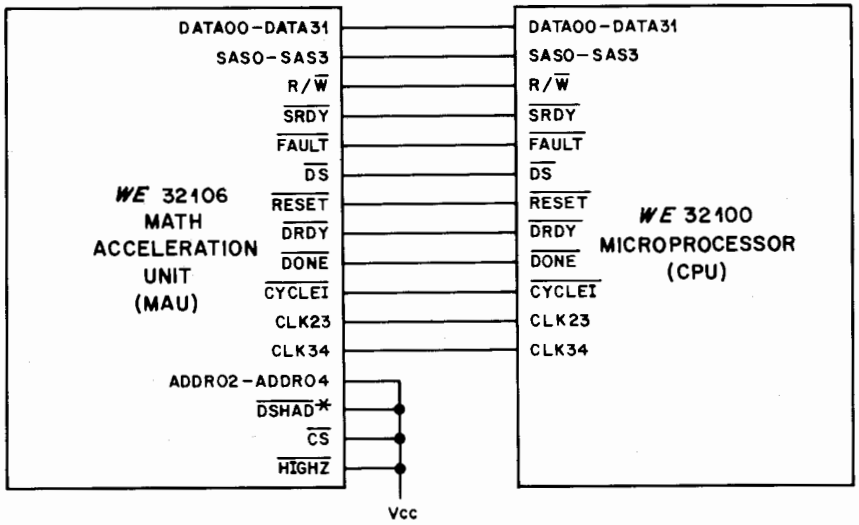


Figure 3. MAU-CPU Peripheral Mode Protocol Interconnections



*If an MMU is included in the system, this signal must be connected to the MMU's $\overline{\text{DSHAD}}$ output.

Figure 4. MAU-CPU Coprocessor Mode Protocol Interconnections

divided by zero, will produce an NaN. Such NaNs can be used to convey diagnostic information regarding their creation.

Neither max nor min (notmm). The value of a field in which all bits are neither all 1s nor all 0s.

Normal zero. The exponent and the significand are the format's minimum. Normal zero may have either a positive or a negative sign. Only the extended format has unnormalized zeros.

Normalized. If the number is nonzero, its significand shifts to the left while its exponent decrements until the leading significand bit becomes 1; the exponent is regarded as if its range were unlimited. If the significand is zero, the number becomes normal zero. Normalizing a number does not change its sign.

Quiescent state. The MAU accepts and reacts to data read and write if chip-select is asserted and the coprocessor does broadcast accesses only. It accepts resets, does not read from or write to any internal registers, and does not affect the rest of the system in any way.

Significand. The component of a binary floating-point number that consists of an explicit or implicit leading bit to the left of its binary point and a fraction field to the right of the binary point.

Unnormalized. Occurs only in the double-extended format. The number's exponent is greater than the format's minimum and the explicit leading bit is zero. If the significand is zero this is an unnormalized zero.

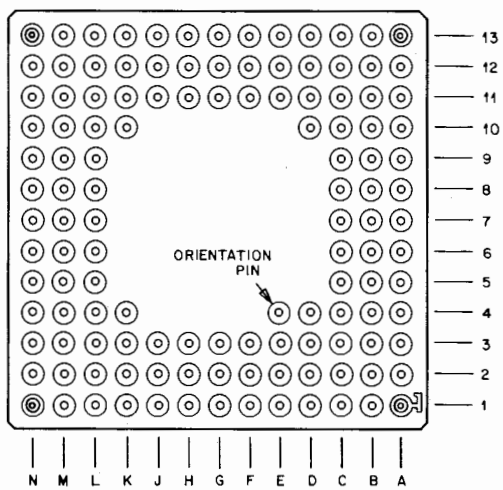
x. Don't care condition.

Pin Descriptions

The *WE* 32106 Math Acceleration Unit is available in 125-pin square, ceramic PGA package. Figure 5 and Tables 21—27 describe the pin assignments.

The term *asserted* in the pin function descriptions means that the signal is driven to its active state either by the MAU (outputs) or by an external device (inputs). The term *negated* means that the signal is driven to its inactive state. A bar over the signal name (e.g., $\overline{DS}$) indicates the signal is active low, logic 0. The 0 bit is the least significant bit for signals which occupy two or more pins (e.g., SAS0). In the signal type column, I indicates an input, O an output, and I/O a bidirectional signal.

Table 21 lists signals in numerical order for the 125-pin square PGA package. Tables 22 through 27 describe signals by functional group.



Bottom View

Figure 5. WE® 32106 MAU 125-Pin Configuration

Numerical Order

Table 21. Pin Decriptions – 125-Pin Square PGA Package

Pin	Name	Type	Description
A1	DATA10	I/O	Data 10
A2	DATA13	I/O	Data 13
A4	DATA16	I/O	Data 16
A6	DATA18	I/O	Data 18
A8	DATA21	I/O	Data 21
A10	DATA24	I/O	Data 24
A11	DATA26	I/O	Data 26
A13	$\overline{\text{SRDY}}$	O	Synchronous ready
B2	DATA11	I/O	Data 11
B3	DATA15	I/O	Data 15
B5	DATA17	I/O	Data 17
B7	DATA19	I/O	Data 19
B8	DATA20	I/O	Data 20
B9	DATA22	I/O	Data 22
B10	DATA23	I/O	Data 23
B11	DATA25	I/O	Data 25
B12	DATA30	I/O	Data 30
C1	DATA09	I/O	Data 09
C3	DATA12	I/O	Data 12
C4	+5V	—	Power
C5	GND	—	Ground
C6	+5V	—	Power
C7	GND	—	Ground
C8	+5V	—	Power
C9	GND	—	Ground
C10	+5V	—	Power
C11	DATA27	I/O	Data 27
C13	DATA28	I/O	Data 28
D2	DATA08	I/O	Data 08
D4	DATA14	I/O	Data 14
D11	+5V	—	Power
D12	DATA29	I/O	Data 29
E1	DATA07	I/O	Data 07
E3	GND	—	Ground
E4	—	—	Device socket orientation pin
E11	GND	—	Ground
E13	DATA31	I/O	Data 31
F2	DATA05	I/O	Data 05
F3	+5V	—	Power
F11	+5V	—	Power
F12	CLK23	I	Input clock 23
G1	DATA06	I/O	Data 06
G3	GND	—	Ground
G11	GND	—	Ground
G12	CLK34	I	Input clock 34
H2	DATA02	I/O	Data 02
H3	+5V	—	Power
H11	+5V	—	Power
H12	$\overline{\text{DONE}}$	O	Coprocessor done
J1	DATA04	I/O	Data 04
J3	GND	—	Ground
J11	GND	—	Ground
J13	$\overline{\text{FAULT}}$	O	Fault
K2	DATA01	I/O	Data 01
K3	+5V	—	Power
K12	ADDR03	I	Address 03
L1	DATA03	I/O	Data 03
L3	DATA00	I/O	Data 00
L4	NC	—	No connection
L5	GND	—	Ground
L6	+5V	—	Power
L7	GND	—	Ground
L8	+5V	—	Power
L9	GND	—	Ground
L10	+5V	—	Power
L11	ADDR04	I	Address 04
M2	$\overline{\text{DSHAD}}$	I	Data bus shadow
M4	SAS2	I	Access status code 2
M6	SAS1	I	Access status code 1
M7	$\overline{\text{DS}}$	I	Data strobe
M8	$\overline{\text{RESET}}$	I	Reset
M10	$\overline{\text{DTACK}}$	O	Data transfer acknowledge
M11	$\overline{\text{DRDY}}$	I	Data ready
M12	R/$\overline{\text{W}}$	I	Read/write
N1	$\overline{\text{HIGHZ}}$	I	High imppedance
N3	$\overline{\text{CYCLEI}}$	I	Cycle initiate
N5	SAS3	I	Access status code 3
N6	SAS0	I	Access status code 0
N8	$\overline{\text{CS}}$	I	Chip select
N9	NC	—	No connection
N11	NC	—	No connection
N13	ADDR02	I	Address 02

Functional Group

Table 22. Address and Data Signals

Name	Pin	Type	Description
DATA00–DATA31	L3, K2, H2, L1, J1, F2, G1, E1, D2, C1, A1, B2, C3, A2, D4, B3, A4, B5, A6, B7, B8, A8, B9, B10, A10, B11, A11, C11, C13, D12, B12, E13	I/O	**Data.** These pins provide a bidirectional bus to transmit data to and from the MAU.
ADDR02–ADDR04	N13, K12, L11	I	**Address.** These bits are used to select the peripheral mode registers when MAU is in peripheral mode.

Table 23. Interface and Control

Name	Pin	Type	Description
$\overline{CS}$	N8	I	**Chip Select.** Assertion puts MAU in peripheral mode.
$\overline{CYCLEI}$	N3	I	**Cycle Initiate.** Assertion indicates that CPU has started a bus cycle.
$\overline{DONE}$	H12	O	**Done.** Assertion indicates to CPU that MAU has completed execution of the current instruction.
$\overline{DRDY}$	M11	I	**Data Ready.** Assertion indicates to MAU that no bus exceptions have been detected during current bus cycle.
$\overline{DS}$	M7	I	**Data Strobe.** When low during a read operation, this signal indicates that MAU may place data on data bus. When low during a write operation, this signal indicates that MAU may remove data from data bus.
$\overline{DSHAD}$	M2	I	**Data Bus Shadow.** Assertion causes the MAU to 3-state the data bus if it was driving it.
$\overline{DTACK}$	M10	O	**Data Transfer Acknowledge.** Assertion signals CPU to end current access. Used for asynchronous handshaking between CPU and MAU in peripheral mode.
$\overline{SRDY}$	A13	O	**Synchronous Ready.** Assertion causes CPU to begin termination of either a coprocessor broadcast or a status fetch.

Table 24. Status Signals

Name	Pin	Type	Description
R/$\overline{W}$	M12	I	**Read/Write.** A 1 indicates a read from MAU; a 0 indicates a write to the MAU.
SAS0—SAS3	N6, M6, M4, N5	I	**Access Status Codes.** These pins describe the type of bus cycle being executed. SAS0 is the least significant bit of the access status code.

Bit 3,2,1,0	Description
0001	Coprocessor data write
0010	Autovector interrupt acknowledge
0011	Coprocessor data fetch
0101	Coprocessor broadcast
0110	Coprocessor status fetch
1011	Interrupt acknowledge

Table 25. Bus Exception Signals

Name	Pin	Type	Description
$\overline{FAULT}$	J13	O	**Fault.** Assertion indicates that an exception occured during execution of an instruction in the MAU.
$\overline{RESET}$	M8	I	**Reset.** Assertion causes MAU to execute its reset routine.

Table 26. Clocks

Name	Pin	Type	Description
CLK34	G12	I	**Input Clock.** The falling edge of this clock signifies the beginning of a machine cycle. This clock input has the same frequency as CLK23 and lags it by 90°.
CLK23	F12	I	**Input Clock.** This clock input has the same frequency as CLK34 and leads it by 90°.

Table 27. Test

Name	Pin	Type	Description
$\overline{HIGHZ}$	N1	I	**High Impedance.** 3-states all output signals for testing.

Characteristics

V_{CC} = 5.0 V ± 5%, V_{SS} = 0 V, C_L = 130 pF, T_A = 0 to 70 °C

Timing Characteristics

The low voltage threshold is 0.8 V; the high voltage threshold is 2.0 V. All TTL timing specifications are referenced to or from these threshold voltages. CMOS clock references are to and from $V_{CC}/2$. All timing information is subject to change. All min and max values are in ns.

Table 28. Timing Characteristics

Num	Symbol	Description	Fig(s)	10 MHz		14 MHz		18 MHz	
				Min	Max	Min	Max	Min	Max
1	tSASVC34L	Access status code set-up time*	8,9,10, 11,12	13	—	13	—	10	—
2	tC34LSASX	Access status code hold time	8,9,10, 11,12	20	—	20	—	16	—
3	tCYCLC23L	Cycle initiate set-up time	8,9,10, 11,12	11	—	11	—	7	—
4	tC23LCYCX	Cycle initiate hold time	8,9,10, 11,12	20	—	20	—	16	—
5	tDATVC23H	Data set-up time (ID test)	8	35	—	33	—	26	—
7	tDRYVC23H	Data ready set-up time	8,9,10, 11	13	—	13	—	10	—
8	tC23HDRYX	Data ready hold time	8,9,10, 11	20	—	20	—	16	—
9	tDATVC34H	Data set-up time	9,14	13	—	13	—	9	—
10	tC34HDATX	Data hold time	9,14	25	—	20	—	16	—
13	tCSLC23H	Chip select set-up time*	13,14	13	—	13	—	10	—
14	tDSBHCSX	Chip select hold time	13,14	0	—	0	—	0	—
15	tRWHC23H	Read/write strobe set-up time*	13,14	13	—	13	—	10	—
16	tDSBHRWX	Read/write strobe hold time	13,14	0	—	0	—	0	—
17A	tDSBLC23H	Data strobe set-up time	8, 9, 10, 11	13	—	13	—	10	—
17B	tDSBLC23H	Data strobe set-up time*	13,14	11	—	13	—	10	—
18	tC23HDSBX	Data strobe hold time	8,9,10, 11,13,14	20	—	20	—	16	—
19	tDSHLDATZ	Data 3-state time	12	—	75	—	50	—	39
20	tDSHHDATV	Data valid time	12	—	75	—	50	—	39
21	tRSRLC23L	Reset set-up time	—	15	—	13	—	10	—
22	tRSRLRSRH	Reset valid time (to guarantee reset)		8Tc	—	8Tc	—	8Tc	—
23	tADDVC23H	Address set-up time	13,14	24	—	15	—	12	—
24	tDSBHADDX	Address hold time	13,14	0	—	0	—	0	—

* This is an asynchronous signal. Set-up times are specified for testing and informational purposes.

Table 28. Timing Characteristics (Continued)

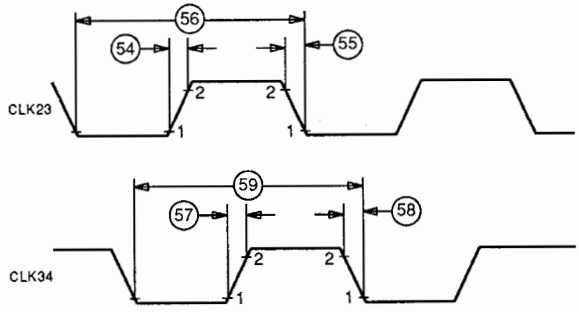
Num	Symbol	Description	Fig(s)	10 MHz		14 MHz		18 MHz	
				Min	Max	Min	Max	Min	Max
40	tC23HSRYL	Synchronous ready assertion time	8,10	—	48	—	32	—	25
41	tC23HSRYZ	Synchronous ready 3-state time	8,10	—	43	—	39	—	30
42	tC23HFATV	Fault assertion time	10	—	48	—	32	—	25
43	tC23HFATZ	Fault 3-state time	10	—	43	—	40	—	31
44	tC34HDONL	Done assertion time	10	—	48	—	32	—	25
45	tC34HDONZ	Done 3-state time	10	—	43	—	34	—	26
47	tC34HDATV	Data assertion time	10,11,12	—	103	—	69	—	54
48	tC34LDATZ	Data 3-state time	10,11	—	70	—	48	—	37
49	tHIZOUTZ	Outputs 3-state time	—	—	100	—	70	—	54
50	tC23LDTAL	Data acknowledge assertion time	13,14	—	45	—	32	—	25
51	tDSBHDTAZ	Data acknowledge 3-state time	13,14	—	55	—	40	—	31
52	tC23HDATV	Data assertion time	13	—	57	—	43	—	33
53	tDSBHDATZ	Data 3-state time	13	—	50	—	34	—	30
53A	tC23J	Clock 23 jitter time	—	—	0.5	—	0.5	—	0.5
53B	tC23E	Clock 23 duty cycle error time**	—	—	3	—	2	—	2
54	tC23L1C23HZ	Clock 23 rise time	7	—	4	—	4	—	4
55	tC23HZC23L1	Clock 23 fall time	7	—	4	—	4	—	4
56	tC23L1C23L1	Clock 23 period (nominal)	7	100	—	71.4	—	56	—
56A	tC34J	Clock 34 jitter time	—	—	0.5	—	0.5	—	0.5
56B	tC34E	Clock 34 duty cycle error time**	—	—	3	—	2	—	2
57	tC34L1C34HZ	Clock 34 rise time	7	—	4	—	4	—	4
58	tC34H2C34L1	Clock 34 fall time	7	—	4	—	4	—	4
59	tC34HC34H	Clock 34 period (nominal)	7	100	—	71.4	—	56	—
60	tSKEW	Clock skew**	—	—	3	—	2	—	2

* This is an asynchronous signal. Set-up times are specified for testing and informational purposes.

** At 70 °C ambient and Vcc = 4.75 V, the allowable clock skew and clock duty cycle errors are shown for each operating frequency. These skew and error specs do not increase by more than 1 ns per 30 C° and 1 ns per 250 mV away from this operating point and do not exceed the specified value throughout the full operating range.

Timing Diagrams

These timing diagrams represent a subset of all possible transactions and are intended only for the purposes of displaying and clarifying timing relationships.



Notes:
Duty Cycle Error – The duty cycle of each clock input may deviate from 50% but should not exceed timing specification numbers 53B and 56B for CLK23 and CLK34, respectively.
Skew – CLK23 nominally leads CLK34 by 90° (1/4 clock period). This phase lead should never exceed timing specification number 60.
Jitter – The period of each clock input may deviate from its nominal value but should not exceed timing specification numbers 53A and 56A for CLK23 and CLK34, respectively.

Figure 6. Clock Inputs

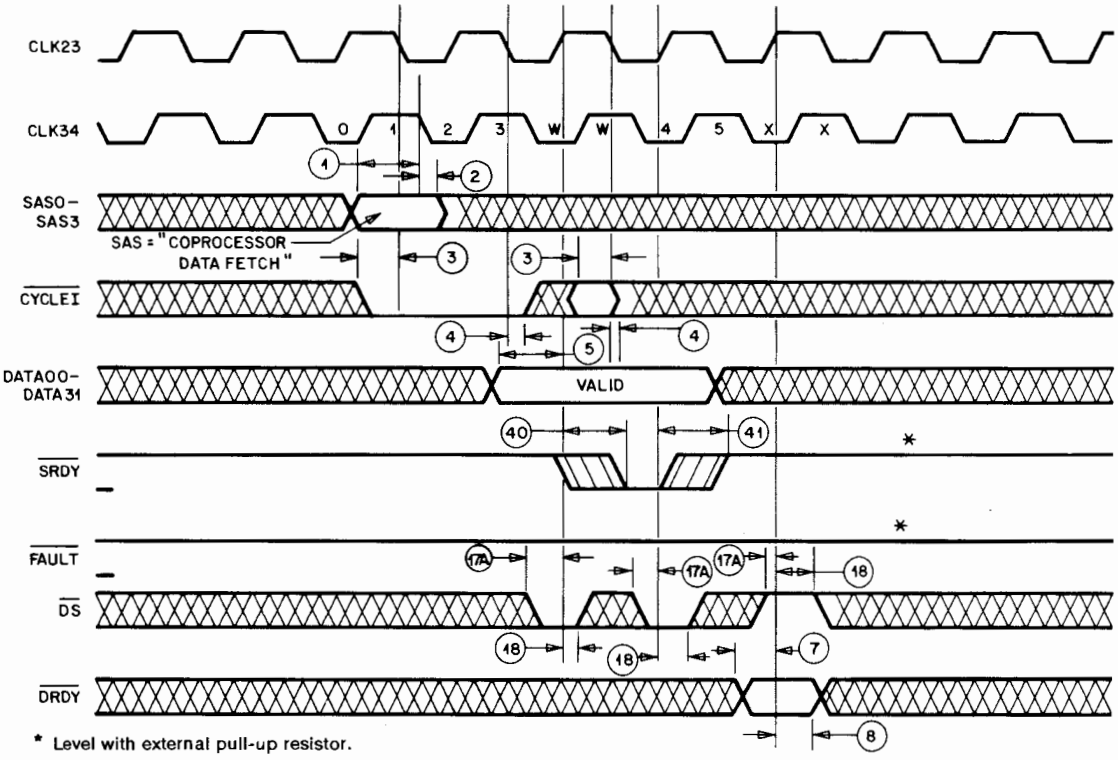
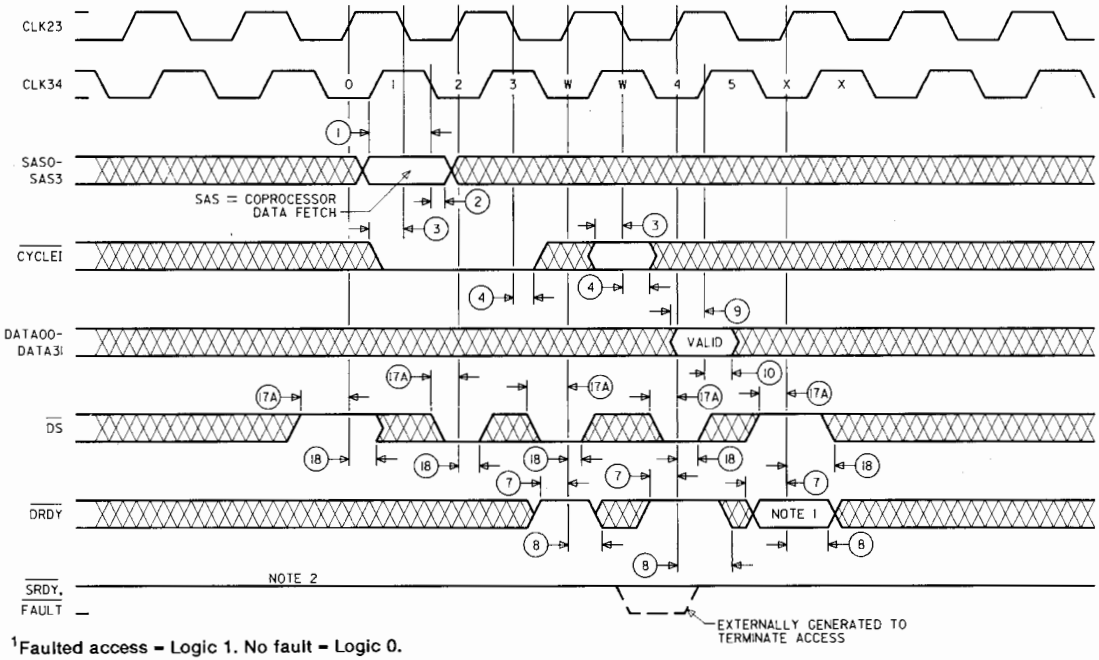


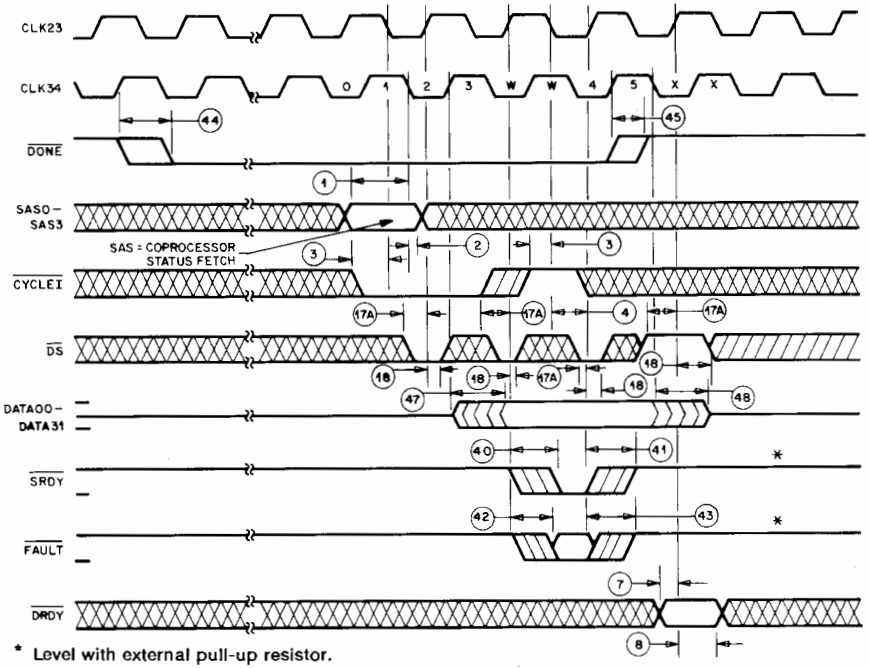
Figure 7. Coprocessor Broadcast (1 Wait-State)



[1]Faulted access = Logic 1. No fault = Logic 0.

[2]Level with external pull-up resistor.

Figure 8. Coprocessor Data Fetch (1 Wait-State)



* Level with external pull-up resistor.

Figure 9. Coprocessor Status Fetch

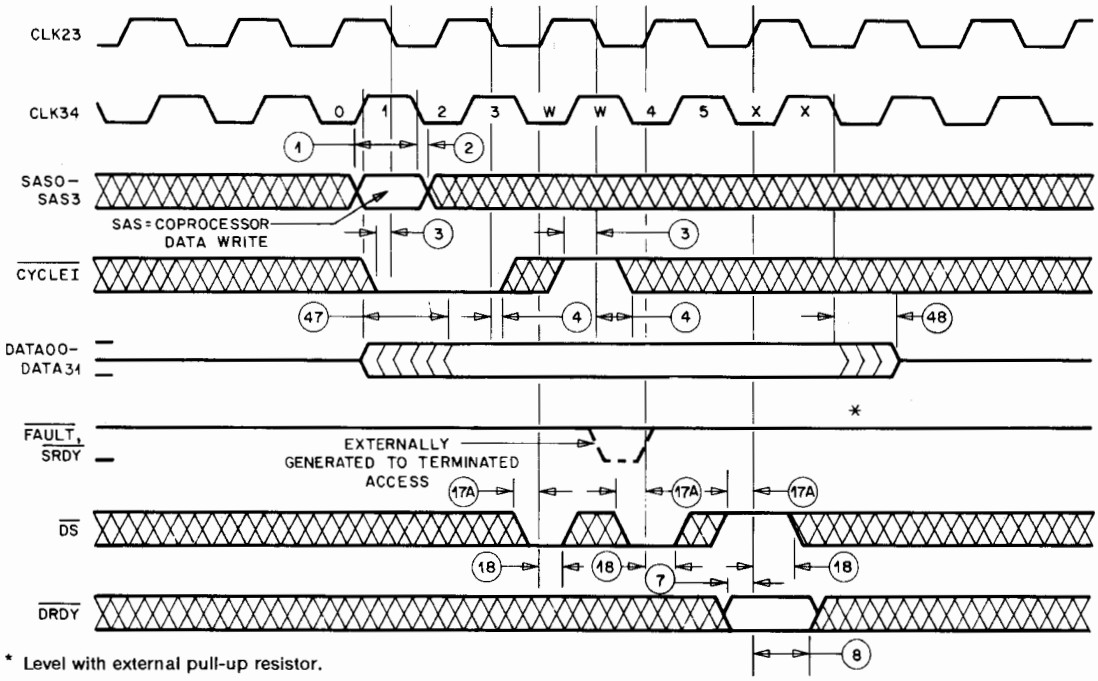


Figure 10. Coprocessor Data Write (1 Wait-State)

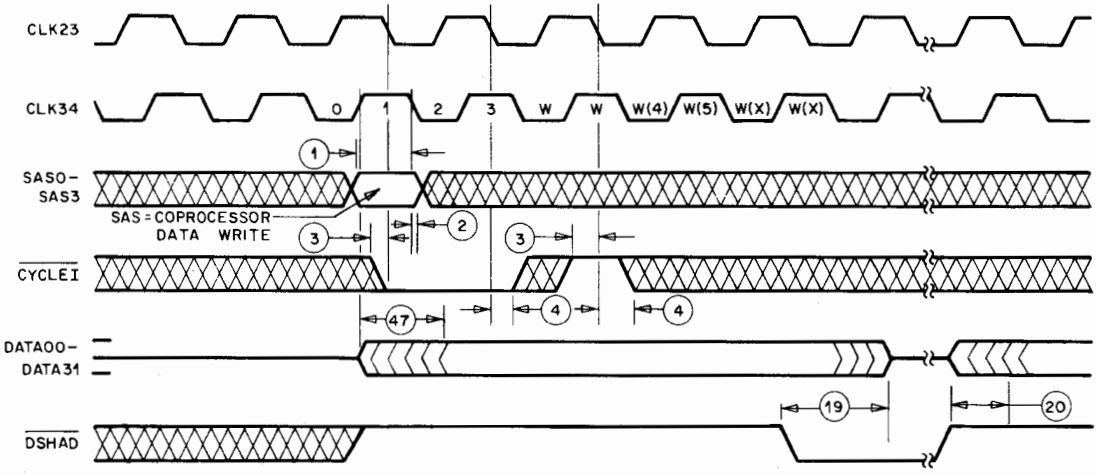


Figure 11. Coprocessor Data Write ($\overline{\text{DSHAD}}$ Assertion)

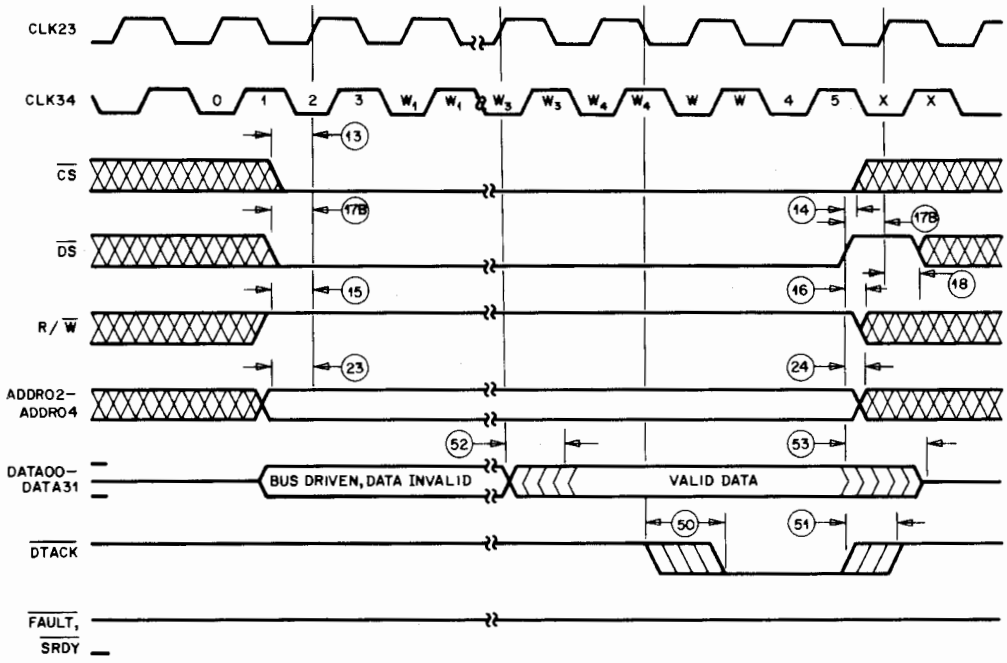


Figure 12. Peripheral Mode Read

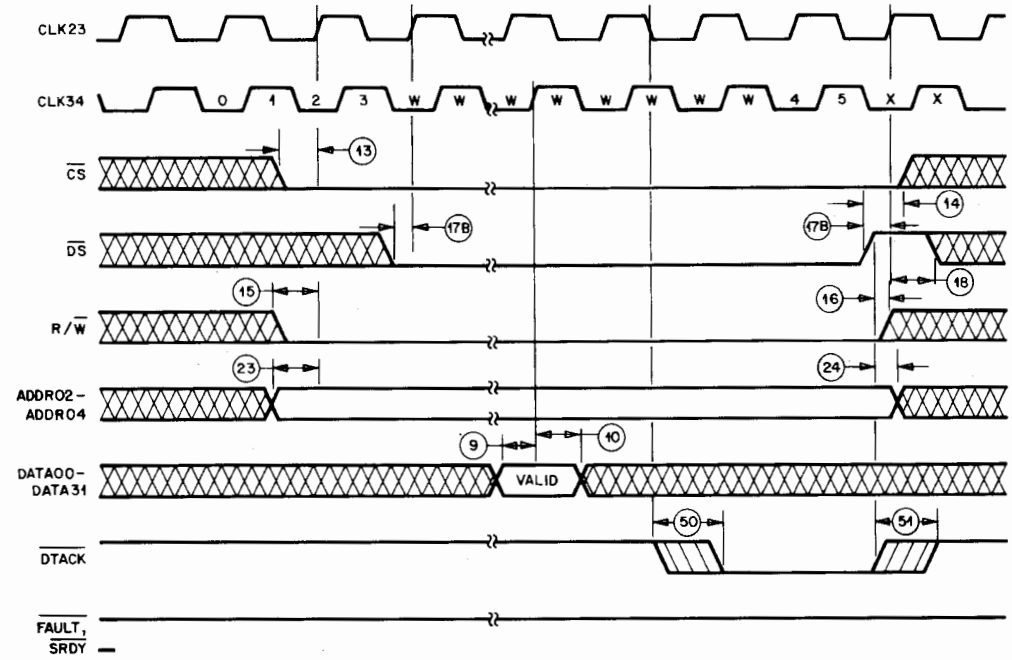


Figure 13. Peripheral Mode Write

Electrical Characteristics

Inputs

All inputs, except the CMOS input clocks, are TTL-compatible.

Table 29. DC Input Parameters

Inputs		Min	Max	Unit
TTL input voltage	high-level	2	V_{CC}	V
	low-level	0	0.8	V
CMOS clocks input voltage	high-level	V_{CC} – 1.3	V_{CC}	V
	low-level	0	0.8	V
TTL input loading current (2.0 V ≤ V_{IH} ≤ V_{CC})	high-level	0	0.01	mA
TTL input loading current (0 V ≤ V_{IL} ≤ 0.8 V)	low-level	–0.01	0	mA
CMOS clocks input loading current (V_{CC} – 1.5 V ≤ V_{IH} ≤ V_{CC})	high-level	0	0.01	mA
CMOS clocks input loading current (0 ≤ V_{IL} ≤ 1.0)	low-level	–0.01	0	mA

Outputs

Listed below are descriptions of the outputs assigned to classes 1 and 2.

Class 1: This class is capable of driving one TTL load or 8 PNP Schottky TTL loads and has current allowance for an external holding resistor employed in 3-state buffers. The minimum holding resistor value is 2.7 kΩ.

Class 2: The signal in this class is an open drain output used for wired logic operations, allowing more than one device to drive a node without conflict. An external resistor is required to pull this signal high. The minimum pull-up resistor value is 680 Ω.

The following lists the outputs assigned to Classes 1 and 2:

Class 1	Class 2
DATA00—DATA31	$\overline{\text{DONE}}$
	$\overline{\text{SRDY}}$
	$\overline{\text{FAULT}}$
	$\overline{\text{DTACK}}$

Table 30. DC Output Parameters

Outputs		Min	Max	Unit
Output sink current (I_{OL}) (V_{OL} ≤ 0.4 V)	Class 1	—	3.5	mA
	Class 2	—	12	mA
Output source current (I_{OH}) (V_{OH} ≥ 2.4 V)	Class 1	—	3.5	mA
	Class 2	—	10	μA
Output logic levels*	high-level	2.4	—	V
	low-level	—	0.4	V

* Referenced to system ground (GND).

Operating Conditions

Table 31. Operating Conditions

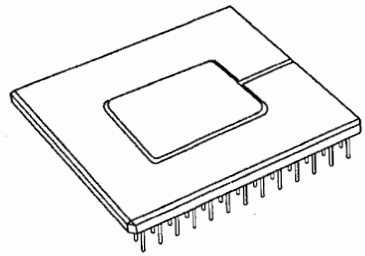
Parameter		Symbol	Min	Nom	Max	Unit
Supply voltage		V_{CC}	4.75	5.00	5.25	V
Input load capacitance	TTL inputs	C_{IN}	—	—	12	pF
	CMOS clocks		—	—	7	pF
Total output load capacitance	Class 1	C_L	—	—	130	pF
	Class 2		—	—	130	pF
Ambient temperature at the microprocessor pins		T_A	0	—	70	°C
Humidity range		—	5%	—	95%	RHNC*
Power dissipation	at 10 MHz	PD	—	—	0.5	W
	at 14 MHz		—	—	0.7	W
	at 18 MHz		—	—	0.9	W
Operating frequency		F	—	—	18	MHz
Capacitive derating factor (25 pF≤C_L≤225 pF)	at 10 MHz	dt/dC	—	0.1	—	ns/pF
	at 14 MHz		—	0.1	—	ns/pF
	at 18 MHz		—	.08	—	ns/pF

* Relative humidity, noncondensing.

WE® 32206 Math Acceleration Unit

Description

The *WE* 32206 Math Acceleration Unit (MAU), provides floating-point capability for the *WE* 32200 Microprocessor and is fully compatible with the IEEE Standard for Binary Floating-Point Arithmetic (ANSI/IEEE Std. 754-1985). The *WE* 32206 MAU is implemented in 1-micron CMOS technology. It provides single (32-bit), double (64-bit), and double-extended (80-bit) precision; and add, subtract, multiply, divide, remainder, square root, and compare operations as well as sin, cos, atan, and pi. The operand, result, and status and command information transfers take place over a 32-bit bidirectional data bus that provides the interface to the host microprocessor. The *WE* 32206 MAU is footprint, protocol, and upward



object code compatible with the *WE* 32106 MAU. The MAU operates at 24-MHz and higher versions and requires only a single 5 V supply; and is packaged in a 125-pin square, ceramic pin grid array (PGA).

Features

- Compatible with ANSI/IEEE Standard 754-1985 for Binary Floating-Point Arithmetic
- Single (32-bit), double (64-bit), and double-extended (80-bit) precision capability
- Add, subtract, multiply, divide, remainder, negate, absolute value, and square root functions
- Transcendental functions (sine, cosine, arctangent), and pi
- Compare, move, and rounding to integral value functions
- Eight 80-bit user registers
- Coprocessor and peripheral mode interfaces available
- Symmetric integer, decimal, and floating-point conversions
- 32-bit I/O interface

Description 395
Features 395
User Information 397
System Design 397
Protocol Information 397
Coprocessor Mode Protocol 397
Peripheral Mode Protocol 397
Registers 397
Auxiliary Status Register 397
Operand Registers 399
Command Registers 399
Data Register 401
Interrupts 402
Restart Capability 402
System Configuration Restrictions 402
Peripheral Mode 402
Coprocessor Mode 402
MAU-CPU Interconnections 403
Programming 404
Data Types and Formats 404
Single-Precision 404
Double-Precision 405
Double-Extended-Precision 405
Decimal 406
Integer 407
Rounding 407
Faults 407
Exceptions 407
Peripheral Mode Accesses 408
Operating System Considerations 408
Peripheral Mode Protocol 408
Coprocessor Mode Protocol 409
Context Saving and Restoring 409
Peripheral Mode 409
Coprocessor Mode 409
Instruction Set Summary by Functional Groupings 409
Arithmetic Group 409
Data Transfer Group 410
Logical Group 410
Conversion Group 410
Transcendental Group 410
Miscellaneous Group 410
Instruction Set Summary by Mnemonic and Opcode 410
Definitions 413
Pin Descriptions 414
Numerical Order 415
Functional Groups 416
Characteristics 418
Timing Characteristics 418
Timing Diagrams 420
Electrical Characteristics 426
Inputs 426
Outputs 427
Operating Conditions 427

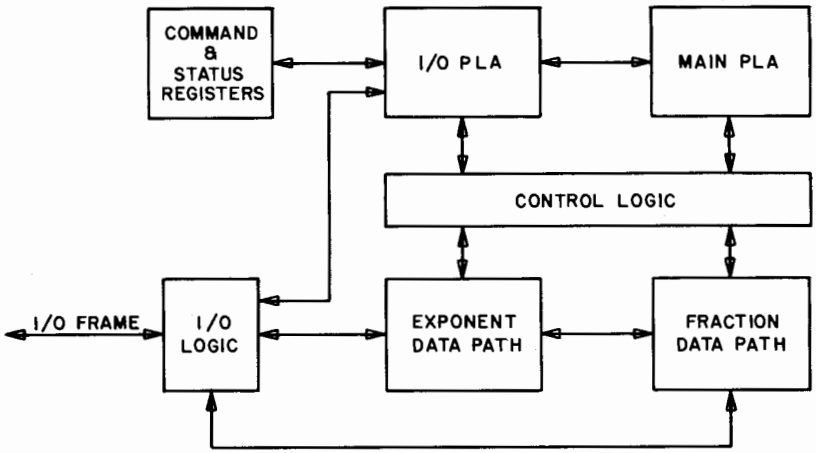


Figure 1. WE® 32206 Math Acceleration Unit Block Diagram

User Information

The MAU has two major subsystems: the fraction datapath and the exponent datapath. The data width of each datapath was chosen to facilitate calculations in the widest data format. All operands are converted, upon entry to the MAU from memory, to the double-extended precision format. This causes no loss of precision and saves internal hardware that would be necessary to explicitly support the narrower formats in all internal operations.

System Design

Protocol Information

See Definitions for an explanation of the terminology used in the following text.

The MAU interacts with a system via two different protocols: coprocessor and peripheral mode protocol. The coprocessor mode protocol allows high system throughput when the MAU is used with the *WE* 32200 Microprocessor. The peripheral mode protocol allows use of the MAU with any CPU that can perform data read and write bus transactions.

Coprocessor Mode Protocol. In coprocessor mode protocol, the CPU initiates an MAU transaction by performing a coprocessor broadcast access. This access includes a word in the MAU command register (CR) format (see Registers). The MAU checks the ID field of this word against the MAU ID value of 0. If they match, the word is stored in the MAU CR.

If any of the operand specifiers in the command word indicate that an operand is to be obtained from memory, the MAU waits until the proper number of coprocessor data fetch bus transactions occur. The words are stored in internal registers.

The MAU performs the operation, generating a result, new condition codes, and possibly an exception. The $\overline{\text{DONE}}$ pin is asserted and the MAU waits until a coprocessor status fetch access is seen. If an exception is present, the MAU faults the access and returns to the quiescent state. If there is no exception, a word containing the current auxiliary status register (ASR) value is returned.

If the result is to be placed in memory, the MAU waits until the proper number of coprocessor data write bus transactions occur, putting a word of the result on the bus as each transaction occurs. The MAU then returns to the quiescent state.

Peripheral Mode Protocol. In the peripheral mode protocol, the MAU registers appear as memory-mapped locations and are accessed via normal read and write operations. The CPU (or any other bus master) starts an MAU transaction by writing a word into the MAU command register. When the write access is completed, the MAU clears the result available (RA) bit in the auxiliary status register. The command is then executed using the operands available in registers F0 through F7, or by reading values via the data register (DR).

If any of the operands are to come from memory, the bus master must write the operands into the DR one word at a time after the CR has been written. These operands are latched into internal registers. If the result is to be placed in memory, the bus master must read the result from the DR one word at a time and transfer the words to memory.

After the operation has been completed, the MAU updates the ASR contents to reflect the new condition codes, exception bits, and any other bits resulting from the operation. The RA bit in the ASR is set (1), $\overline{\text{PIREQ}}$ is asserted, and the MAU enters the quiescent state. $\overline{\text{PIREQ}}$ may be used to interrupt the CPU upon completion of a peripheral mode MAU operation. $\overline{\text{PIREQ}}$ can be negated by asserting $\overline{\text{PIACK}}$ for easy interfacing to an interrupt controller.

Registers

Auxiliary Status Register. The auxiliary status register (ASR) controls the remainder operation (partial remainder bit), signals the state of an operation (result available bit), disables and records exceptions (mask and sticky bits), controls rounding of results (round control bits), and records condition codes (negative and zero bits).

The negative, zero, integer overflow, and inexact bits in the ASR match the positions of the negative, zero, overflow and carry bits (the condition codes) in the PSW register of the *WE* 32200 Microprocessor. This allows the bits to be copied into the PSW as part of the coprocessor status access and to be easily tested by CPU software.

Table 1. Auxiliary Status Register

Bit	15	14	13	12	11	10	9	8	7	6	5	4	3 1	0
Field	WF	IM	OM	UM	QM	PM	IS	OS	US	QS	PR	UW	VER	FE

Bit	31	30 26	25	24	23 22	21	20	19	18	17	16
Field	RA	X	ECP	NTNC	RC	N	Z	IO	PS	CSC	UO

Bit	Field	Contents	Description
0	FE	Feature enable	Enables the WF and UW bits of the auxiliary status register when set.
1—3	VER	Version number	Value=001 for *WE* 32206 Math Acceleration Unit, 000 for *WE* 32106 Math Acceleration Unit. These bits are only readable and cannot be written.
4	UW	Unaligned word bit	Set when a fault condition occurs due to an unaligned word being received by the MAU, if bit 0 (feature enable) is set.
5	PR	Partial remainder	Set when result of a remainder operation is a partial remainder; cleared (0) when result is a full remainder.
6	QS	Divide by zero sticky	Set if the divisor is normalized zero and dividend is a finite nonzero number.
7	US	Underflow sticky	Set if exponent of a rounded result of an arithmetic operation is too small to be represented in the exponent field of the destination format.
8	OS	Overflow sticky	Set if exponent of a rounded result of an arithmetic operation is too large for the exponent field of the destination format.
9	IS	Invalid operation sticky	Set if a result cannot be stored in a destination legally, or if illegal operands are given to some operation.
10	PM	Inexact mask	If this bit is set by the user, an exception occurs when bit 18 (inexact sticky) is set. If this bit is cleared, there are no inexact exceptions.
11	QM	Divide by zero mask	If this bit is set by the user, an exception occurs when bit 6 is set. If this bit is cleared, there are no divide by zero exceptions.
12	UM	Underflow mask	If this bit is set by the user, an exception occurs when bit 7 is set. If this bit is cleared, there are no underflow exceptions.
13	OM	Overflow mask	If this bit is set by the user, an exception occurs when bit 8 is set. If this bit is cleared, there are no overflow exceptions.
14	IM	Invalid operation mask	If this bit is set by the user, an exception occurs when bit 9 is set. If this bit is cleared, there are no invalid operation exceptions.
15	WF	Write fault indicator	Set when a fault condition occurs during the writing of any result to memory, if bit 0 (feature enable) is set. Remains set until the instruction completes without fault. If a RDASR, WRASR, LDR, EROF, or NOP occurs while this bit is set, it will not be interpreted as a restart of the operation that originally caused the fault. When this bit is set, the MAU, upon restart of the operation by the CPU, will not reexecute the operation, but will instead return a DONE signal, and use the result already residing in the data register from the previously faulted operation and store it in memory. This bit may be cleared with a WRASR.

Table 1. Auxiliary Status Register (Continued)

Bit	Field	Contents	Description
16	UO	Unordered	Set when a compare operation results in an unordered indication; otherwise it is cleared.
17	CSC	Context switch control	Set on every MAU instruction execution except for RDASR, WRASR, LDR, EROF, or NOP.
18	PS	Inexact sticky	Set if the result cannot be specified in the destination format. This bit is cleared on reset. For any COMPARE operation, the MAU will output the status of the unordered (UO) bit when the ASR is copied back to the CPU PSW.
19	IO	Integer overflow	Set when a convert float to integer operation causes an overflow.
20	Z	Zero	Set if result of last operation is zero. If result is nonzero, bit is cleared.
21	N	Negative	Set if result of the last operation is negative. If result is positive, bit is cleared. Note that it is possible for an operation to result in both bits 20 and 21 being set since negative zero is a representable number.
22, 23	RC	Round control	Represent the round control mode. The code is interpreted as: **Bit Code — Description** 00 — Round to nearest 01 — Round towards plus infinity 10 — Round towards minus infinity 11 — Round towards zero (truncation)
24	NTNC	Nontrapping NaN control	This bit is tested when an invalid operation exception occurs and bit 14 is cleared. If this bit is set, an exception occurs and bit 9 is set. It is expected that software will write a virtual program counter value into the fraction portion of the nontrapping NaN generated.
25	ECP	Exception condition present	Set if any one of the floating-point exception conditions except inexact is present.
26—30	Unused	Unused	These bits appear as 0 when read.
31	RA	Result available	Cleared at beginning of an operation and set when result of an operation is available. During the quiescent state, this bit has a value of 1.

Operand Registers. Each of the eight operand registers (F0—F7) is 80-bits and contains one operand in extended format (see Table 2). The registers are shown as 96-bits because they are read and written as three 32-bit words through the data register. In peripheral mode, bits 80 through 95 are ignored during writes and returned as 0s during read operations. Registers F0 through F7 are unchanged on reset; they are indeterminate on power-up.

Command Register. The command register (CR) accepts command words which are used to initiate an MAU transaction. The command register format is shown in Table 3.

Table 2. Operand Registers

Bit	95 80	79	78 64	63	62 0
Field	Unused	Sign	Exponent	J	Fraction

Bit	Field	Contents	Description
0—62	Fraction	Fraction	These bits represent the fractional part of the number.
63	J	Explicit bit	The J bit resides to the left of the binary point in the 2^0 position. Together, the J bit and fraction field can represent values in the range 0 to $2-(2^{-63})$.
64—78	Exponent	Exponent	The exponent field contains an exponent that is biased by 16383.
79	Sign	Sign	A 1 represents a negative value; 0 represents a positive value.
80—95	Unused	—	—

Table 3. Command Register

Bit	31 24	23 21	20	19	18	17	16 15	14 10	9 7	6 4	3 0
Field	ID	Unused	RB1	RB2	RB3	RCS	RC	Opcode	Op1	Op2	Op3

Bit	Field	Contents	Description
0—3	Op3	Operand specifier	Specifies whether destination operand is an MAU register, a memory-based operand of a given size, or nonexistent (no operand). Even though the register destinations are specified as single-, double-, or double-extended, the result is stored in the registers in double-extended precision, but rounded to the specified precision. The precision designations are used for rounding and checking for underflow and overflow. The value of this field is interpreted as:

Bit 3,2,1,0	Operand Reg	Destination Precision	Bit 3,2,1,0	Operand Reg	Destination Precision
0000	F0 or F4	Single	1000	F0 or F4	Double-extended
0001	F1 or F5	Single	1001	F1 or F5	Double-extended
0010	F2 or F6	Single	1010	F2 or F6	Double-extended
0011	F3 or F7	Single	1011	F3 or F7	Double-extended
0100	F0 or F4	Double	1100	—	Memory-based single-word
0101	F1 or F5	Double	1101	—	Memory-based double-word
0110	F2 or F6	Double	1110	—	Memory-based triple-word
0111	F3 or F7	Double	1111	—	No operand

In all cases, the actual register chosen depends on the RB3 specifier.

Bit	Field	Contents	Description
4—6	Op2	Operand specifier	Specifies whether second source operand is an MAU register, a memory-based operand of a given size, or nonexistent (no operand). The value of this field is interpreted as:

Bit 6,5,4	Operand Location
000	Register F0 or F4 (depending on RB2 specifier)
001	Register F1 or F5 (depending on RB2 specifier)
010	Register F2 or F6 (depending on RB2 specifier)
011	Register F3 or F7 (depending on RB2 specifier)
100	Memory-based single-word
101	Memory-based double-word
110	Memory-based triple-word
111	No operand

Table 3. Command Registers (Continued)

Bit	Field	Contents	Description
7—9	Op1	Operand specifier	Specifies whether first source operand is an MAU register, a memory-based operand of a given size, or nonexistent (no operand). The value of this field is interpreted as: **Bit 9,8,7 — Operand Location** 000 — Register F0 or F4 (depending on RB1 specifier) 001 — Register F1 or F5 (depending on RB1 specifier) 010 — Register F2 or F6 (depending on RB1 specifier) 011 — Register F3 or F7 (depending on RB1 specifier) 100 — Memory-based single-word 101 — Memory-based double-word 110 — Memory-based triple-word 111 — No operand
10—14	Opcode	Operation code	Specifies operation to be performed.
15—16	RC	Round control	Determines the method of rounding used if the RCS bit = 1. When the RC bits in the CR are in control, they have the following meaning: **Bit 15,16 — Description** 00 — Round to nearest 01 — Round towards positive infinity 10 — Round towards minus infinity 11 — Round towards zero (truncation)
17	RCS	Round control selection	Determines where the control of the rounding is located. It has the following meaning: **Bit 17 — Description** 0 — Round control is determined by the RC bits in the ASR 1 — Round control is determined by the RC bits in the CR (ASR value is unchanged)
18—20	RB3—RB1	Register bank	Specifies which bank of registers will be used as an operand: **Bit 18, 19, or 20 — Meaning** 0 — Use register F0, F1, F2, or F3 1 — Use register F4, F5, F6, or F7
21—23	—	Unused	These bits are returned as 0 when read.
24—31	ID	Processor ID number	Specifies identification number of the processor that should react to the command word. The MAU ID is 0.

Data Register. The data register (DR) is used to read and write operands via peripheral mode accesses. The DR appears as three 32-bit words in the peripheral mode address space. On occurrence of an exception, the information supplied to the trap handler is stored in the DR. This information is summarized in Table 4. An extract result on fault (EROF) instruction can be executed to retrieve this information from the DR.

Table 4. Data Register Trap Handler Information

Exception Type	Information Supplied
Invalid operation	If either of the source operands is a trapping NaN, the DR will contain the NaN converted to double-extended precision (80 bits) if necessary. If both source operands are trapping NaNs or infinities of different signs, the second operand (Op2) in double-extended precision (80 bits) is stored in the DR.
Overflow or underflow	The significand along with the 17-bit internal result exponent and result sign are stored in the DR. The most significant bit of the 17-bit exponent behaves like a sign bit in 2's complement notation. An additional bit (bit 79) in the exponent ensures that no significant exponent bits are lost from an internal representation on occurrence of an overflow or underflow condition. The exponent value is biased by 16383. The least significant bit (L) of the unrounded result, as well as the guard (G), round (R), and sticky (S) bits of the unrounded result will be stored in bits 95 through 92, respectively, of operand 3. The format is: Bit: 95 | 94 | 93 | 92 | 91 82 | 81 | 80 64 | 63 | 62 0 Field: L | G | R | S | Unused | Sign | Exponent | J | Fraction SIGNIFICAND
Divide by zero	The dividend (Op2) converted to double-extended precision, if necessary, is stored in the DR. The L, G, R and S bits are stored in bits 95 through 92, respectively.
Inexact	The rounded result converted to double-extended precision, if necessary, is stored in the DR. The L, G, R, and S bits are stored in bits 95 through 92, respectively.

Interrupts

Once a transaction has been initiated via a peripheral mode write to the CR, the MAU ignores any interrupt acknowledge or autovector interrupt acknowledge bus cycles. This continues until the next MAU transaction is started.

Using coprocessor protocol, once an operation has been initiated via a coprocessor broadcast access, the MAU responds to any interrupt acknowledge or autovector interrupt acknowledge bus cycles by aborting the transaction and returning to the quiescent state.

Restart Capability

During peripheral mode accesses, reads and writes will always terminate normally, with no fault indication. It is possible that the CPU will read the DR successfully and then encounter a fault when writing to memory. The CPU may then restart the instruction causing another read to the DR.

Using coprocessor protocol, the MAU is fully restartable because it is capable of resuming any instruction at the point of the memory store. See Table 1 (the ASR WF bit) for a description.

System Configuration Restrictions

Peripheral Mode. Under peripheral mode protocol, care must be taken to ensure that no more than one coprocessor drives the $\overline{\text{DONE}}$ signal at any one time. Also, if non-addressed devices cause bus exceptions in peripheral mode, incorrect data may be written to the MAU during writes.

Coprocessor Mode. The MAU ID number is fixed, so it is not possible to have more than one MAU on the same bus in coprocessor mode.

Because the MAU reacts to all coprocessor broadcast accesses, it is not possible to have more than one coprocessor protocol transaction in progress at any one time. If an MAU and several other coprocessors exist on one bus, only one transaction can be in progress at any one time.

MAU-CPU Interconnections. Several MAU signals are interfaced directly to the CPU. These signals are shown on Figures 2 and 3.

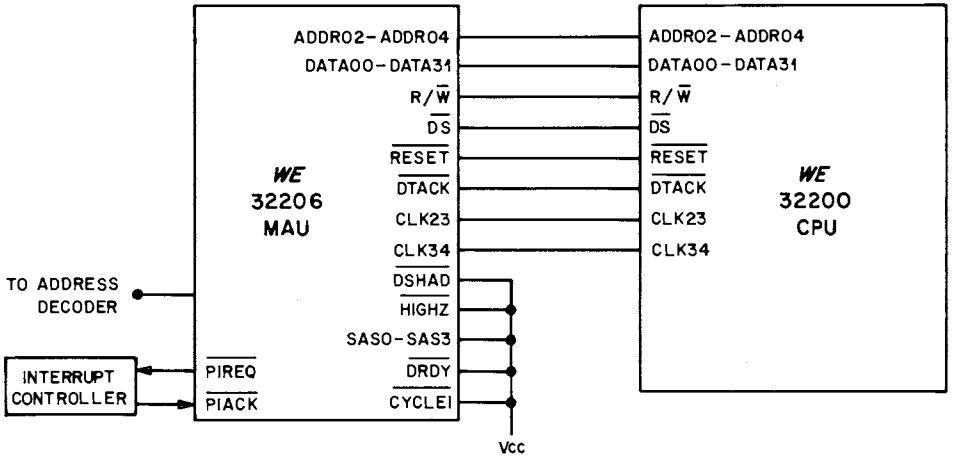
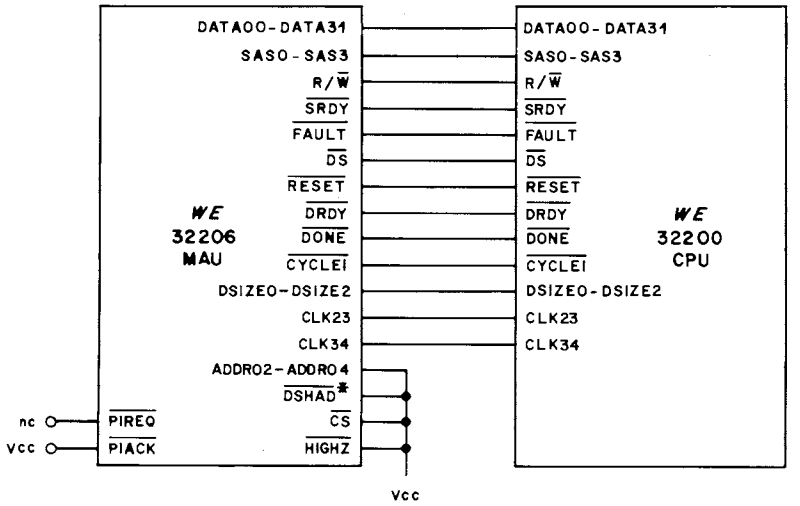


Figure 2. MAU-CPU Peripheral Mode Protocol Interconnections



* If an MMU is included in the system, this signal must be connected to the MMU's $\overline{\text{DSHAD}}$ output.

Figure 3. MAU-CPU Coprocessor Mode Protocol Interconnections

Programming

Data Types and Formats

The *WE* 32206 Math Acceleration Unit (MAU) supports single, double, double-extended, decimal, and integer data formats.

The integer format provides a binary representation of signed integers in the 32-bit 2's complement format.

Single-precision, double-precision, and double-extended-precision formats are available for representing floating point numbers. The single-precision format provides an eight bit exponent and an exponent bias, which allows the reciprocal of all normalized numbers to be represented without overflow.

With double-precision, the exponent range is sufficient to ensure that the product of eight 32-bit terms cannot overflow the 64-bit format.

Double-extended-precision provides a format with a range and precision that is greater than double-precision format but does not require twice as many bits for representation. Having greater precision, double-extended-precision numbers lessen the chance of a final result that has been contaminated by excessive roundoff error; with greater range, the chance of an intermediate overflow aborting a computation whose result would have been representable in a basic format is lessened.

The decimal format provides a method for representing 18-digit signed decimal numbers in BCD (i.e., 18 BCD digits and a sign nibble).

A discussion of data formats follows.

Single-Precision. Single-precision operands are 32 bits long. The single-precision format is shown and described in Table 5. Table 6 shows how certain special-case values are represented using the single-precision format.

Table 5. Single-Precision Format

Bit	31	30 23	22 0
Field	Sign	Exponent	Fraction

Bit	Field	Description
0—22	Fraction	**Fraction.** A 23-bit string that encodes the significant bits of the number. For normalized numbers, an implicit bit that has a value of 1 resides immediately to the left of the binary point. The implicit bit and fraction can represent values in the range 1 to $(2-2^{-23})$, inclusive.
23—30	Exponent	**Exponent.** These 8 bits represent an exponent biased by 127.
31	Sign	**Sign Bit.** A 1 represents a negative value; 0 represents a positive value.

Table 6. Single-Precision Special-Case Values

			Fraction Value	
Value Name	Sign Value	Exponent Value	Bit 22	Bits 0—61
Nontrapping NaN	x	max	1	x
Trapping NaN	x	max	0	nonzero
Positive infinity	0	max	min	
Negative infinity	1	max	min	
Positive zero	0	min	min	
Negative zero	1	min	min	
Denormalized number	x	min	nonzero	
Normalized number	x	notmm	x	

Note: x = don't care

Double-Precision. The double-precision operands are 64 bits long. The double-precision format is presented in Table 7. Table 8 shows how certain special-case values are represented using the double-precision format.

Double-Extended-Precision. Double-extended-precision operands are 80 bits long. Table 9 describes the double-extended-precision format. Table 10 shows how certain special-case values are represented using the double-extended-precision format.

Table 7. Double-Precision Format

Bit	63	62 52	51 0
Field	Sign	Exponent	Fraction

Bit	Field	Description
0—51	Fraction	**Fraction.** A 52-bit string that encodes the significant bits of the number. For normalized numbers, an implicit bit that has a value of 1 resides immediately to the left of the binary point. The implicit bit and the fraction can represent values in the range 1 to $(2-2^{-52})$, inclusive.
52—62	Exponent	**Exponent.** These 11 bits represent an exponent biased by 1023.
63	Sign	**Sign Bit.** A 1 represents a negative value; 0 represents a positive value.

Table 8. Double-Precision Special-Case Values

Value Name	Sign Value	Exponent Value	Fraction Value	
			Bit 51	Bits 0—50
Nontrapping NaN	x	max	1	x
Trapping NaN	x	max	0	nonzero
Positive infinity	0	max	min	
Negative infinity	1	max	min	
Positive zero	0	min	min	
Negative zero	1	min	min	
Denormalized number	x	min	nonzero	
Normalized number	x	notmm	x	

Note: x = don't care.

Table 9. Double-Extended-Precision Format

Bit	95 80	79	78 64	63	62 0
Field	Unused	Sign	Exponent	J	Fraction

Bit	Field	Description
0—62	Fraction	**Fraction.** A 63-bit string that encodes the significant bits of the number.
63	J	**Explicit Bit.** This bit resides immediately to the left of the binary point. The fraction and explicit bit together can represent numbers in the range 0 to $(2-2^{-63})$, inclusive.
64—78	Exponent	**Exponent.** These 15 bits represent an exponent biased by 16383.
79	Sign	**Sign Bit.** A 1 represents a negative value; 0 represents a positive value.
80—95	Unused	**Unused.** These bits are ignored and overwritten with 0s.

Table 10. Double-Extended-Precision Special-Case Values

Value Name	Sign Value	Exponent Value	J Value	Fraction Value	
				Bit 62	Bits 0—61
Nontrapping NaN	x	max	x	1	x
Trapping NaN	x	max	x	0	nonzero
Positive infinity	0	max	0	min	
Negative infinity	1	max	0	min	
Positive zero	0	min	0	min	
Negative zero	1	min	0	min	
Denormalized number	x	min	0	nonzero	
Unnormalized number	x	notmm	0	x	
Normalized number	x	notmax	1	x	

Note: x = don't care.

Decimal. Decimal operands are 76 bits long. Table 11 shows the format for a decimal operand and describes each bit position. When doing float-to-decimal and decimal-to-float conversion operations, software support is needed to provide the range required by the IEEE standard for these operations. Table 12 shows how certain special-case values are represented using the decimal format.

Table 11. Decimal Format

Bit	95 76	75 72	71 68	67 64	63 60	59 56	55 52	51 48	47 44	43 40
Field	UNUSED	D17	D16	D15	D14	D13	D12	D11	D10	D9

Bit	39 36	35 32	31 28	27 24	23 20	19 16	15 12	11 8	7 4	3 0
Field	D8	D7	D6	D5	D4	D3	D2	D1	D0*	Sign

* D stands for digit. Each digit is 4-bits wide.

Bit	Field	Description
0—3	Sign	**Sign Bits.** These 4 bits represent the sign of the number. They are also used to represent NaN's and infinity. The sign value has the following meaning:

Bit 3,2,1,0	Meaning	Bit 3,2,1,0	Meaning
0000	Positive infinity	1000	Trapping NaN
0001	Negative infinity	1001	Trapping NaN
0010	Positive NaN	1010	Positive number
0011	Negative NaN	1011	Negative number
0100	Trapping NaN	1100	Positive number
0101	Trapping NaN	1101	Negative number
0110	Trapping NaN	1110	Positive number
0111	Trapping NaN	1111	Positive number

4—75	D0—D17	**Digits.** These 72 bits make up 18 decimal digits (each digit is 4-bits wide). The leftmost digit (D17) is the most significant.
76—95	Unused	**Unused.** These bits are ignored and overwritten with 0s.

Table 12. Decimal Format Special Case Values

Value Name	Digit Values	Sign Value
Nontrapping NaN	x	0010 or 0011
Nontrapping NaN	1010,1011,1100,1101, 1110 or 1111	1010, 1011, 1100, 1101, 1110 or 1111
Trapping NaN	x	0100, 0101, 0110, 0111, 1000 or 1001
Positive infinity	x	0000
Negative infinity	x	0001
Number	0000 through 1001	1010, 1011, 1100, 1101 1110 or 1111

Note: x = don't care.

Integer. An integer number is represented as a 2's complement 32-bit operand. This representation is shown on Figure 4.

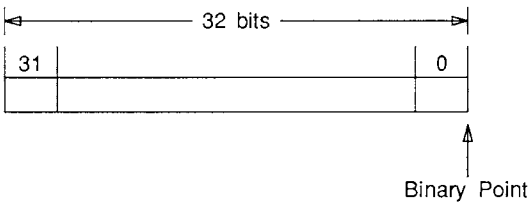


Figure 4. Integer Format

Rounding

The MAU performs rounding at the end of every arithmetic operation; it is not a separate operation that is triggered via an opcode in the command word.

Rounding is controlled by bits 22 and 23 of the ASR or bits 15 and 16 of the CR. By changing the value of these bits, results can be rounded to the nearest representable number, towards positive infinity, towards minus infinity, or towards zero (truncation). See Tables 1 and 3 for a description of the round control options.

Faults

When using the peripheral mode protocol, the MAU does not signal any faults and does not react to any faults on the system bus. If an operation results in an exception, the exception is signaled to the CPU by setting the appropriate ASR bits.

With the coprocessor mode protocol, if an exception occurs as the result of an operation, the MAU faults the coprocessor status fetch access after the ASR contents have been updated to reflect the result of the operation. If a coprocessor data fetch or a coprocessor data write access is not completed successfully, the MAU waits until the access is completed.

If a coprocessor broadcast access occurs during an MAU transaction, the MAU aborts the current transaction. If the ID associated with the access is not the MAU ID, the MAU enters the quiescent state. If the ID is equal to the MAU ID, the MAU starts the operation specified by the access.

The bus master that initiates a coprocessor broadcast access must also detect when no coprocessor reacts to the access.

Exceptions

For each exception type, there is a mask bit and a sticky bit in the ASR. If the condition associated with the exception type is satisfied, the MAU sets the sticky bit for that exception type. If the mask bit is 1, an exception occurs. If the mask bit is 0, no exception occurs.

The MAU never changes the mask bits or clears the sticky bits. These actions, if necessary, must be done using software.

The exceptions follow an order of precedence so that the highest priority exception is taken if more than one occurs. The ordering from highest priority to lowest priority is:

Invalid operation (highest priority)
Divide by zero, overflow, and underflow
Inexact (lowest priority).

Peripheral Mode Accesses

In peripheral mode, objects in the MAU can be accessed as memory-mapped addresses. A peripheral mode access occurs when there is a data read or data write access and the MAU chip select input is being asserted. The MAU latches the address associated with the access and interprets it as shown in Table 13. If the access is a read, the contents of the selected MAU object are returned as data. If the access is a write, the data associated with the access are stored in the MAU object selected.

The ASR and DR are readable and writable in peripheral mode. The CR can be written in peripheral mode, but not read.

The *WE* 32206 Math Acceleration Unit faults all non-word accesses when the FE bit in the ASR is enabled. The following is true only when the FE bit is cleared.

All peripheral mode accesses are treated as word accesses by the MAU. If the *WE* 32200 CPU is used to perform byte or halfword reads from the MAU in peripheral mode, these accesses will be completed correctly because the CPU ignores the extra bytes returned by the MAU. If the CPU is used to perform byte or halfword writes to the MAU, these accesses will be completed, but a full word of data is written into the MAU. The specified byte or halfword will be written correctly, but the rest of the data written may take on any value. Byte and halfword reads and writes are byte-order dependent operations.

Operating System Considerations

Peripheral Mode Protocol. Using peripheral mode protocol, the MAU does not abort its operation when an interrupt acknowledge cycle occurs. The peripheral mode protocol assumes that the CPU will be executing a stream of discrete instructions to implement the protocol, and thus would not be able to back up the first instruction in the event of an interrupt. The CPU is free to allow or disallow interrupts during MAU operations, however, MAU context saving is simplified if the interrupts can be disabled.

The state of the MAU must be saved and restored later if an interrupt causes control of the MAU to be given to another process. If the MAU is in the middle of an operation and the RA bit of the ASR is cleared, or if $\overline{\text{PIREQ}}$ is cleared, the MAU's state cannot be saved and restored without explicit software provisions. If the first CPU instruction of the sequence can be found, the context of the partially executed transaction can be saved so that execution can resume there. The context to be saved and

Table 13. Peripheral Mode Address Fields

Bit	31 — 5	4 — 2	1 — 0
Field	Unused	Object	Unused

Bit	Field	Description
0—1	Unused	Ignored by the MAU.
2—4	Object	Select which object is to be accessed as per object value specified. (Bit 2 is the least significant bit.)

Object Bit 4 3 2	Object Addressed
0 0 0	ASR
0 0 1	ASR
0 1 0	ASR
0 1 1	ASR
1 0 0	DR bits 64—95
1 0 1	DR bits 32—63
1 1 0	DR bits 0—31
1 1 1	CR

5—31	Unused	Ignored by the MAU.

restored consists of the ASR, DR, and F0—F7. It may be necessary to run the context save and restore sequences with interrupts disabled.

If an interrupt occurs and the RA bit of the ASR has been set, the MAU is placed in the quiescent state. The transaction may or may not have completed (some reads from the DR may not be complete). The context to be saved and restored consists of the ASR, DR, and F0—F7 (See Context Saving and Restoring.)

Coprocessor Mode Protocol. If an interrupt occurs during an MAU transaction but before the $\overline{\text{DONE}}$ signal goes to 0, the MAU aborts the operation. It is possible that the ASR has been changed to reflect exceptions or new condition codes; however, when the CPU returns from the interrupt it can restart the transaction from the beginning, and the spurious ASR value will not affect the repeated operation.

The context of the MAU must be saved and restored later if the interrupt causes control of the MAU to be given to another process. The context to be saved and restored consists of the ASR, DR, and F0—F7.

Context Saving and Restoring

Peripheral Mode. The MAU context can be saved by:

- Reading the ASR via peripheral mode and saving its value.
- Clearing the ASR.
- Reading the three DR words via peripheral mode and saving their value.
- Reading registers F0—F7 via MOVE instructions and saving their values in memory locations. The values will appear in the DR and must then be moved to memory.

The MAU context can be restored by:

- Performing eight MOVE instructions to restore registers F0—F7. The values must be written to the DR via peripheral mode accesses.
- Performing an MAU MOVE operation with the source being a memory location (the value must be written into the DR via a peripheral mode access) and the destination being a memory location (the value will appear in the DR). This restores the DR.
- Writing the saved ASR value into the ASR via peripheral mode.

Coprocessor Mode. The MAU context can be saved by:

- Reading DR with an extract result on fault (**EROF**) instruction and saving its value.
- Reading the ASR via peripheral mode or a move from ASR (RDASR) instruction and saving its value.
- Clearing the ASR.
- Performing eight MAU MOVE operations to save registers F0—F7.

The MAU context can be restored by:

- Clearing the ASR.
- Performing eight MAU MOVE operations to restore registers F0—F7.
- Writing the saved ASR value into the ASR via peripheral mode or a move to ASR (WRASR) instruction.
- Performing a load DR operation using the LDR instruction or via peripheral mode move to restore DR.
- Clearing the ASR.

Instruction Set Summary by Functional Groupings

Arithmetic Group. These instructions perform arithmetic operations on data.

Table 14. Arithmetic Group

Instruction	Mnemonic	Opcode
Absolute Value	ABS	0x0C
Add	ADD	0x02
Subtract	SUB	0x03
Multiply	MUL	0x06
Divide	DIV	0x04
Remainder	REM	0x05
Square root	SQRT	0x0D
Negate	NEG	0x17
Round to integral value	RTOI	0x0E

Data Transfer Group. These instructions transfer data to and from registers and memory.

Table 15. Data Transfer Group

Instruction	Mnemonic	Opcode
Move	MOVE	0x07
Move from ASR	RDASR	0x08
Move to ASR	WRASR	0x09
Load DR	LDR	0x18

Logical Group. These instructions perform logical operations on data.

Table 16. Logical Group

Instruction	Mnemonic	Opcode
Compare	CMP	0x0A
Compare with exceptions	CMPE	0x0B
Compare with flags swapped	CMPS	0x1A
Compare with exceptions and flags swapped	CMPES	0x1B

Conversion Group. These instructions perform symmetric decimal, floating-point, and integer conversions.

Table 17. Conversion Group

Instruction	Mnemonic	Opcode
Decimal to floating-point	DTOF	0x11
Floating-point to decimal	FTOD	0x12
Floating-point to integer	FTOI	0x0F
Integer to floating-point	ITOF	0x10

Transcendental Group. These instructions perform transcendental functions.

Table 18. Transcendental Group

Instruction	Mnemonic	Opcode
Sine	SIN	0x1C
Cosine	COS	0x1D
Arc tangent	ATAN	0x1E
Pi	PI	0x1F

Note: The worst case error for these functions in round nearest mode is equal to one unit in the last place for single- and double-precision. The worst case error for this function in other rounding modes is equal to two units in the last place for single- and double-precision. For double-extended-precision, the result is guaranteed to be at least as accurate as double-precision, and will usually be more accurate.

Miscellaneous Group. These instructions do not fall into any of the previous categories.

Table 19. Miscellaneous Group

Instruction	Mnemonic	Opcode
Extract result on fault	EROF	0x14
No operation	NOP	0x13

Instruction Set Summary by Mnemonic and Opcode

The *WE* 32206 Math Acceleration Unit (MAU) instruction set provides arithmetic, logical, data transfer, and conversion functions. Table 20 lists the instruction set mnemonics alphabetically, while Table 21 lists the instruction set opcodes numerically.

The *WE* 32200 Microprocessor allows up to two memory-based operands for each coprocessor protocol instruction. The MAU allows up to three memory-based operands so that it can be used with a CPU that supports three memory-based operand operations.

Unassigned opcodes cause behavior identical to that of the no operation (NOP) opcode.

Table 20. Instruction Set Summary by Mnemonic

Mnemonic	Opcode	Instruction	Operation	Condition Codes Affected
ABS	0x0C	Absolute value	Clear sign bit of operand 1 and copy operand 1 into operand 3. Operand 1 is unaffected.	N,Z,IO,PR,UO
ADD	0x02	Add	Add operand 1 to operand 2 and store result in operand 3.	N,Z,IO,PR,UO
ATAN	0x1E	Arc tangent	Compute arctangent of operand 1 and store in operand 3. Operand 1 must be in the range $-1 \leq x \leq 1$.	N,Z,IO,PR,UO
CMP	0x0A	Compare	Compare operand 1 to operand 2 and set condition codes. Operand 1 and operand 2 are not modified.	N,Z,IO,PR,UO
CMPE	0x0B	Compare with exceptions	Compare operand 1 to operand 2 and set condition codes. Operand 1 and operand 2 are not modified.	N,Z,IO,PR,UO
CMPES	0x1B	Compare with exceptions and flags swapped	Compare operand 1 to operand 2 and set condition codes. Operand 1 and operand 2 are not modified.	N,Z,IO,PR,UO
CMPS	0x1A	Compare with flags swapped	Compare operand 1 to operand 2 and set condition codes. Operand 1 and operand 2 are not modified.	N,Z,IO,PR,UO
COS	0x1D	Cosine	Compute cosine of operand 1. Operand 1 must be greater than $-\pi/2$ and less than $+\pi/2$	N,Z,IO,PR,UO
DIV	0x04	Divide	Divide operand 2 by operand 1 and store result in operand 3.	N,Z,IO,PR,UO
DTOF	0x11	Convert decimal to floating-point	Convert operand 1 (in decimal format) to floating point format and then copy it into operand 3. Operand 1 is unaffected.	N,Z,IO,PR,UO
EROF	0x14	Extract result on fault	Place value of data register in operand 3. Only execute this instruction when a fault condition occurs. If this instruction is executed under normal conditions, the value returned is indeterminate.	None
FTOD	0x12	Convert floating-point to decimal	Round operand 1 to an integral value and then convert it into decimal format and copy it into operand 3. Operand 1 is unaffected.	N,Z,IO,PR,UO

Table 20. Instruction Set Summary by Mnemonic (Continued)

Mnemonic	Opcode	Instruction	Operation	Condition Codes Affected
FTOI	0x0F	Convert floating-point to integer	Convert operand 1 from floating-point to 2's complement integer and copy it into operand 3.	N,Z,IO,PR,UO
ITOF	0x10	Convert integer to floating-point	Convert operand 1 from 2's complement integer format to floating-point and copy it into operand 3.	N,Z,IO,PR,UO
LDR	0x18	Load data register	Place operand 1 in data register.	Undefined
MOVE	0x07	Move	Copy operand 1 into operand 3. Operand 1 is unaffected.	N,Z,IO,PR,UO
MUL	0x06	Multiply	Multiply operand 1 by operand 2 and store result in operand 3.	N,Z,IO,PR,UO
NEG	0x17	Negate	Compliment sign of operand 1 and copy operand 1 into operand 3. Operand 1 is unaffected.	N,Z,IO,PR,UO
NOP	0x13	No operation	Cause any operation in progress to terminate without affecting any of the MAU internal registers.	None
PI	0x1F	Pi	Store the constant pi in operand 3.	N,Z,IO,PR,UO
RDASR	0x08	Move from ASR	Copy ASR into operand 3. The ASR is unaffected.	None
REM	0x05	Remainder	Divide operand 2 by operand 1 and store remainder in operand 3*.	N,Z,IO,PR,UO
RTOI	0x0E	Round to integral value	Round operand 1 to an integral value in floating point format and then copy it into operand 3. Operand 1 is unaffected.	N,Z,IO,PR,UO
SIN	0x1C	Sine	Compute sine of operand 1. Operand 1 must be greater than $-\pi/2$ and less than $+\pi/2$.	N,Z,IO,PR,UO
SQRT	0x0D	Square root	Compute square root of operand 1 and store in operand 3. Operand 1 is unaffected.	N,Z,IO,PR,UO
SUB	0x03	Subtract	Subtract operand 1 from operand 2 and store result in operand 3.	N,Z,IO,PR,UO
WRASR	0x09	Move to ASR	Copy operand 1 into ASR. Operand 1 is unaffected.	N,Z,IO,PR,UO

Note: If PR = 1, then the result is a partial remainder. To get the true remainder, repeat the operation with the previous result, Operand 3, replacing the previous dividend, Operand 2. Repeat until PR = 0, indicating a true remainder.

* Let N = INT(Operand 2/Operand 1) in round-to-nearest mode. Store the remainder (Operand 2 – (N x Operand 1)) in Operand 3.

Table 21. Instruction Set Summary by Opcode

Opcode	Mnemonic	Instruction
0x02	ADD	Add
0x03	SUB	Subtract
0x04	DIV	Divide
0x05	REM	Remainder
0x06	MUL	Multiply
0x07	MOVE	Move
0x08	RDASR	Move from ASR
0x09	WRASR	Move to ASR
0x0A	CMP	Compare
0x0B	CMPE	Compare with exceptions
0x0C	ABS	Absolute value
0x0D	SQRT	Square root
0x0E	RTOI	Round to integral value
0x0F	FTOI	Convert floating-point to integer
0x10	ITOF	Convert integer to floating-point
0x11	DTOF	Convert decimal to floating-point
0x12	FTOD	Convert floating-point to decimal
0x13	NOP	No operation
0x14	EROF	Extract result on fault
0x17	NEG	Negate
0x18	LDR	Load data register
0x1A	CMPS	Compare with flags swapped
0x1B	CMPES	Compare with exceptions and flags swapped
0x1C	SIN	Sine
0x1D	COS	Cosine
0x1E	ATAN	Arctangent
0x1F	PI	Pi

Definitions

Biased exponent. The sum of the exponent and a constant (bias) chosen to make the biased exponent's range nonnegative.

Binary floating-point number. A bit string characterized by three components; a sign, a signed exponent, and a significand. Its numerical value, if any, is the signed product of its significand and two raised to the power of its exponent.

Denormalized. A number is denormalized if its exponent field contains all zeros, its J (explicit) bit is zero, and its fraction field is nonzero.

Exponent. That component of a binary floating point number which normally signifies the power to which two is raised in determining the value of the represented number. Occasionally the exponent is called the signed or unbiased exponent.

Format's maximum (max). The value of a field such that all bits in the field are 1.

Format's minimum (min). The value of a field such that all bits in the field are 0.

Fraction. The field of the significand that lies to the right of its implied binary point.

NaN. Not a number; a symbolic entity encoded in floating point format. Operations that have no mathematical interpretation, such as zero divided by zero will produce a NaN. Such NaNs can be used to convey diagnostic information regarding their creation.

Neither max nor min (notmm). The value of a field such that all bits are neither all 1s nor all 0s.

Normal zero. The exponent and the significand are the format's minimum. Normal zero may have either a positive or a negative sign. Only the extended format has unnormalized zeros.

Normalized. If the number is nonzero, shift its significand to the left while decrementing its exponent until the leading significand bit becomes one; the exponent is regarded as if its range were unlimited. If the significand is zero, the number becomes normal zero. Normalizing a number does not change its sign.

Quiescent state. The MAU accepts and reacts to data read and write if chip select is asserted, and coprocessor broadcast accesses only. It accepts resets, does not read or write to any internal registers, and does not affect the rest of the system in any way.

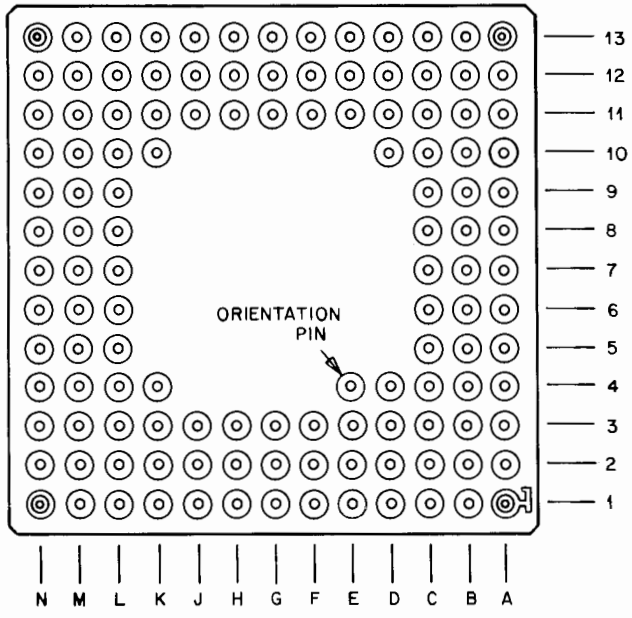
Significand. The component of a binary floating point number which consists of an explicit or implicit leading bit to the left of its binary point and a fraction field to the right of the binary point.

Unnormalized. Occurs only in the double extended format. The number's exponent is greater than the format's minimum and the explicit leading bit is zero. If the significand is zero this is an unnormalized zero.

Pin Descriptions

The *WE* 32206 Math Acceleration Unit is available in a 125-pin square, hermetic, ceramic PGA package. Figure 5 and Tables 22—27 describe the pin assignments.

The term *asserted* in the pin function descriptions means that the signal is driven to its active state either by the MAU (outputs) or by an external device (inputs). The term *negated* means that the signal is driven to its inactive state. A bar over the signal name (e.g., $\overline{DS}$) indicates the signal is active low, logic 0. The 0 bit is the least significant bit for signals which occupy two or more pins (e.g., SAS0). In the signal type column, I indicates an input, O an output, and I/O is a bidirectional signal.



Bottom View

Figure 5. WE® 32206 MAU 125-Pin Configuration

Numerical Order

Table 22. Pin Descriptions – 125-Pin Square PGA Package

Pin	Name	Type	Description
A1	DATA10	I/O	Data 10
A2	DATA13	I/O	Data 13
A4	DATA16	I/O	Data 16
A6	DATA18	I/O	Data 18
A8	DATA21	I/O	Data 21
A10	DATA24	I/O	Data 24
A11	DATA26	I/O	Data 26
A13	$\overline{\text{SRDY}}$	O	Synchronous ready
B2	DATA11	I/O	Data 11
B3	DATA15	I/O	Data 15
B5	DATA17	I/O	Data 17
B6	GND	—	Ground
B7	DATA19	I/O	Data 19
B8	DATA20	I/O	Data 20
B9	DATA22	I/O	Data 22
B10	DATA23	I/O	Data 23
B11	DATA25	I/O	Data 25
B12	DATA30	I/O	Data 30
C1	DATA09	I/O	Data 09
C2	GND	—	Ground
C3	DATA12	I/O	Data 12
C4	+5V	—	Power
C5	GND	—	Ground
C6	+5V	—	Power
C7	GND	—	Ground
C8	+5V	—	Power
C9	GND	—	Ground
C10	+5V	—	Power
C11	DATA27	I/O	Data 27
C12	GND	—	Ground
C13	DATA28	I/O	Data 28
D2	DATA08	I/O	Data 08
D3	+5V	—	Power
D4	DATA14	I/O	Data 14
D11	+5V	—	Power
D12	DATA29	I/O	Data 29
E1	DATA07	I/O	Data 07
E3	GND	—	Ground
E4	—	—	Device socket orientation pin
E11	GND	—	Ground
E12	GND	—	Ground
E13	DATA31	I/O	Data 31
F2	DATA05	I/O	Data 05
F3	+5V	—	Power
F11	+5V	—	Power
F12	CLK23	I	Input clock 23
G1	DATA06	I/O	Data 06
G3	GND	—	Ground
G11	GND	—	Ground
G12	CLK34	I	Input clock 34
G13	$\overline{\text{PIACK}}$	I	Peripheral interrupt acknowledge
H2	DATA02	I/O	Data 02
H3	+5V	—	Power
H11	+5V	—	Power
H12	$\overline{\text{DONE}}$	O	Coprocessor done
H13	$\overline{\text{PIREQ}}$	O	Peripheral interrupt request
J1	DATA04	I/O	Data 04
J2	GND	—	Ground
J3	GND	—	Ground
J11	GND	—	Ground
J12	DSIZE0	I	Data size 0
J13	$\overline{\text{FAULT}}$	O	Fault
K2	DATA01	I/O	Data 01
K3	+5V	—	Power
K11	DSIZE1	I	Data size 1
K12	ADDR03	I	Address 03
L1	DATA03	I/O	Data 03
L3	DATA00	I/O	Data 00
L4	+5V	—	Power
L5	GND	—	Ground
L6	+5V	—	Power
L7	GND	—	Ground
L8	+5V	—	Power
L9	GND	—	Ground
L10	+5V	—	Power
L11	ADDR04	I	Address 04
L12	DSIZE2	I	Data size 2

Table 22. Pin Descriptions – 125-Pin Square PGA Package (Continued)

Pin	Name	Type	Description
M2	$\overline{\text{DSHAD}}$	I	Data bus shadow
M3	GND	—	Ground
M4	SAS2	I	Access status code 2
M5	GND	—	Ground
M6	SAS1	I	Access status code 1
M7	$\overline{\text{DS}}$	I	Data strobe
M8	$\overline{\text{RESET}}$	I	Reset
M9	GND	—	Ground
M10	$\overline{\text{DTACK}}$	O	Data transfer acknowledge
M11	$\overline{\text{DRDY}}$	I	Data ready
M12	R/$\overline{\text{W}}$	I	Read/Write
N1	$\overline{\text{HIGHZ}}$	I	High impedance
N3	$\overline{\text{CYCLEI}}$	I	Cycle initiate
N5	SAS3	I	Access status code 3
N6	SAS0	I	Access status code 0
N8	$\overline{\text{CS}}$	I	Chip select
N9	NC	—	No connection
N11	NC	—	No connection
N13	ADDR02	I	Address 02

Functional Groups

Table 23. Address and Data Signals

Name	Pin(s)	Type	Description
DATA00–DATA31	L3, K2, H2, L1, J1, F2, G1, E1, D2, C1, A1, B2, C3, A2, D4, B3, A4, B5, A6, B7, B8, A8, B9, B10, A10, B11, A11, C11, C13, D12, B12, E13	I/O	**Data.** These pins provide a bidirectional bus to transmit data to and from the MAU.
ADDR02–ADDR04	N13, K12, L11	I	**Address.** These bits are used to select the peripheral mode registers when MAU is in peripheral mode.

Table 24. Interface and Control

Name	Pin	Type	Description
$\overline{\text{CS}}$	N8	I	**Chip Select.** Assertion puts MAU in peripheral mode.
$\overline{\text{CYCLEI}}$	N3	I	**Cycle Initiate.** Assertion indicates that CPU has started a bus cycle.
$\overline{\text{DONE}}$	H12	O	**Done.** Assertion indicates to CPU that MAU has completed execution of the current instruction.
$\overline{\text{DRDY}}$	M11	I	**Data Ready.** Assertion indicates to MAU that no bus exceptions have been detected during current bus cycle.
$\overline{\text{DS}}$	M7	I	**Data Strobe.** When low during a read operation, this signal indicates that MAU may place data on data bus. When low during a write operation, this signal indicates that MAU may remove data from data bus.
$\overline{\text{DSHAD}}$	M2	I	**Data Bus Shadow.** Assertion causes the MAU to 3-state the data bus if it was driving it.
$\overline{\text{DTACK}}$	M10	O	**Data Transfer Acknowledge.** Assertion signals CPU to end current access. Used for asynchronous handshaking between CPU and MAU in peripheral mode.

Table 24. Interace and Control (Continued)

Name	Pin	Type	Description
$\overline{\text{PIACK}}$	G13	I	**Peripheral Interrupt Acknowledge.** When asserted this signal negates the $\overline{\text{PIREQ}}$ signal.
$\overline{\text{PIREQ}}$	H13	O	**Peripheral Interrupt Request.** When asserted the MAU has completed an operation in peripheral mode. This signal is negated by the $\overline{\text{PIACK}}$ signal.
$\overline{\text{SRDY}}$	A13	O	**Synchronous Ready.** Assertion of this signal causes the CPU to terminate a coprocessor broadcast or status fetch.

Table 25. Status Signals

Name	Pin(s)	Type	Description
DSIZE0—DSIZE2	J12 K11 L12	I	**Data Size.** These pins indicate the size of a bus transaction from a *WE* 32200 CPU. DSIZE0 is the least significant bit of the data size code. **Bit 2,1,0** — **Description** 001 — 0 bytes 100 — Word 101 — Double word 110 — Halfword 111 — Byte
R/$\overline{\text{W}}$	M12	I	**Read/Write.** A 1 indicates a read from the MAU; a 0 indicates a write to the MAU.
SAS0—SAS3	N6, M6, M4, N5	I	**Access Status Codes.** These pins describe the type of bus cycle being executed. SAS0 is the least significant bit of the access status code. **Bit 3,2,1,0** — **Description** 0001 — Coprocessor data write 0010 — Autovector interrupt acknowledge 0011 — Coprocessor data fetch 0101 — Coprocessor broadcast 0110 — Coprocessor status fetch 1011 — Interrupt acknowledge

Table 26. Bus Exception Signals

Name	Pin	Type	Description
$\overline{\text{FAULT}}$	J13	O	**Fault.** Assertion indicates that an exception occurred during execution of an instruction in the MAU.
$\overline{\text{RESET}}$	M8	I	**Reset.** Assertion causes MAU to execute its reset routine.

Table 27. Clocks

Name	Pin	Type	Description
CLK34	G12	I	**Input Clock.** The falling edge of this clock signifies the beginning of a machine cycle. This clock input has the same frequency as CLK23 and lags it by 90°.
CLK23	F12	I	**Input Clock.** This clock input has the same frequency as CLK34 and leads it by 90°.

Table 28. Test

Name	Pin	Type	Description
$\overline{\text{HIGHZ}}$	N1	I	**High Impedance.** 3-states all output signals for testing.

Characteristics

V_{CC} = 5.0 V ± 5%, V_{SS} = 0 V, C_L = 130 pF, T_A = 0 to 70 °C

Timing Characteristics

The low voltage threshold is 0.8 V; the high voltage threshold is 2.0 V. All TTL timing specifications are referenced to or from these threshold voltages. CMOS clock references are to and from $V_{CC}/2$. All minimum and maximum values are in ns.

Table 29. Timing Characteristics

Num	Symbol	Description	Figure(s)	24 MHz	
				Min	Max
1	tC34LSASV	Access status code set-up time	7,8,9,10,11	3	—
2	tC34LSASX	Access status code hold time	7,8,9,10,11	3	—
3	tC23LCYCL	Cycle initiate set-up time	7,8,9,10,11	3	—
4	tC23LCYCX	Cycle initiate hold time	7,8,9,10,11	3	—
5	tC23HDATV	Data set-up time (ID test)	7	21	—
6	tC23HDATX	Data hold time (ID test)	—	3	—
7	tC23HDRYV	Data ready set-up time	7,8,9,10	3	—
8	tC23HDRYX	Data ready hold time	7,8,9,10	3	—
9	tC34HDATV	Data set-up time	8,13	3	—
10	tC34HDATX	Data hold time	8,13	9	—
13	tC23HCSL	Chip select set-up time *	12,13	3	—
14	tDSBHCSX	Chip select hold time	12,13	8	—
15	tC23HRWH	Read/write strobe set-up time *	12,13	3	—
16	tDSBHRWX	Read/write strobe hold time	12,13	3	—
17A	tC23HDSBL	Data strobe set-up time	7,8,9,10	3	—
17B	tC23HDSBL	Data strobe set-up time *	12,13	3	—
18	tC23HDSBX	Data strobe hold time	7,8,9,10, 12,13	3	—

Table 29. Timing Characteristics (Continued)

Num	Symbol	Description	Figure(s)	24 MHz	
				Min	Max
19	tDSHLDATZ	Data 3-state time	11	—	29
20	tDSHHDATV	Data valid time	11	—	29
21	tC33LRSRL	Reset set-up time *	—	3	—
22	tRSRLRSRH	Reset valid time (to guarantee reset)	—	8TC	—
23	tC23HADDV	Address set-up time *	12,13	3	—
24	tDSBHADDX	Address hold time	12,13	3	—
25	tC34LDSZV	Data size set-up time	—	3	—
26	tC34LDSZX	Data size hold time	—	3	—
27	tPACLASBV	$\overline{\text{PIACK}}$ assertion time	—	13	—
40	tC23HSRYL	Synchronous ready assertion time	7,9	—	23
41	tC23HSRYZ	Synchronous ready 3-state time	7,9	—	23
42	tC23HFATV	Fault assertion time	9	—	23
43	tC23HFATZ	Fault 3-state time	9	—	23
44	tC34HDONL	Done assertion time	9	—	23
45	tC34HDONZ	Done 3-state time	9	—	23
47	tC34HDATV	Data assertion time	9,10,11	—	23
48	tC34LDATZ	Data 3-state time	9,10	—	23
49	tHIZOUTZ	Outputs 3-state time	—	—	23
50	tC23LDTAL	Data acknowledge assertion time	12,13	—	23
51	tDSBHDTAZ	Data acknowledge 3-state time	12,13	—	24
52	tC23HDATV	Data assertion time	12	—	23
53	tDSBHDATZ	Data 3-state time	12	—	23
54	tC34LPIRV	Peripheral interrupt request assertion time	—	—	23
55	tPIALPIRZ	Peripheral interrupt request 3-state time	—	—	29
65	tC23J	Clock 23 jitter time	—	—	0.5
66	tC23E	Clock 23 duty cycle error time**	—	—	2
67	tC23L1C23HZ	Clock 23 rise time	6	—	4
68	tC23HZC23L1	Clock 23 fall time	6	—	4
69	tC23L1C23L1	Clock 23 period (nominal)	6	41	—
70	tC34J	Clock 34 jitter time	—	—	0.5
71	tC34E	Clock 34 duty cycle error time**	—	—	2

* This is an asynchronous signal. Set-up times are specified for testing and informational purposes.

Table 29. Timing Characteristics (Continued)

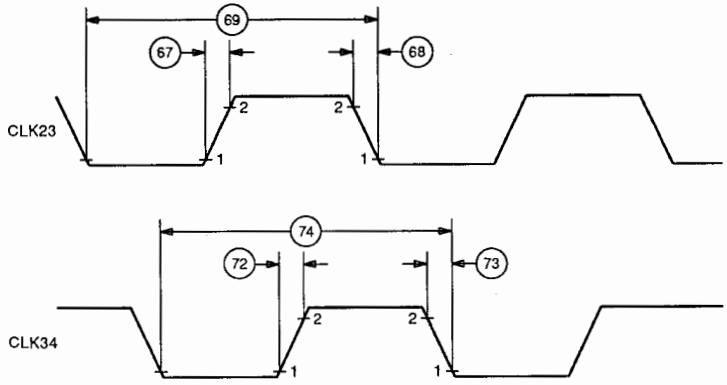
Num	Symbol	Description	Figure(s)	24 MHz	
				Min	Max
72	tC34L1C34HZ	Clock 34 rise time	6	—	4
73	tC34H2C34L1	Clock 34 fall time	6	—	4
74	tC34HC34H	Clock 34 period (nominal)	6	41	—
74	tSKEW	Clock skew	—	—	2

* This is an asynchronous signal. Set-up times are specified for testing and informational purposes.

**At 70 °C ambient and Vcc = 4.75 V, the allowable clock skew and clock duty cycle errors are shown for each operating frequency. These skew and error specs do not increase by more than 1 ns per 30 C° and 1 ns per 250 mV away from this operating point and do not exceed the specified value throughout the full operating range.

Timing Diagrams

These timing diagrams represent a subset of all possible transactions and are intended only for the purposes of displaying and clarifying timing relationships.



Notes:
Duty Cycle Error -- The duty cycle of each clock input may deviate from 50% but should not exceed timing specification numbers 66 and 71 for CLK23 and CLK34, respectively.
Skew – CLK23 nominally leads CLK34 by 90° (1/4 clock period). This phase lead should never exceed timing specification number 74.
Jitter – The period of each clock input may deviate from its nominal value but should not exceed timing specification numbers 65 and 70 for CLK23 and CLK34, respectively.

Figure 6. Clock Inputs

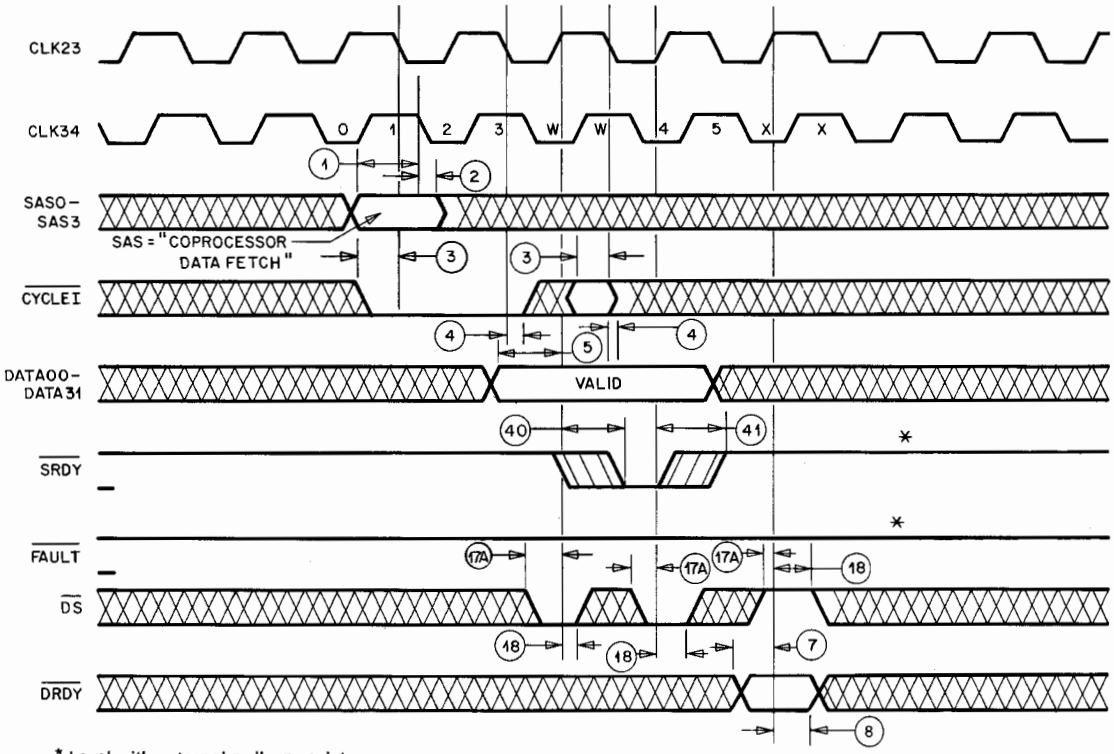
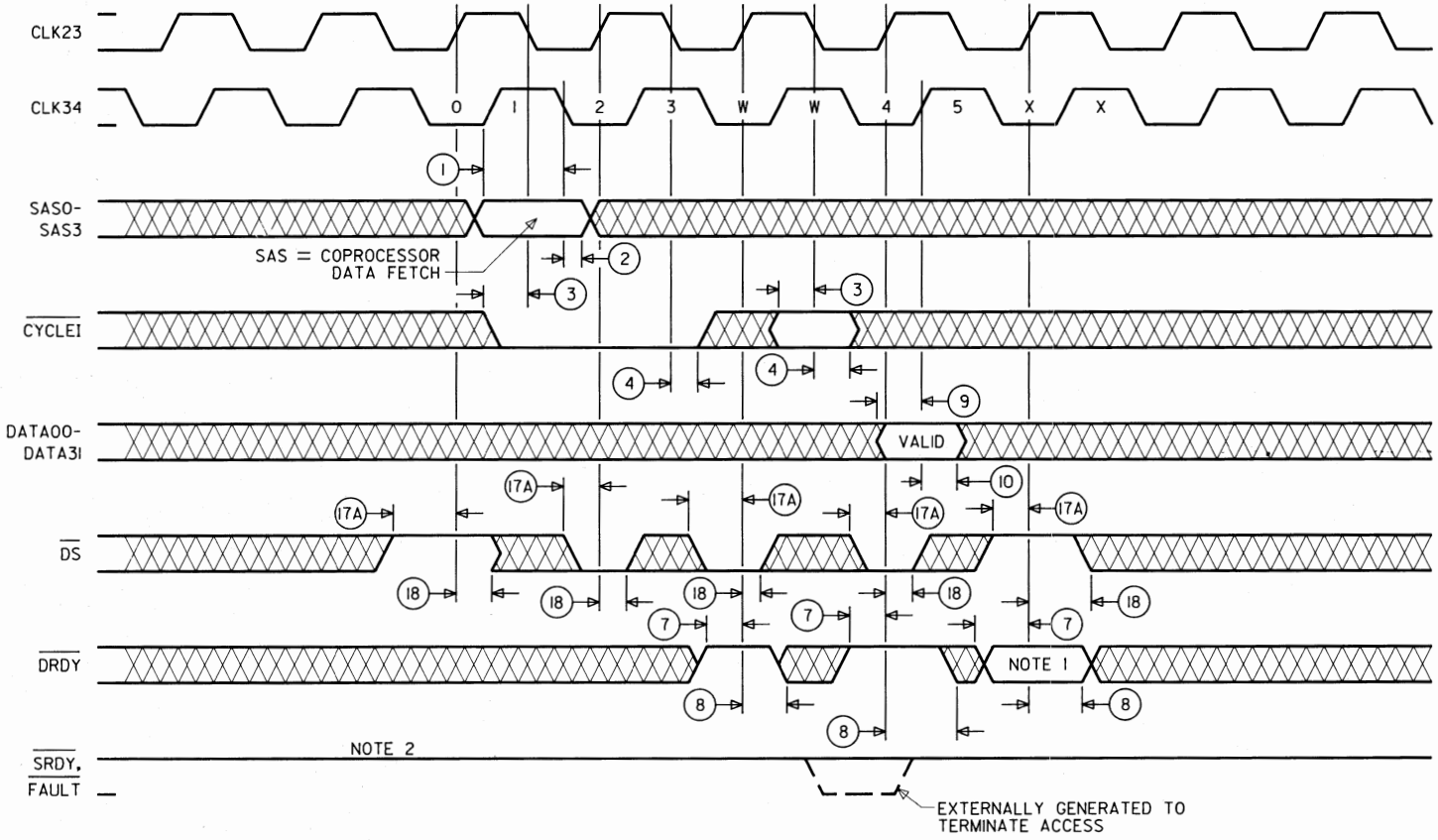


Figure 7. Coprocessor Broadcast (1 Wait-State)



[1]Faulted access = Logic 1. No fault= Logic 0.
[2]Level with external pull-up resistor

Figure 8. Coprocessor Data Fetch (1 Wait-State)

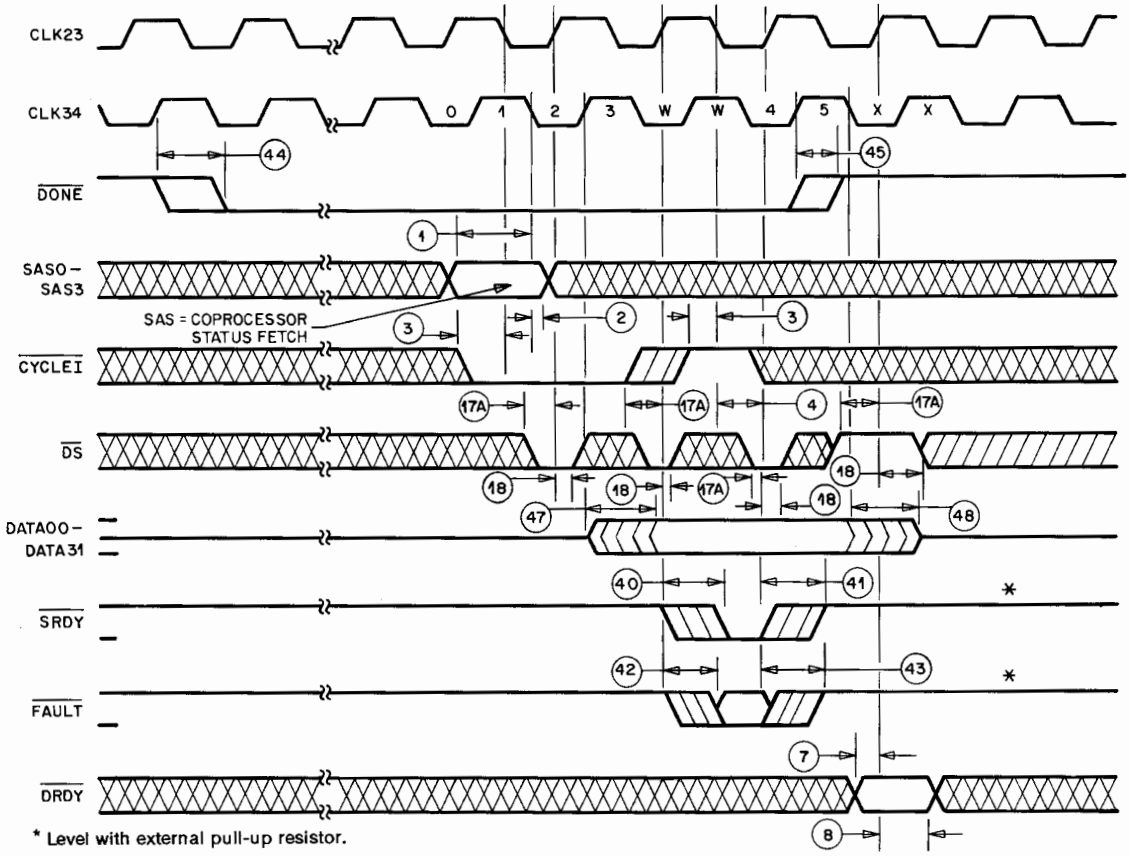


Figure 9. Coprocessor Status Fetch

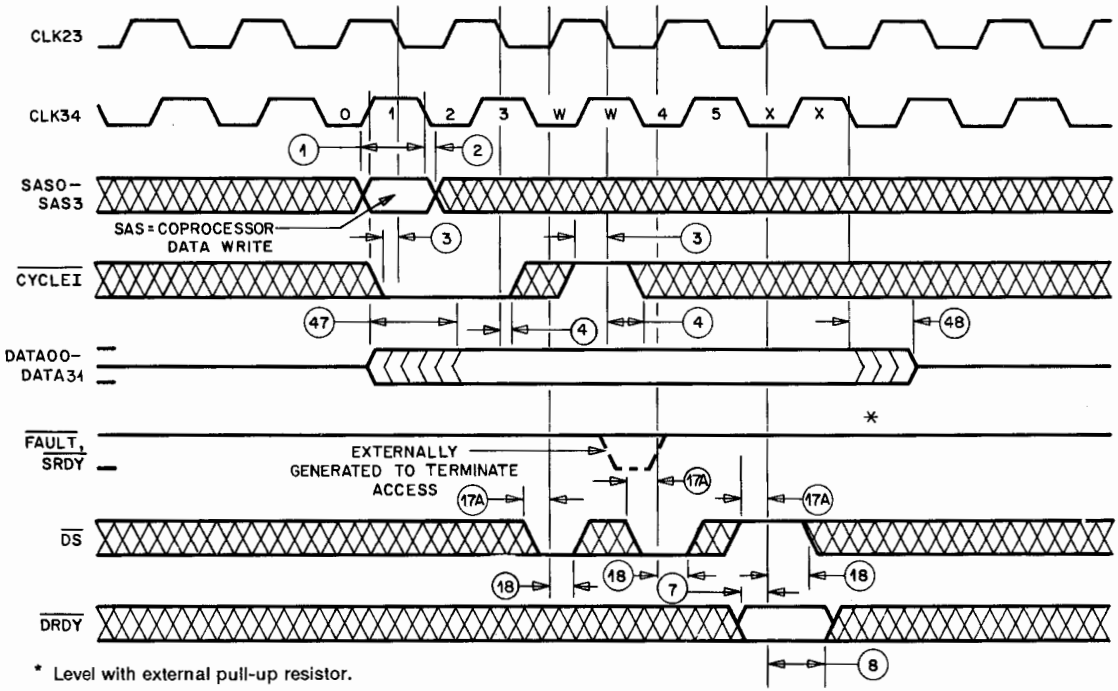


Figure 10. Coprocessor Data Write (1 Wait-State)

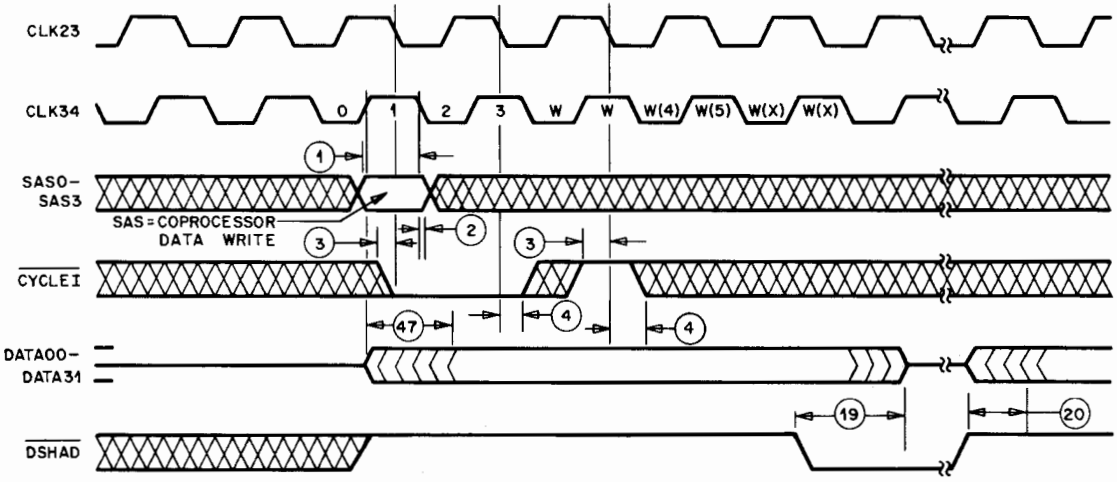


Figure 11. Coprocessor Data Write ($\overline{\text{DSHAD}}$ Assertion)

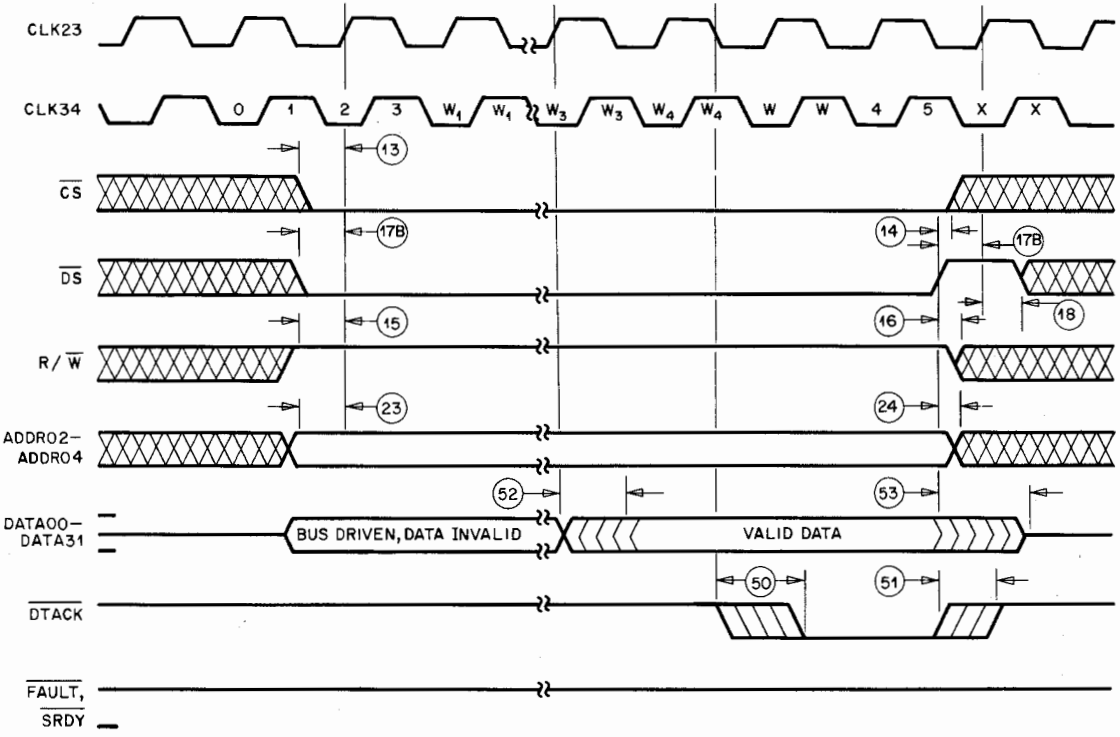


Figure 12. Peripheral Mode Read

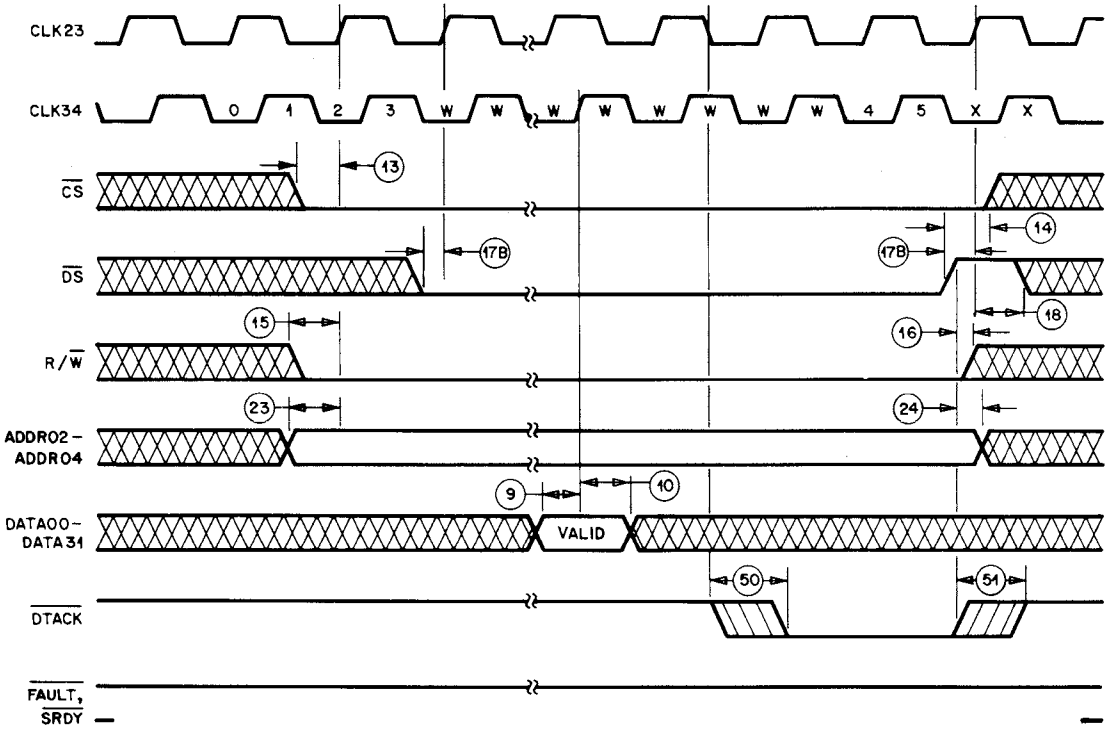


Figure 13. Peripheral Mode Write

Electrical Characteristics

Inputs

All inputs, except the CMOS input clocks, are TTL compatible.

Table 30. DC Input Parameters

Inputs		Min	Nom	Max	Unit
TTL input voltage	high-level	2	—	V_{CC}	V
	low-level	0	—	0.8	V
CMOS clocks input voltage	high-level	V_{CC} – 1.3	—	V_{CC}	V
	low-level	0	—	0.8	V
TTL input loading current: for V_{IH} ($2.0\ V \leq V_{IH} \leq V_{CC}$)	high-level	0	—	0.01	mA
for V_{IL} ($0\ V \leq V_{IL} \leq 0.8\ V$)	low-level	–0.01	—	0	mA
CMOS clocks input loading current: for V_{IH} ($V_{CC} - 1.3\ V \leq V_{IH} \leq V_{CC}$)	high-level	0	—	0.01	mA
for V_{IL} ($0 \leq V_{IL} \leq 0.8\ V$)	low-level	–0.01	—	0	mA

Outputs

Listed below are descriptions of the outputs assigned to classes 1 and 2.

Class 1: This class is capable of driving one TTL load or 8 PNP Schottky TTL loads and has current allowance for an external holding resistor employed in 3-state buffers. The minimum holding resistor value is 2.7 kΩ.

Class 2: The signal in this class is an open drain output used for wired logic operations, allowing more than one device to drive a node without conflict. An external resistor is required to pull this signal high. The minimum pull-up resistor value is 680 Ω.

The following lists the outputs assigned to Classes 1 and 2.

Class 1	Class 2
DATA00—DATA31	$\overline{\text{DONE}}$
	$\overline{\text{SRDY}}$
	$\overline{\text{FAULT}}$
	$\overline{\text{DTACK}}$
	$\overline{\text{PIREQ}}$

Table 31. DC Output Parameter

Outputs		Min	Nom	Max	Unit
Output sink current (I_{OL}) ($V_{OL} \leq 0.4$ V)	Class 1	—	—	3.5	mA
	Class 2	—	—	12	mA
Output source current (I_{OH}) ($V_{OH} \geq 2.4$ V)	Class 1	—	—	3.5	mA
	Class 2	—	—	10	μA
Output logic levels*	high-level	2.4	—	—	V
	low-level	—	—	0.4	V

* Referenced to system ground (GND).

Operating Conditions

Parameter		Symbol	Min	Nom	Max	Unit
Supply voltage		V_{CC}	4.75	5.00	5.25	V
Input load capacitance	TTL Inputs	C_{IN}	—	—	12	pF
	CMOS Clocks		—	—	7	pF
Total output load capacitance	Class 1	C_L	—	—	130	pF
	Class 2		—	—	130	pF
Ambient temperature at the microprocessor pins		T_A	0	—	70	°C
Humidity range		—	5%	—	95%	RHNC*
Power dissipation	at 24 MHz	PD	—	—	1.8	W
Operating frequency		F	24	—	—	MHz

*Relative humidity, noncondensing.

WE® 32-Bit Microprocessors and Peripherals Packaging

Description

The *WE* 32-Bit Microprocessors and Peripherals (excluding the clock) are available in regular and extended-temperature pin grid array (PGA) packages. The *WE* 32102 Clock is available in a 16-pin hermetically-sealed metal can (double-width, dual-in-line package). Table 1 lists the devices and the package for each.

Table 1. Device Packages

Package	Figure	Device
16-pin metal can	1	32102
125-pin PGA	2	32100, 32101, 32103, 32106, 32206
133-pin PGA	2	32104, 32200, 32201, 32204

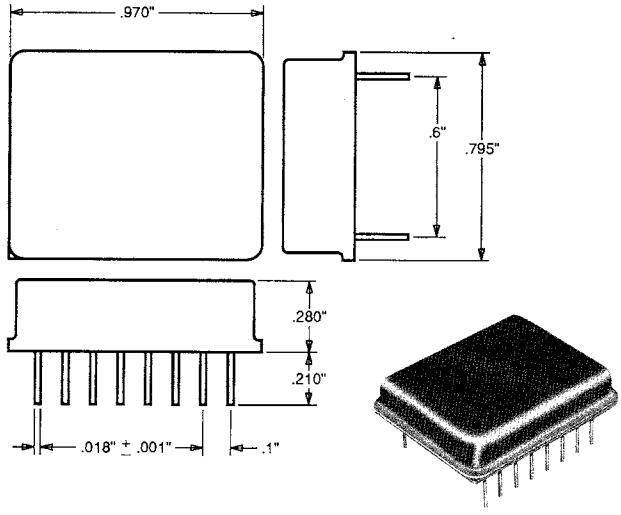


Figure 1. 16-Pin Metal Can Package Photograph and Outline

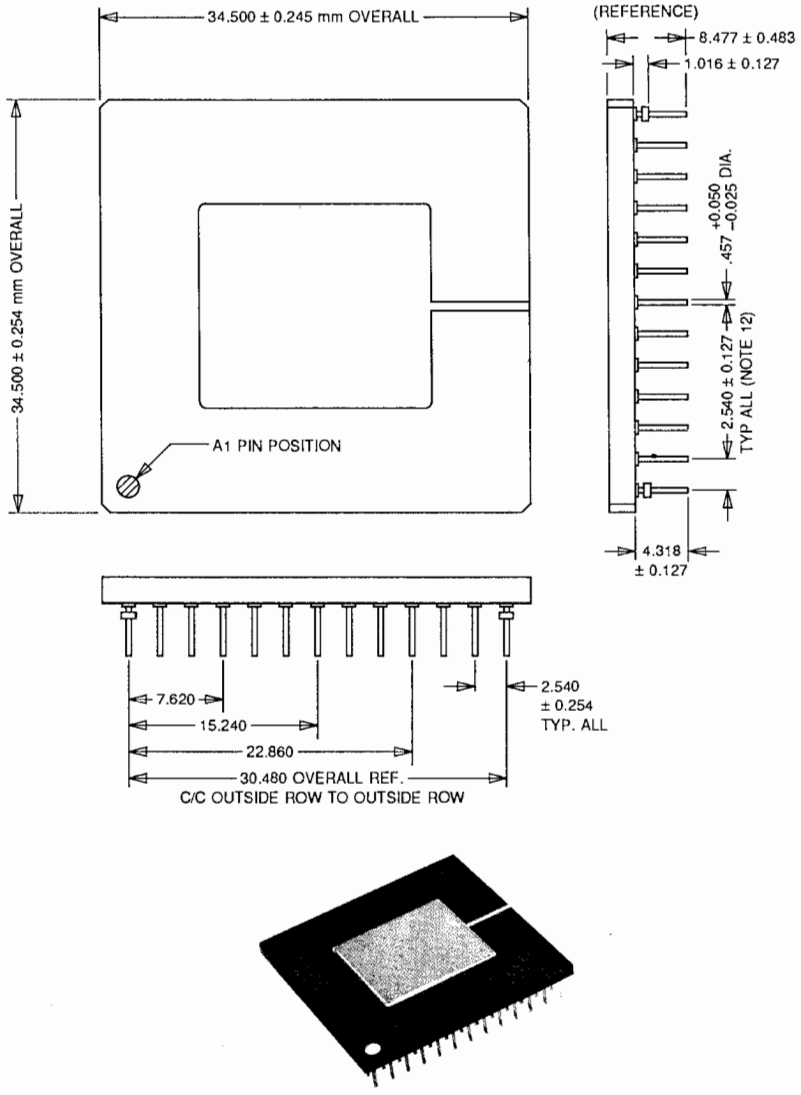


Figure 2. 125- and 133-Pin PGA Package Photograph and Outline

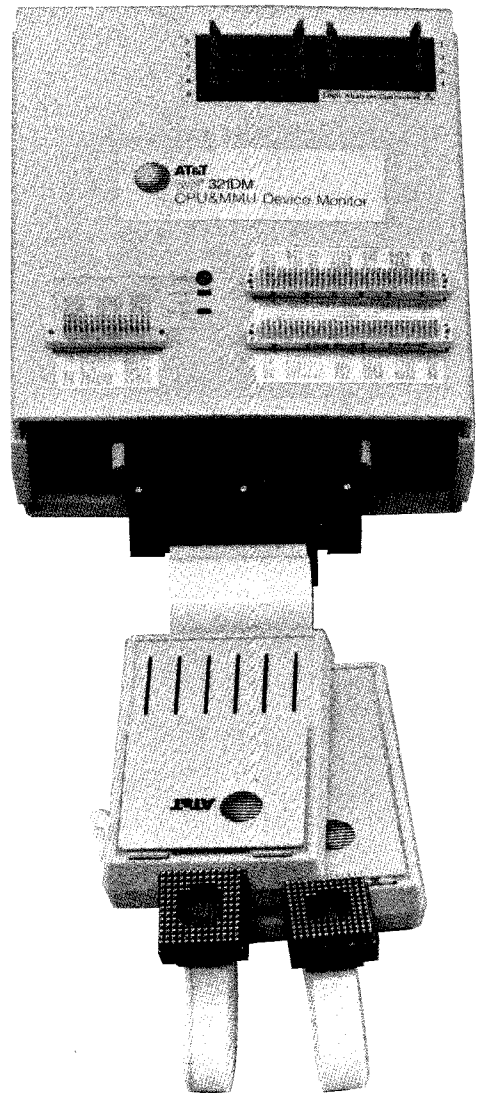
Chapter 3

Development Support Tools

WE® 321DM CPU & MMU Device Monitor

Description

The *WE* 321DM CPU & MMU Device Monitor provides a low cost solution for debugging *WE* 32100 Microprocessor and *WE* 32101 Memory Management Unit based applications operating at frequencies up to 18 MHz, with a logic analyzer that is limited to a maximum sample rate of 10 MHz. The device monitor supports zero wait-state block fetch accesses at frequencies as high as 18 MHz.



Features

- Signal observation of target systems operating at frequencies up to 18 MHz with zero wait states
- Eight logic analyzer access connectors (one connector is user definable)
- Timing connectors for direct access to all CPU and MMU signals
- Software support package for the Hewlett-Packard 64000 Logic Development System
- Supports block fetch accesses at target system frequencies as high as 18 MHz
- Switch selected transparent or latched operation mode
- Switch selected single or multiple sampling points per bus cycle

User Information

Hardware Overview

The device monitor consists of a logic retiming unit, a CPU cable system, an MMU cable system, and seven 25-wire ribbon cables, which provide a standard interface to a logic analyzer. The device monitor allows signal observation of high-speed target systems with a logic analyzer that is limited to 10-MHz sampling rates by:

- Generating logic analyzer sample points based on bus cycle activity and guaranteeing a minimum of 100 ns between sample points at target system operating frequencies as high as 18 MHz.
- Sampling and holding signals as necessary to assure 30 ns signal set-up and 0 ns hold times with respect to selected sampling points and a target system operating frequency up to 18 MHz.
- Providing the flexibility of observing a narrow window of bus activity in detail or observing a larger window of bus activity with reduced detail but with a greater perspective of system bus activity.

Logic Retiming Unit

The logic retiming unit is responsible for generating signal sampling points for the logic analyzer at guaranteed intervals of at least 100 ns while the target system is operating at a frequency of up to 18 MHz.

The logic retiming unit contains two slide switches for selecting the operating mode. One selects either transparent (nonlatched) mode or latched operating mode. In the latched mode, the other slide switch selects either single or multiple sample points per bus cycle.

Transparent (Nonlatched) Mode. Transparent mode is used for target system operating frequencies $\leq$ 10 MHz. Data samples can be taken every clock cycle and all signals can be captured.

Latched Mode. Latched mode is used for target system operating frequencies > 10 MHz. The latch guarantees that the signals are held long enough to ensure accurate data sampling.

The logic retiming unit also contains an LED power indicator, three timing access connectors, eight logic analyzer access

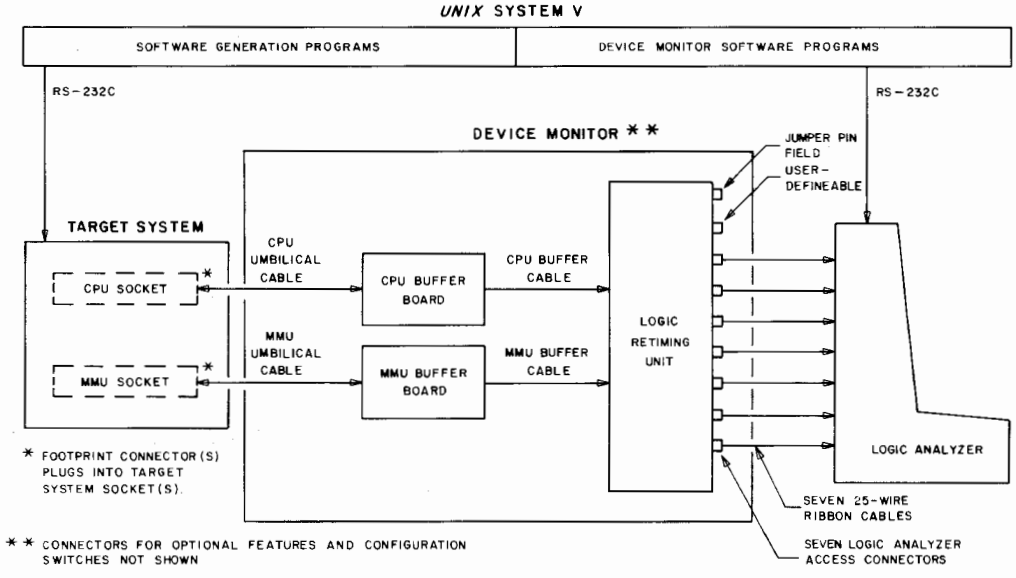


Figure 1. Device Monitor Configuration

connectors (one is user-definable), and a jumper pin field for monitoring any user-selected signal from the target system or timing access connectors.

CPU and MMU Cable Systems

The CPU and MMU cable systems contain footprint connectors that replace the CPU and MMU devices in the target system. The device monitor can be used whether or not the target system contains an MMU. The footprint connectors plug directly into the CPU and MMU sockets in the target system. The footprint connectors are linked via 6-inch umbilical cables to two buffer boards (one for the CPU and one for the MMU). The device(s) replaced in the target system by the footprint connector(s) plugs directly into a zero insertion force socket(s) on the buffer board(s). The buffer boards connect via 12.5 inch cables to the logic retiming unit. The overall length of each cable system is 24.5 inches.

Timing Access Connectors

The CPU and MMU signals can be observed before entering the logic circuitry of the logic retiming unit. Three timing access connectors located on the top of the logic retiming unit are provided for this purpose. Two of the connectors provide access to the CPU signals. The other connector provides access to the MMU signals. Spade adapter connectors are provided which, when plugged into the access connectors, provide a pin field for easy timing probe connection. The pinouts of the timing access connectors are labeled on the top of the logic retiming unit housing.

Logic Analyzer Access Connectors

Nine 26-pin male connectors are located on top of the logic retiming unit. Seven of these provide a standard interface for a logic analyzer. Of the remaining connectors, one is user-definable and connects pin-to-pin to the other connector, a jumper pin field. This pin field is used to monitor any user-chosen signal from the timing access connectors or target system. Seven 25-wire ribbon cables are provided to connect the logic analyzer to the logic analyzer access connectors.

Power Sources

The CPU and MMU devices on the two cable system buffer boards are powered by the target system. The logic retiming unit and the buffers and latches on the buffer boards are powered by a power supply in the logic retiming unit. An LED indicates when power is applied to the logic retiming unit.

Logic Analyzer

A Hewlett-Packard 64000 Logic Development System is the recommended logic analyzer for signal observation. The 64000 System makes full use of the development features provided by the device monitor. Other logic state analyzers may be used, e.g., the 64000 System 25-MHz state analyzer, by connecting user-supplied adapter cables between the state data and clock pods of the logic analyzer and the logic analyzer access connectors on the logic retiming unit of the device monitor (see Figure 1).

Software Overview

The *WE* 321DM-SP Device Monitor Software Programs supplied with the device monitor provide symbolic tracing and disassembly with the 64000 System. This software includes three program trace disassemblers, three configuration files, 64000 System upload and download utilities, a symbol table utility, and a communication file for communicating with the host *UNIX* System from the 64000 console. The program trace disassemblers allow hardware bus traces obtained by using the 64000 System to be displayed in *WE* 32100 Microprocessor assembly language format. The configuration files assign descriptive names to the signals available at the logic analyzer access connectors, providing easy signal recognition when performing signal observation. The upload utility provides test result uploading from the 64000 System to the host *UNIX* System. The download utility provides symbolic definition downloading from the host *UNIX* System to the 64000 System for symbolic debugging. The symbol table utility translates object file symbolic information for use by the 64000 System for symbolic debugging.

WE® 321DM CPU & MMU Device Monitor

The following hosts (running *UNIX* System V) are supported:

- *AT&T* 3B2 Computers
- *WE* 321SB VMEbus Single Board Computer
- *VAX* 11/780 Computer.

System Configuration

- Standard
  - ☐ Logic retiming unit
  - ☐ CPU cable system
  - ☐ MMU cable system
  - ☐ Seven 25-wire ribbon cables
  - ☐ *WE* 321DM-SP Device Monitor Software Programs on tape or floppy disk with documentation.
- User-Supplied
  - ☐ ASCII terminal
  - ☐ Host computer running *UNIX* System V Release 2
  - ☐ Hewlett-Packard 64000 Logic Development System or equivalent logic analyzer
  - ☐ *WE* 32100 Microprocessor based target system containing 125-pin square PGA socket(s)
  - ☐ *WE* 321SG Software Generation Programs on tape or floppy disk with documentation.

Advance

WE® 322DM CPU & MMU Device Monitor

Description

The *WE* 322DM CPU & MMU Device Monitor provides a low cost solution for debugging *WE* 32200 Microprocessor and *WE* 32201 Memory Management Unit based applications operating at frequencies of up to 24 MHz, with a logic analyzer that is limited to a maximum sample rate of 10 MHz. The device monitor supports zero wait-state block fetch accesses at frequencies as high as 24 MHz.

Features

- Signal observation of target systems operating at frequencies of up to 24 MHz with zero wait states
- Eight logic analyzer access connectors (one connector is user definable)
- Timing connectors for direct access to all CPU and MMU signals
- Software support package for the Hewlett-Packard 64000 Logic Development System
- Supports block fetch accesses at target system frequencies as high as 24 MHz
- Switch selected transparent or latched operation mode
- Switch selected single or multiple sampling points per bus cycle

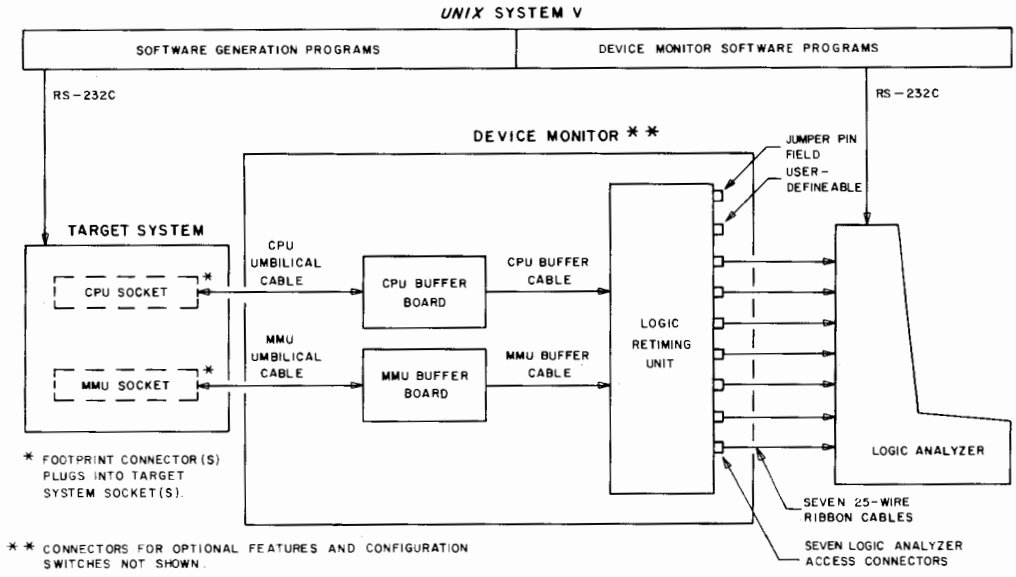


Figure 1. Device Monitor Configuration

User Information

Hardware Overview

The device monitor consists of a logic retiming unit, a CPU cable system, an MMU cable system, and seven 25-wire ribbon cables, which provide a standard interface to a logic analyzer.

The device monitor allows signal observation of high-speel target systems with a logic analyzer that is limited to 10-MHz sampling rates by:

- Generating logic analyzer sample points based on bus cycle activity and guaranteeing a minimum of 100 ns between sample points at target system operating frequencies as high as 24 MHz.
- Sampling and holding signals as necessary to assure 30 ns signal set-up and 0 ns hold times with respect to selected sampling points and a target system operating frequency of up to 24 MHz.
- Providing the flexibility of observing a narrow window of bus activity in detail or observing a larger window of bus activity with reduced detail but with a greater perspective of system bus activity.

Logic Retiming Unit

The logic retiming unit is responsible for generating signal sampling points for the logic analyzer at guaranteed intervals of at least 100 ns while the target system is operating at a frequency of up to 24 MHz.

The logic retiming unit contains two slide switches for selecting the operating mode. One selects either transparent (nonlatched) mode or latched operating mode. In the latched mode, the other slide switch selects either single or multiple sample points per bus cycle.

Transparent (Nonlatched) Mode. Transparent mode is used for target system operating frequencies $\leq$ 10 MHz. Data samples can be taken every clock cycle and all signals can be captured.

Latched Mode. Latched mode is used for target system operating frequencies > 10 MHz. The latch guarantees that the signals are held long enough to ensure accurate data sampling.

The logic retiming unit also contains an LED power indicator, three timing access connectors, eight logic analyzer access connectors (one is user-definable), and a jumper pin field for monitoring any user-selected signal from the target system or timing access connectors.

CPU and MMU Cable Systems

The CPU and MMU cable systems contain footprint connectors that replace the CPU and MMU devices in the target system. The device monitor can be used whether or not the target system contains an MMU. The footprint connectors plug directly into the CPU and MMU sockets in the target system. The footprint connectors are linked via 6-inch umbilical cables to two buffer boards (one for the CPU and one for the MMU). The device(s) replaced in the target system by the footprint connector(s) plugs directly into a zero insertion force socket(s) on the buffer board(s). The buffer boards connect via 12.5 inch cables to the logic retiming unit. The overall length of each cable system is 24.5 inches.

Timing Access Connectors

The CPU and MMU signals can be observed before entering the logic circuitry of the logic retiming unit. Three timing access connectors located on the top of the logic retiming unit are provided for this purpose. Two of the connectors provide access to the CPU signals. The other connector provides access to the MMU signals. Spade adapter connectors are provided which, when plugged into the access connectors, provide a pin field for easy timing probe connection. The pinouts of the timing access connectors are labeled on the top of the logic retiming unit housing.

Logic Analyzer Access Connectors

Nine 26-pin male connectors are located on top of the logic retiming unit. Seven of these provide a standard interface for a logic analyzer. Of the remaining connectors, one is user-definable and connects pin-to-pin to the other connector, a jumper pin field. This pin field is used to monitor any user-chosen signal from the timing access connectors or target system. Seven 25-wire ribbon cables are provided to connect the logic analyzer to the logic analyzer access connectors.

Power Sources

The CPU and MMU devices on the two cable system buffer boards are powered by the target system. The logic retiming unit and the buffers and latches on the buffer boards are powered by a power supply in the logic retiming unit. An LED indicates when power is applied to the logic retiming unit.

Logic Analyzer

A Hewlett-Packard 64000 Logic Development System is the recommended logic analyzer for signal observation. The 64000 System makes full use of the development features provided by the device monitor. Other logic state analyzers may be used, e.g., the 64000 System 25-MHz state analyzer, by connecting user-supplied adapter cables between the state data and clock pods of the logic analyzer and the logic analyzer access connectors on the logic retiming unit of the device monitor (see Figure 1).

Software Overview

The *WE* 322DM-SP Device Monitor Software Programs supplied with the device monitor provide symbolic tracing and disassembly with the 64000 System. This software includes three program trace disassemblers, three configuration files, 64000 System upload and download utilities, a symbol table utility, and a communication file for communicating with the host *UNIX* System from the 64000 console. The program trace disassemblers allow hardware bus traces obtained by using the 64000 System to be displayed in *WE* 32200 Microprocessor assembly language format. The configuration files assign descriptive names to the signals available at the logic analyzer access connectors, providing easy signal recognition when performing signal observation. The upload utility provides test result uploading from the 64000 System to the host *UNIX* System. The download utility provides symbolic definition downloading from the host *UNIX* System to the 64000 System for symbolic debugging. The symbol table utility translates object file symbolic information for use by the 64000 System for symbolic debugging.

The following hosts (running *UNIX* System V) are supported:

- *AT&T* 3B2 Computers
- *VAX* 11/780 Computer.

System Configuration

- Standard
  - ☐ Logic retiming unit
  - ☐ CPU cable system
  - ☐ MMU cable system
  - ☐ Seven 25-wire ribbon cables
  - ☐ *WE* 321DM-SP Device Monitor Software Programs on tape or floppy disk with documentation.
- User-Supplied
  - ☐ ASCII terminal
  - ☐ Host computer running *UNIX* System V Release 2
  - ☐ Hewlett-Packard 64000 Logic Development System or equivalent logic analyzer
  - ☐ *WE* 32200 Microprocessor based target system containing 133-pin square PGA socket(s).

WE® 321DS Microprocessor Development System

Description

The *WE* 321DS Microprocessor Development System is an integrated hardware and software system used to develop, test, and debug *WE* 32100 Microprocessor based applications. It consists of items that can be ordered individually. The main component of the development system is the *WE* 321AP Microprocessor Analysis Pod. The analysis pod emulates the *WE* 32100 Microprocessor, *WE* 32101 Memory Management Unit, and *WE* 32102 Clock functions in the target system under development, and it is essential for diagnosing target hardware and software problems. The analysis pod includes IMP, an interactive monitor program for assembly-level debugging. The development system also includes the *WE* 321SD Development Software Programs, a package of *UNIX* System based software that runs on a *AT&T* 3B2/300 or 3B2/400 Computer, or a *VAX* 11/780 Computer. This package includes utility programs for downloading from a host system using IMP. Options for this package include: Ferret, a C-level symbolic debugger, and bus state analysis software that supports logic analysis with the Hewlett-Packard 64000 Logic Development System.

Features

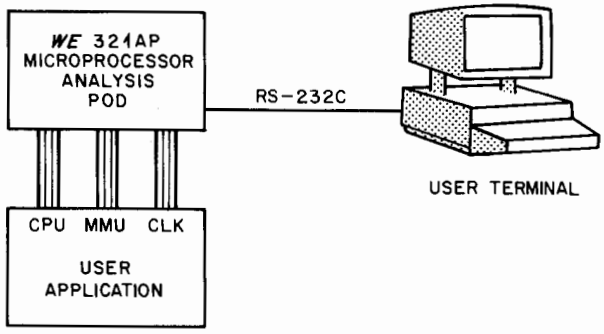
- Emulation of the *WE* 32100 Microprocessor, *WE* 32101 Memory Mangement Unit, and *WE* 32102 Clock in the target system with no wait states
- 52 Kbytes of static RAM available to user
- Assembly-level debugger and optional C-level symbolic debugger
- Physical and virtual mode debugging capability
- Memory map option switch to select pod-based or target-based interrupt and exception pointers
- Hardware and software breakpointing
- Math acceleration unit support
- Two RS-232C serial I/O ports
- Logic analyzer access connectors
- Stand-alone pod that can operate without additional support
- Target signal isolation
- Reset and abort pushbuttons
- On-board or external clock
- Eight-character alphanumeric LED display
- System self-test firmware
- Optional software support package for the Hewlett-Packard 64000 Logic Development System (logic analyzer)

User Information

The *WE* 321DS Microprocessor Development System can operate in any of the four configurations. All four configurations require a terminal (system console) for user interaction with the debuggers and a 5 V power supply for the analysis pod.

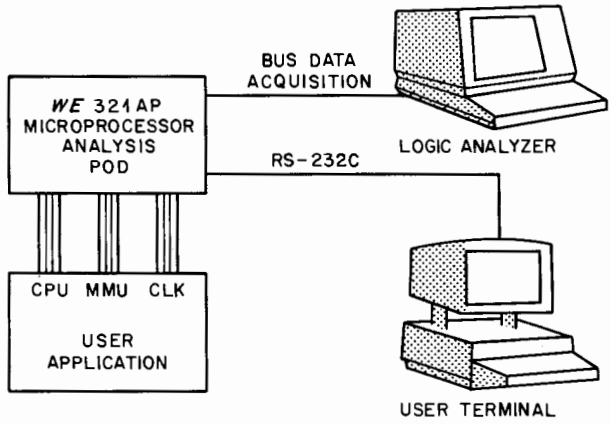
System Configuration

Microprocessor Analysis Pod Stand-Alone (Figure 1). The *WE* 321AP Microprocessor Analysis Pod is used in the stand-alone configuration during early hardware debugging stages. In this configuration, the analysis pod provides in-circuit emulation of the *WE* 32100 Microprocessor, *WE* 32101 Memory



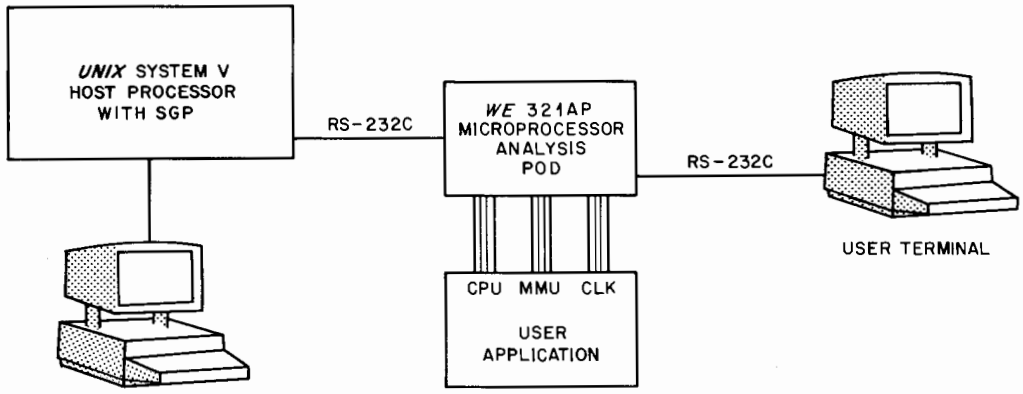
- Real-time emulation, no wait states
- Basic hardware debug
- Assembly-level software debug

Figure 1. Stand-Alone Configuration



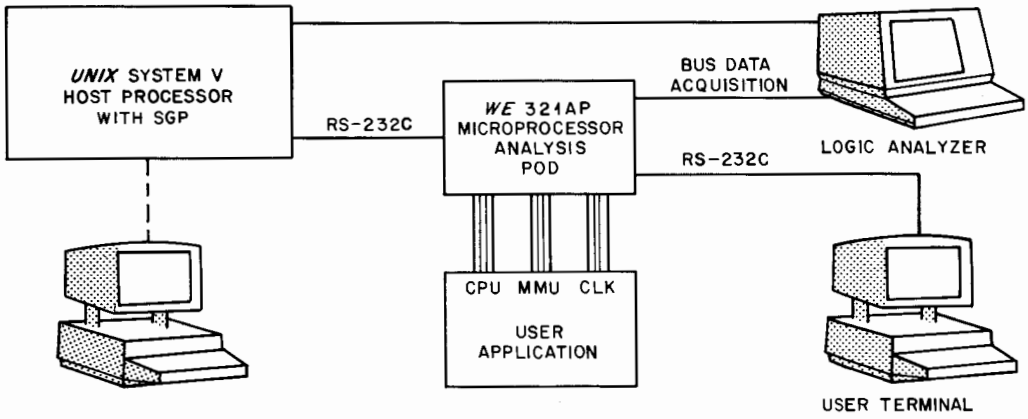
- Monitor all CPU, MMU, and clock signals
- Synchronous target system analysis
- Integrated state/timing analysis
- Trace disassembly

Figure 2. Hardware Development Workstation



- *UNIX* System V based: *AT&T* 3B2 Computer or *VAX* 11/780 Computer
- High-level language development: C, Fortran, Pascal
- Download from *UNIX* System host
- C-level symbolic debugger

Figure 3. Software Development – IMP or Ferret Mode



- Full hardware/software feature set
- Highest productivity

Figure 4. Hardware and Software Integrated Workstation – IMP or Ferret Mode

Management Unit, and *WE* 32102 Clock functions. The in-circuit emulator controls the target system and is necessary for diagnosing unstable target hardware. Software testing is supported through the IMP ROM-based assembly-level debugger. IMP controls user hardware and software through mechanisms such as breakpoints, interrupts, reset, and stack manipulation.

Hardware Development Workstation (Figure 2). All operations that can be performed in the stand-alone configuration can be performed in this configuration. In addition, target hardware can be monitored by a logic analyzer. The logic analyzer is interfaced to the target system through the analysis pod. A configuration file for the 64000 System defines the signal names and polarities of the 120-channel logic analyzer access probes. A disassembler displays 64000 System logic traces in *WE* 32100 Microprocessor instruction format.

Software Development Workstation (Figure 3). In this configuration, a *UNIX* System host is interfaced with the analysis pod to allow software, which has been developed on the *UNIX* System, to be downloaded into the analysis pod or target system memory. Software testing is supported by two debuggers, IMP and Ferret. The IMP is used for assembly-level debugging, while Ferret supports C-level debugging. Ferret operates by remote control from the *UNIX* System host, the same host that supports the analysis pod. Ferret supports a variety of advanced features such as action lists, symbolic debugging, and process qualification.

Hardware and Software Integrated Workstation (Figure 4). This configuration is the most powerful set-up of the *WE* 321DS Microprocessor Development System and operates with IMP or Ferret. The user can integrate hardware and software most efficiently by combining both the hardware and software development workstations.

Development System Firmware/Software Feature Summary

Development Firmware Programs

- Assembly-level debugger (IMP)
  - □ Debug in physical and virtual mode
  - □ Display and modify registers/memory
  - □ Set breakpoints for program trace
  - □ Download object files from a *UNIX* System host
  - □ Initiate program execution
  - □ Support for MMU
  - □ Support for MAU.
- System self-test

Development Software Programs (*WE* 321SD Software Development Programs)

- Standard support software
- C-level symbolic debugger (Ferret) (optional)
  - □ Symbolic, high-level input and output operations
  - □ Breakpoints with action lists
  - □ Block-structured command language
  - □ Symbolic stack back trace and memory dump
  - □ Single-step program execution.
- 64000 System support software (optional)
  - □ Symbol table utility
  - □ Download and upload utilities
  - □ Trace disassembler
  - □ Configuration files.

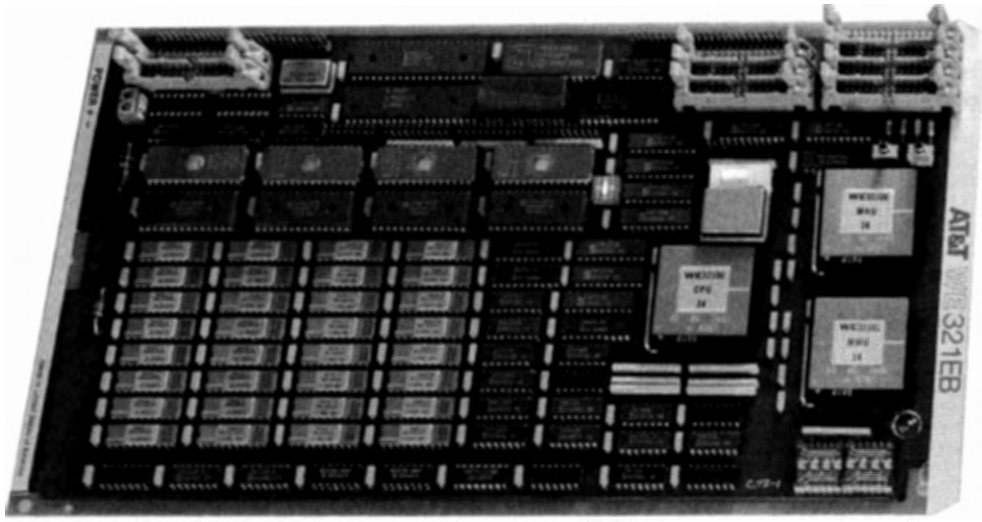
WE® 321EB Microprocessor Evaluation Board

Description

The *WE* 321EB Microprocessor Evaluation Board is a *WE* 32100 Microprocessor based single board microcomputer evaluation system. It allows the evaluation of the hardware and software capabilities and the performance of the *WE* 32100 Microprocessor, the *WE* 32101 Memory Management Unit (MMU), and the *WE* 32106 Math Acceleration Unit (MAU) in an application environment. The evaluation board contains a *WE* 32100 Microprocessor, a *WE* 32101 Memory Management Unit, a *WE* 32106 Math Acceleration Unit, a *WE* 32102 Clock, a ROM-based interactive monitor program (IMP) and system self-test, 96 Kbytes of RAM, and I/O circuitry. The evaluation board requires a single 5 V, 3 A power source for operation and operates at 18 MHz.

Features

- *WE* 32100 Microprocessor
- *WE* 32101 Memory Management Unit
- *WE* 32106 Math Acceleration Unit
- *WE* 32102 Clock (18-MHz)
- 64 Kbytes of high-speed static RAM
- 32 Kbytes of additional RAM
- 64-Kbyte ROM-resident interactive monitor program (IMP) and system self-test program
- Program trace and breakpoint capability
- Seven individually maskable interrupts
- Twenty-four programmable parallel I/O lines
- Two RS-232C serial I/O ports with selectable baud rate (300—9600)
- Communication with terminal and with *UNIX* System host
- Three programmable 16-bit interval timers
- Eight-character LED display (17-segment)
- Push-button reset and abort switches



User Information

Hardware

The *WE* 321EB Microprocessor Evaluation Board includes a *WE* 32100 Microprocessor, a *WE* 32101 Memory Mangement Unit, and a *WE* 32106 Math Acceleration Unit. The microprocessor operates with accesses that are capable of being performed with zero wait states (3-cycle operation for the CPU and 5-cycle operation for the CPU/MMU combination). The evaluation board uses an 18-MHz clock. Other standard components for the *WE* 321EB Microprocessor Evaluation Board include 64 Kbytes of ROM containing IMP and the system self-test, 96 Kbytes of RAM, a system programmable peripheral interface, three programmable 16-bit timers, a programmable interrupt controller with seven available levels, a dual UART (DUART), and an eight-character LED display.

Memory is physically divided into three banks: bank 0 – 64-Kbyte resident monitor program; bank 1 – 32-Kbytes of RAM; and bank 2 – 64-Kbytes of fast static RAM. Banks 1 and 2 are provided for program development and scratch pad space. Bank 0 can be reconfigured to accept 32, 64, or 128 Kbytes of EPROM (2764, 27128, or 27256 devices).

Communication with the evaluation board is provided by two RS-232C serial I/O ports. One of these ports is configured for terminal communications; the other is configured for communications with a host system running a *UNIX* Operating System. This procedure allows programs to be downloaded to the evaluation board for execution and debugging without reconfiguring the cabling. Both ports may be reconfigured for user needs.

The board is supplied with an eight-character alphanumeric display that is used by the interactive monitor program (IMP) and the system self-test to display messages. Each LED consists of 16 segments and a decimal segment. The display is also available for general-purpose use. The 24 programmable parallel I/O lines (8255A Programmable Peripheral Interface, PPI) and the three 16-bit counter/timers (8254-2 Programmable Interval Timer, PIT) are also provided for general-purpose use. In addition, the board is supplied with a 9519A Programmable Interrupt Controller (PIC), which supports seven vectored levels of user-generated interrupts. The interrupt structure is easily expanded either by cascading 9519A PIC devices or by designing and adding a new interrupt circuit on or off board.

Push-button abort and reset switches are located above the MAU. The reset switch is used to reset the entire board, including I/O devices. Depression of the abort switch causes the current process to be interrupted. In both cases, control returns to the monitor program.

Access to board signals is provided at four different areas on the board: the logic analyzer connectors, system expansion connector, I/O access connector, and the jumpers.

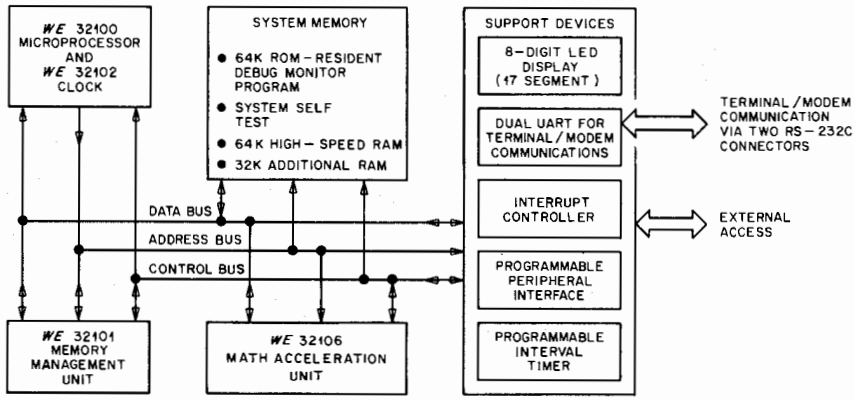


Figure 1. Simplified Block Diagram

Software

The evaluation board is supplied with 64-Kbyte EPROM that contains the interactive monitor program (IMP) and system self-test program. The IMP is an assembly-level, hex-oriented, debugging tool designed to aid in the verification of hardware and the debugging of firmware. The following are some of its key features:

- Download object files from host system
- Initiate program execution
- Trace program
- Display and set software breakpoints
- Select baud rates (300—9600 baud) for host/terminal communications
- Select physical/virtual modes.

The system self-test program checks all of the functions of the board for minimal performance. The self-test program is invoked when power is applied or when the reset button is depressed. The results of the self-test are displayed on the board's LEDs.

The *WE* 321EB Microprocessor Evaluation Board may be ordered with or without the associated host computer *WE* 321SE Evaluation Software Programs. These programs, which provide an I/O library and IMP support software for the evaluation board, are available for an *AT&T* 3B2/300 or 400 Computer or *VAX* 11/780 Computer. Also available are the *WE* 321SG Software Generation Programs. These programs, which are accessed while the IMP is in the transparent mode, allow the user the ability to create, compile, and link programs on a *UNIX* System host before downloading them to the evaluation board.

IMP Command Summary

Register Commands
Display or set CPU registers
Restore user registers

Memory Commands
Display or set memory
Fill memory with specified data
Copy block
Disassemble programs in memory
Execute previous memory command

Data Link Commands
Display or set baud rate (user terminal/data link)
Download host object file
Read host file into memory
Open transparent mode

Execution Control Commands
Display, set, or remove breakpoints
Call user procedure
Initiate program execution
Display or set trace buffer
Provide nonreal-time trace
Switch processes

MMU Control Commands
Flush cache
Set physical or virtual addressing mode

MAU Control Commands
Display or set floating-point numbers in memory
Display or set MAU registers

Addressing Aids
Display or set qualifier
Force physical or virtual mode for current command
Offset from frame or argument pointer

Additional Control Commands
Print addressing mode and breakpoint status
Translate to physical address
Print IMP news and command list
Execute command line as a loop
Delay
Identify, print, or remove process id
Invoke self-test routine

Chapter 4

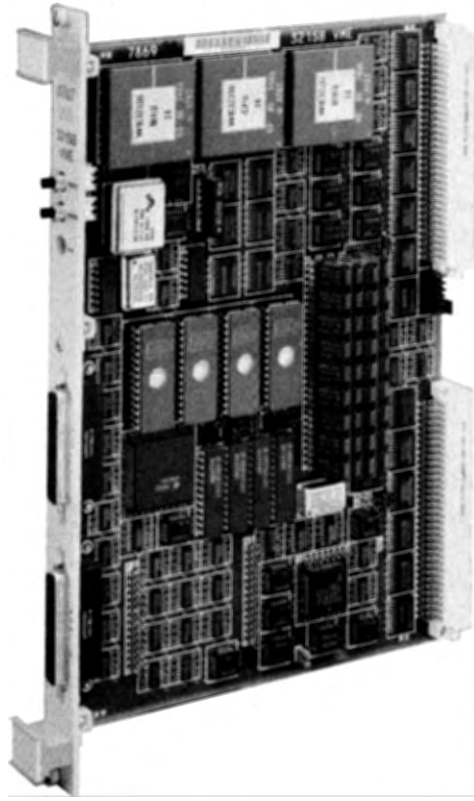
Board-Level Products

WE® 321SB VMEbus Single Board Computer

Description

The *WE* 321SB VMEbus Single Board Computer (SBC) is a high-performance, 32-bit, single board computer that facilitates the rapid configuration of user-defined VMEbus computer systems. Using the SBC with the *UNIX* System V/VME, users can design and assemble open-architecture, demand-paged *UNIX* Systems that are object file and floppy disk format compatible with the *AT&T* 3B Computer family. Thus, many hundreds of applications and language packages available for the *AT&T* 3B Computers will execute directly on a VMEbus computer system built using this SBC and *UNIX* System V/VME. This compatibility gives SBC users immediate access to one of the largest collections of off-the-shelf *UNIX* System software available today.

The VMEbus standard has many supporters and there are over 1000 commercially available bus-compatible products. *UNIX* System V/VME aids users who need to add VMEbus boards to their systems. AT&T has developed *UNIX* System V/VME device drivers for a number of third party VMEbus boards. AT&T has also released driver design guides to help users develop their own device drivers. The SBC is available in both 14- and 18-MHz frequency versions.



Features

- Full support for *UNIX* System V/VME
- *WE* 32100 Microprocessor
- *WE* 32101 Memory Management Unit
- *WE* 32106 Math Acceleration Unit
- 1 Mbyte of DRAM with byte parity
- Up to 256 Kbytes of EPROM or ROM
- On-board debug monitor and self-test firmware
- Two RS-232C serial I/O ports
- Three user-programmable counter/timers
- VMEbus master/slave capabilities
- Manufactured for reliability
- Surface-mount technology
- Optional software/firmware
  - ☐ *UNIX* System V/VME with demand paging
  - ☐ *WE* 321SG C-Software Generation Programs, (native or cross) including an optimized C compiler (source or binary code)
  - ☐ Device Drivers for selected VMEbus peripheral boards (source or binary code)
  - ☐ *WE* 321SB-VFG Firmware Generation Programs
  - ☐ *WE* 321SB-VSD Software Debug Utility Programs-Source Code.

User Information

Hardware

The *WE* 32100 Microprocessor (CPU), *WE* 32101 Memory Management Unit (MMU) and *WE* 32106 Math Acceleration Unit (MAU) are the essential elements of the *WE* 321SB VMEbus Single Board Computer. These devices operate synergistically to assure high system performance.

The second generation CPU provides several times the performance found in many mini-computer systems. It features an on-chip instruction cache storing up to sixty-four 32-bit words to reduce overall memory access time and improve performance. A three-stage pipeline prepares instructions for execution while previous instructions are executing, thereby greatly increasing throughput.

The MMU supports a full 32-bit-wide virtual and physical address space. The result is a linear 32-bit address space, as seen by the programmer. The on-chip logic of the MMU supports automatic miss processing, referenced-bit and modified-bit updating, and fault detection and correction.

The MAU provides both IEEE standard floating-point and decimal high-speed arithmetic capabilities (it benchmarks at over 1 million whetstones). Operating as a coprocessor to the CPU, the MAU works to off-load mathematically-intensive operations from the CPU, thereby enhancing system performance.

The *WE* 321SB VMEbus Single Board Computer (SBC) provides full 32-bit (A32/D32) master and slave interfaces to the VMEbus. The SBC operates with an on-board system clock. On-board memory is divided into one bank of EPROM/ROM byte-wide sockets and one bank of DRAM. The EPROM/ROM bank can be populated with 2764-, 27128-, 27256-, or 27512-type EPROMs for a total of 32, 64, 128, or 256 Kbytes of EPROM, respectively. One Mbyte of DRAM with byte parity is supplied on-board. Byte, halfword (16 bits), word (32 bits) and read/modify/write accesses of memory are supported. Refresh of the DRAM bank is handled automatically on-board.

The SBC contains an AMD Am9513A System Timing Controller (STC) to provide many types of counting, sequencing, and timing capabilities. A variety of programmable operating modes and control features allow the STC to be configured for particular applications, as well as to be dynamically reconfigured under program control. The STC contains five 16-bit counters; one of these is dedicated to on-board circuitry, one is reserved, and the other three are available to the user.

Two serial RS-232C I/O ports provide communication support between the SBC, a terminal, and a host computer system. A Signetics SC2681 Dual-UART (DUART) provides the serial interface that features programmable baud rate, bits per character, stop bits and parity.

The SBC contains control and status registers that are accessed either by writing control to or reading status from appropriate addresses. The status register indicates the type of event that caused an exception. The control register provides software control over hardware functions.

Fifteen levels of interrupts are implemented. Eight of these are onboard; seven are inputs for the VME backplane. In addition, the SBC also supports one nonmaskable interrupt level.

A system status LED, located on the front panel, is visible when the SBC is plugged into a VMEbus card cage. The LED is wired to SBC circuitry that also generates the VMEbus SYSFAIL signal. Reset and abort pushbutton switches are located on the front panel. Pressing the abort button generates a nonmaskable interrupt to the CPU. Pressing the reset switch causes a local reset of the SBC. A reset also occurs on a power up, a VMEbus SYSRESET or CPU reset exception, and can occur under software control.

The *WE* 321SB VMEbus Single Board Computer is manufactured using AT&T advanced surface-mount technology to provide a product with an extraordinary size-to-function ratio. With all of its features, the SBC measures just 160 by 233 mm (6.3 by 9.2 inches), the standard double-height Eurocard size.

Firmware

A debug monitor program is included in on-board EPROM. EPROM contents include power-up board self-test and initialization. Also provided are low-level observation and control capabilities, including program tracing, single-

stepping, register and memory examination and modification, exception and interrupt handling, and breakpointing. Software can be cross-developed and downloaded for stand-alone execution and development. A floppy and hard disk boot facility for *UNIX* System V/VME is also included.

The on-board EPROMs can be replaced by the user, thereby facilitating the use of user-developed firmware. The *WE* 321SB-VFG Firmware Generation Programs contain source and object files that provide the capability to build firmware for the SBC.

Operating Systems

AT&T is offering a highly-tuned release of *UNIX* System V Release 3.1 for the *WE* 321SB VMEbus Single Board Computer. *UNIX* System V, developed by AT&T Bell Laboratories and available since 1982, has received worldwide acceptance as a standardized product. *UNIX* System V/VME includes the most recent enhancements from AT&T, such as demand paging, record/file locking, and simplified system administration.

UNIX System V/VME also includes device drivers for third-party system controller and floppy/hard disk controller boards (and SMD disk-system controller, serial I/O, and network cards for Release 3.1). Additional device drivers are available from AT&T for Release 2.1, including an SMD disk-system controller and intelligent serial I/O. Drivers may also be developed by users, following the guidelines provided in the **UNIX® System V/VME Driver Design Guide** (Release 2.1), the **UNIX® System V/VME Block/Character Interface Driver Development Guide** (Release 3.1), or the **UNIX® System V/VME Driver Design Block/Character Reference Manual** (Release 3.1).

UNIX System V/VME users who want to develop software can use the *WE* 321SG C-Software Generation Programs, which include an AT&T enhanced C compiler and utilities. The *WE* 321SG C-Software Generation Programs are supported either in a stand-alone or cross development environment. Compilers for other languages are available as well.

For users requiring support for real-time operation, JMI Software Consultants, Inc., has ported their popular *C EXECUTIVE** real-time operating system to the *WE* 321SB VMEbus Single Board Computer.

* Registered trademark of JMI Software Consultants, Inc., 904 Sheble Lane, Spring House, PA, (215) 628-0840.

Documentation

Complete and thorough documentation is available from AT&T for the *WE* 321SB VMEbus Single Board Computer and *UNIX* System V/VME. In addition to the numerous documents available from AT&T addressing the *UNIX* Operating System in general, there are several product-specific documents for the SBC and *UNIX* System V/VME. These cover all aspects of configuration, operations, and programming, including such areas such as VMEbus system assembly, using *UNIX* System V/VME, OEM modifications to System V/VME, and device driver design. All documents are described in the handy **UNIX® System V/VME Documentation Roadmap.**

Additional Firmware/Software

The WE® 321SB-VFG Firmware Generation Programs

The *WE* 321SB VMEbus Single Board Computer (SBC) has a firmware monitor that supports both application and kernel development by original equipment manufacturers (OEMs). When bootstrap devices other than the ones supported by the standard SBC firmware are used, a method is needed to change or add to the set of boot devices (i.e., devices that can be used for initial program load) contained in the SBC resident firmware.

The *WE* 321SB-VFG Firmware Generation Programs contain the object files and makefiles that provide the capability to regenerate firmware for the SBC. It provides:

- Firmware object files and makefiles for the *WE* 321SB VMEbus Single Board Computer
- Device configuration tables
- Format conversion program
- Allows for the following applications:
  - □ Development of firmware for SBC
  - □ Additional bootable devices currently not supported by the SBC resident firmware.

In addition to providing the capability to regenerate the firmware, the firmware generation programs provide the capability of building firmware in which nonstandard devices can be used for the initial program load or "boot" of the *UNIX* VMEbus System. The user

must supply a working firmware device driver, in object form, for the intended boot device. The firmware generation programs are then used to link the debug monitor (DEMON), any supported boot driver(s), and the board self-test code with the new boot driver to create an **a.out** image. A supplied utility, **aconv**, is used to convert the **a.out** format to a format suitable for the typical PROM programmer. The converted image must be downloaded to an EPROM programmer to generate a set of EPROMs similar to the standard SBC resident firmware (with the new boot driver). The supported hosts for the firmware generation programs include the *UNIX* VMEbus System and the *AT&T* 3B2/400 Computer.

The WE® 321SB-VSD Software Debug Utility Programs

The *WE* 321SB-VSD Software Debug Utility Programs (Source) contains source code for the assembly level debugger (DEMON), System Self-Test, and an I/O Library. The DEMON support software consists of a utility that allows downloading of files from the host *UNIX* System to the *WE* 321SB VMEbus Single Board Computer. The System Self Test software provides the capabilities to test all major components on the *WE* 321SB VMEbus Single Board Computer. The I/O Library provides I/O functions that software developers may use when writing stand-alone application code. The following summarizes the features provided by the software debug utilities programs:

- Download utility source
- Filled utility source
- SBC diagnostic source
- Diagnostic Control Program utility source
- ROM-resident firmware source
- Makefiles and image conversion utility
- Permits the modification of:
  - □ Monitor program
  - □ System self-test
  - □ Interrupt structure
  - □ Firmware for quick interrupts.

Applications Software

The following is a sample listing of applications software found in the **AT&T® Computer Software Catalog:**

- Advanced Computer Techniques, *ACT BASIC, ACT Cobol 74, ACT Fortran 77, ACT Pascal* compilers
- American Business Systems, Inc., *Business Accounting System*
- Ashton-Tate, *dBase II*™ database management
- Inspiration Systems, *PREVAIL*℠ database system and applications
- Microsoft Corporation, *Microsoft® Word* word processor
- Microsoft Corporation, *Multiplan®* spreadsheet
- Relational Database Systems, Inc., *File-it!*™ database management
- Relational Technology, Inc., *INGRES*™ database management
- RealWorld Corporation, *RealWorld*™ accounting system
- Ryan-McFarland Corporation, *RM/COBOL*™ compiler
- Syntactics Corporation, *CrystalWriter®* word processor.

The trademarks, service marks, or registered trademarks listed above belong to their respective corporation as presented.

Chapter 5

Support Software

WE® 321SG Software Generation Programs Version 1.1

Description

The *WE* 321SG Software Generation Programs (SGP) facilitate using a host *UNIX* System V for creating, debugging, and downloading software to a *WE* 32100 Microprocessor. This support software provides utility programs that enable the user to write applications software in C language or assembly language.

The SGP are available in three binary products:

- The *WE* 321SG-NAT Native *UNIX* System Software Generation Programs
- The *WE* 321SG-CUX Cross *UNIX* System Software Generation Programs
- The *WE* 321SG-CRS Cross Software Generation Programs.

They are also available in a source product packaged on a nine-track magnetic tape.

Features

Utilities

- C compiler
- Assembler
- Link editor
- Disassembler
- Object code optimizer
- Archiver and library maintainer
- Object file host converter
- Object file compressor
- Object and archive file converter
- Object file dumper
- Source code lister
- Object file orderer
- Symbol table printer
- Object file stripper

Libraries

- Math libraries
- Floating-point emulation library
- Common object file access library
- *UNIX* System libraries
  - Standard C library
  - PW library
  - Debug library
  - LEX library
  - Memory allocation library
  - YACC library

User Information

The three products support a variety of user programming environments. The Native *UNIX* System SGP are used when the user's programs are executed on the same computer system and no other C compiler is resident on that computer system. Both Cross SGP's are used when the user's programs are executed on another computer system (target) or another C compiler is already present on the host computer system and the target system runs an operating system other than the *UNIX* System. The Cross Software Generation Program are used when the host system is running the *UNIX* System. All three allow the user the computing power of a host *UNIX* System to aid in the creation and debugging of software for the *WE* 32100 Microprocessor chip set.

Each of the binary products can be configured in either of two floating-point modes (MAU or FPE). The MAU mode provides enhanced floating-point performance through the use of in-line code generation of the *WE* 32106 Math Acceleration Unit (MAU) instruction set (MIS). Alternatively, at the cost of lower performance, the FPE mode supports target systems operating both with and without a math acceleration unit. The FPE mode compiles floating-point operations as calls to library routines that either execute MIS or emulate the operations, depending on whether a math acceleration unit has been determined to be present at process start-up. For the Cross SGP, the FPE mode assumes that the target system does not have an MAU. This support software provides utility programs that enable the user to write applications software in C language or assembly language, and then translate it into machine code.

The SGP products generate output that fully complies with both the *UNIX* System V Interface Definition, SVID, (i.e., when compiled by the SGP products, the *UNIX* System passes the *UNIX* System V verification suite), and the IEEE Standard for Binary Floating-Point Arithmetic (ANSI/IEEE Std. 745-1985) when executing in a *UNIX* System V environment.

Support Software Descriptions

cc Command. This command calls the optimizing C compiler to translate C source files into assembly language files, assemble the resulting files into relocatable object files, and link the files with other object files and/or libraries to form an absolute object file for the processor.

Assembler. This command constructs a relocatable machine code object file from an assembly language source file.

Link Editor. This program collects relocatable object files produced by the assembler and reconstructs them into a single load file for execution or as input for a subsequent link editing.

Disassembler. This command produces a listing of each specified object file. The listing includes the assembly language statements and the binary code associated with those statements.

Object Code Optimizer. This program is a tool that optimizes the text of an unstripped, link-edited load module.

Archiver and Library Maintainer. This command formats one or more files into an archive file. Its main use is to create and update library files as used by the link editor.

Object File Host Converter. This program converts a *WE* 32100 Microprocessor object file with header information formatted for one host machine into an object file with header information for the specified host machine.

Object File Compressor. This command attempts to reduce the size of an object file by removing duplicate structure and union descriptors from the symbol table.

Object File Dumper. This program dumps selected parts of each of the specified object file arguments. It accepts both object files and archives of object files. The options allow selective processing of each file according to the needs of the programmer.

Source Code Lister. This program produces a C source listing, using a C source file(s) with line number information supplied by its associated object file. Line numbers are printed for each breakpointable line generated by the compiler.

Symbol Table Printer. This command prints the symbol table of each specified object file. Files specified may be relocatable or absolute object files, or archives of these files. Each symbol

table entry contains the name of the symbol, its value expressed as an offset or an address (depending on its storage class), its type and derived type, its size in bytes (if available), and the object file section in which this symbol was defined.

Object File Size Printer. This program prints the section name and size information in bytes for each section in the specified object file.

Object File Stripper. This program strips the symbol table, relocation, and line number information from object files, including archives. It is normally run only on production object files (modules) that have been debugged and tested, since no symbolic debugging information is available from a file once it has been stripped. Options control the amount of information stripped from the symbol table.

Math Libraries. The math libraries provide functions (e.g., sine and tangent) with varying levels of performance.

Floating-Point Emulation Library (Cross SGP Only). The floating-point emulation library contains routines that use *WE* 32100 Microprocessor integer instructions to compute floating-point operations.

Common Object File Access Library. The common object file access library contains access routines. These access routines are a collection of functions for reading an object file. Although the calling program must know the detailed structure of the parts of the object file it processes, the routines effectively isolate the calling program from knowledge of the overall structure of the object file. There are 22 routines which can be classified into four categories: open or close object files, read header or symbol table data, seek the data in a particular section, and return the index of a given entry in the symbol table. There is also a version of this library available that allows profiling of programs.

UNIX System Libraries. Two versions of each library are provided: one for the **MAU** mode and one for the **FPE** mode.

- **Standard C Library** (Cross *UNIX* System and Native *UNIX* System SGP Only) – contains the *UNIX* System calls, the standard I/O package, other C library functions, and the floating-point emulation library. In addition to the standard routines, this library also includes routines that provide full compliance with IEEE Standard for Binary Floating-Point Arithmetic (ANSI/IEEE Std. 754-1985).

  The routines provided are single- and double-precision, binary and decimal conversions. Another version of this library allows profiling of programs.

- **PW Library** – contains a set of routines that are used to support programs written for an earlier version of the *UNIX* System.

- **Debug Library** – contains a start-up routine for the symbolic debugger.

- **LEX Library** – contains routines for use with programs generated by the scanner generator, **lex**.

- **Memory Allocation Library** – contains a faster alternate version of the memory allocation routine, **malloc**.

- **YACC Library** – contains routines for use with programs generated by the parser generator, **yacc**.

Supported Host Computers

The Native *UNIX* System SGP are available for the *AT&T* 3B2/400 Computer and the *WE* 321SB VMEbus Single Board Computer.

The Cross *UNIX* System SGP and Cross SGP are available for the *AT&T* 3B2/300/310/400 Computers, *WE* 321SB VMEbus Single Board Computer, and the *VAX* 11/780 Computer.

WE® 321SG C-Software and Cross C-Software Generation Programs Version 2.0

Description

The *WE* 321SG C-Software and Cross C-Software Generation Programs (C-SGP) facilitate using a host *UNIX* System V for creating software for the *WE* 32100 Microprocessor. This support software provides utility programs that enable the user to write applications software in C language or assembly language.

The C-SGP are available in two binary products:

- The *WE* 321SG C-Software Generation Programs (native C-SGP)
- The *WE* 321SG Cross C-Software Generation Programs (Cross C-SGP).

They are also available in a source product packaged on a nine-track magnetic tape.

Features

Utilities

- C compiler
- Assembler
- Link editor
- Disassembler
- Object code optimizer
- Archiver and library maintainer
- Object file host converter
- Object file compressor
- Object and archive file converter
- Object file dumper
- Source code lister
- Object file orderer
- Symbol table printer
- Object file stripper

Libraries

- Math libraries
- Shared libraries
- Stand-alone library
- Common object file access library
- *UNIX* System libraries
  - Standard C library
  - PW library
  - Debug library
  - LEX library
  - Memory allocation library
  - YACC library

User Information

The C-SGP products satisfy two user programming environments:

- **Native Environment.** The *WE* 321SG C-Software Generation Programs (native C-SGP) replace any other native C compiler already installed in an *AT&T* 3B2 Computer System running *UNIX* System V Release 2.0.5/2.1/3.0/3.1/3.1.1 or *UNIX* System V/VME Release 2.1/3.1
- **Cross Environment.** The *WE* 321SG Cross C-Software Generation Programs (Cross C-SGP) are for host systems running *UNIX* System V Release 2.1/3.1. Programs are developed on the host computer system and then downloaded to the target computer system. The target computer system runs one of the following:
  - ☐ *UNIX* System V Release 2.1/3.1
  - ☐ *UNIX* System V/VME Release 2.1/3.1
  - ☐ An operating system other than the *UNIX* Operating System
  - ☐ No operating system.

An additional software package that is available with the native C-SGP, Advanced Programming Utilities Environment (APU), contains commands such as **ctrace**, **lex**, and **yacc**. These and other commands provide the user with a set of programming tools that allow the programmer to do advanced programming and debugging, create shared libraries, and work in an environment where it may be necessary to track and maintain versions of files and programs.

Each of the binary products can generate code in either of two floating-point modes, **MAU** or **FPE**. The **MAU** mode provides enhanced floating-point performance through the use of in-line code generation for the *WE* 32106 Math Acceleration Unit (MAU). Alternatively, at the cost of lower performance, the **FPE** mode supports target systems operating both with and without a math acceleration unit. The **FPE** mode compiles floating-point operations as calls to library routines that either execute the MAU instruction set (MIS) or emulate the operations, depending on whether a math acceleration unit has been determined to be present at process start-up.

Each of the binary products also provides high-performance math libraries, integer and floating-point optimizations, and floating-point support that conforms to the IEEE Standard for Binary Floating-Point Arithmetic (ANSI/IEEE Std 754-1985). The C-SGP complies completely with the *UNIX* System V Interface Definition (SVID), (i.e., when compiled by the SPG products, the *UNIX* System passes the *UNIX* System V Verification Suite).

Tool and Library Descriptions

cc Command. This command calls the optimizing C compiler to translate C source files into assembly language, assemble the resulting assembly files into relocatable object files, and link these files with other object files and/or libraries to form an absolute object file for the processor.

Assembler. This command constructs a relocatable machine code object file from an assembly language source file.

Link Editor. This program collects relocatable object files produced by the assembler and reconstructs them into a single load file for execution or as input for a subsequent link editing.

Disassembler. This command produces a listing of each specified object file. The listing includes the assembly language statements and the binary code associated with those statements.

Object Code Optimizer. This program is a tool that optimizes the text of an unstripped, link-edited load module.

Archiver and Library Maintainer. This command formats one or more files into an archive file. Its main use is to create and update library files as used by the link editor.

Object File Host Converter. This program converts a *WE* 32100 Microprocessor object file with header information formatted for one host machine into an object file with header information for the specified host machine.

Object File Compressor. This command attempts to reduce the size of an object file by removing duplicate structure and union descriptors from the symbol table.

Object File Dumper. This program dumps selected parts of each of the specified object file arguments. It accepts both object files and archives of object files. The options allow selective processing of each file according to the needs of the programmer.

Source Code Lister. This program produces a C source listing, using a C source file(s) with line number information supplied by its associated object file. Line numbers are printed for each breakpointable line generated by the compiler.

Symbol Table Printer. This command prints the symbol table of each specified object file. Files specified may be relocatable or absolute object files, or archives of these files. Each symbol table entry contains the name of the symbol, its value expressed as an offset or an address (depending on its storage class), its type and derived type, its size in bytes (if available), and the object file section in which this symbol was defined.

Object File Size Printer. This program prints the section name and size information in bytes for each section in the specified object file.

Object File Stripper. This program strips the symbol table, relocation, and line number information from object files, including archives. It is normally run only on production object files (modules) that have been debugged and tested, since no symbolic debugging information is available from a file once it has been stripped. Options control the amount of information stripped from the symbol table.

Math Libraries. The math libraries provide functions (e.g., sine and tangent) with varying levels of performance.

Shared Libraries. These libraries help save memory and disk storage space and also make executable files using library code easier to maintain. The shared libraries are only supported for target systems running *UNIX* System V Release 3.0 or later.

Stand-Alone Library (Cross C-SGP only). The stand-alone library supports cross compilation of programs to *WE* 32100 Microprocessor-based hardware systems that are running on an operating system other than the *UNIX* Operating System.

Common Object File Access Library. The common object file access library contains access routines. These access routines are a collection of functions for reading an object file. Although the calling program must know the detailed structure of the parts of the object file it processes, the routines effectively isolate the calling program from knowledge of the overall structure of the object file. There are 22 routines which can be classified into four categories: open or close object files, read header or symbol table data, seek the data in a particular section, and return the index of a given entry in the symbol table. There is also a version of this library available that allows profiling of programs.

UNIX System Libraries. Two versions of each library are provided: one for the **MAU** mode and one for the **FPE** mode.

- **Standard C Library** – contains the *UNIX* System calls, the standard I/O package, other C library functions, and the floating-point emulation library. In addition to the standard routines, this library also includes routines that provide full compliance with IEEE Standard for Binary Floating-Point Arithmetic (ANSI/IEEE Std. 754-1985).

  The routines provided are single- and double-precision, binary and decimal conversions. Another version of this library allows profiling of programs.

- **PW Library** – contains a set of routines that are used to support programs written for an earlier version of the *UNIX* System.

- **Debug Library** - contains a start-up routine for the symbolic debugger.

- **LEX Library** - contains routines for use with programs generated by the scanner generator, **lex**.

- **Memory Allocation Library** - contains a faster alternate version of the memory allocation routine, **malloc**.

- **YACC Library** - contains routines for use with programs generated by the parser generator, **yacc**.

Supported Host Computers

The native C-Software Generation Programs are available for the AT&T 3B2/310/400/600 Computers, and the *WE* 321SB VMEbus Single Board Computer.

The Cross Software Generation Programs are available for the AT&T 3B2/300/400 Computers and the *WE* 321SB VMEbus Single Board Computer.

UNIX® System V Release 2/3

Description

UNIX System V Release 2/3 source code provides the operating system and user interface for the 32-bit *WE* 32100/32200 Microprocessor chip set. Optionally, this software may be compiled by the user with an optimized version of the *WE* 321SG C-Software and Cross C-Software Generation Programs (C-SGP) for the *WE* 32100 Microprocessor chip set. The operating system has been tested and certified on *AT&T* 3B2 Computers with the optimized C-SGP.

The optimized C-SGP provides several levels of optimization and different types of floating-point support as run-time options.

Features

- Multi-user/multi-tasking
- Enhanced C software generation system with symbolic debugging includes shared library support
- Demand-paged virtual memory gives at least 16 Mbytes of linear address space per process; includes U page
- File and record locking system calls provide user data protection
- Self-configuration allows installation of new drivers without regenerating the system
- Simplified system administration eliminates the need for a full-time, on-site systems expert
- Job control language
- Cross-compilers allow program development in a large computer environment
- Exploits *UNIX* System oriented features of the *WE* 32100 Microprocessor chip set, such as full 32-bit architecture, efficient process switching, memory management, and optional hardware floating-point capability
- Facilitates personalization to user-developed hardware configurations
- Easy to use documentation
- Remote file sharing using Ethernet local area network
- Improved signal mechanisms
- Incremental back-up of nested files systems

Additional Features for UNIX® System V Release 3

- Internationalization, includes support for 8-bit code set, alternate date/time formats, and character class/conversion rules
- Assists menu/forms interface
- Faster Curses/Terminfo support the writing of terminal-independent applications
- Remote file sharing (RFS)
- STREAMS Mechanism and Tools
- AT&T Transport Interface
- Media-independent **uucp**
- Executable shared libraries
- ASSIST Interface
- Performance improvements, including paging the user area, remote file sharing client caching, and smaller and faster curses
- Complies completely with the *UNIX* System V Interface Definition Issue 2 (SVID), (i.e., passes the *UNIX* System V Verification Suite Release 3)

UNIX System V/VME Release 2.1/3.1

Description

UNIX System V/VME Release 2.1/3.1 is a multi-user/multi-testing operating system derived from the *UNIX* System running on *AT&T* 3B Computers that provides operating system support for the *WE* 321SB VMEbus Single Board Computer and the industry standard VMEbus modular design simplifies the manipulation of files and commands. The hierarchical file structure allows for easy adding, deleting, and moving of files within a structure set up by the user. Input to commands can come from a terminal or a file, and output can be directed to files or to peripheral devices. The output of a command can be directed to the input of another command, allowing users to easily create specialized functions.

The *UNIX* System shell is also a programming language that can be used to create user-definable applications and functions.

This product is available in both source and binary (executable) code.

Features

- File and record locking system calls provide user data protection
- Simplified system administration eliminates the need for a full-time, on-site systems expert
- Job control language
- Internationalization, includes support for 8-bit code set, alternate Date/Time formats, and character class/conversion rules
- Exploits *UNIX* System oriented features of *WE* 32100 Microprocessor chip set, such as full 32-bit architecture, efficient process switching, memory management, and optional hardware floating-point capability
- Facilitates personalization to user-developed hardware configurations
- Easy to use documentation
- Remote file sharing using Ethernet local area network
- Incremental back-up of nested file systems
- Device drivers for system controller and floppy/hard disk controller boards
- Improved packaging for system builders

Additional Features for UNIX® System V/VME Release 3.1

- Faster *Curses/Terminfo* support the writing of terminal-independent applications
- Remote file sharing (RFS)
- STREAMS Mechanism and Tools
- AT&T Transport Interface
- Media-independent **uucp**
- Executable shared libraries
- ASSIST Interface
- Performance improvements, including paging the user area, remote file sharing client caching, and smaller and faster curses
- Complies completely with the *UNIX* System V Interface Definition Issue 2 (SVID), (i.e., passes the *UNIX* System V Verification Suite Release 3)

User Information

Using the SBC with the *UNIX* System V/VME, users can design and assemble open-architecture, demand-paged *UNIX* Systems that are object file and floppy disk format compatible with the *AT&T* 3B Computer family. Thus, many hundereds of applications and language packages available for the *AT&T* 3B Computers will execute directly on a VMEbus computer system build using the SBC and *UNIX* System V/VME. This compatability gives SBC users immediate access to one of the largest collections of off-the-shelf *UNIX* System software available today.

The VMEbus standard has many supporters and there are over 1000 commercially available bus-compatible products. *UNIX* System V/VME aids users who need to add VMEbus boards to their systems. AT&T has developed *UNIX* System V/VME device drivers for a number of third party VMEbus boards. AT&T has also released driver design guides to help users develop their own device drivers.

UNIX System V, developed by AT&T Bell Laboratories and available since 1982, has received worldwide acceptance as a standardized product. *UNIX* System V/VME includes the most recent enhancements from AT&T, such as demand paging, record/file locking, and simplified system administration.

The *UNIX* Operating System oversees the execution of many user programs and commands. Although these programs and commands seem to execute simultaneously, each application is scheduled to use the processor for a short period of time to the exclusion of all other programs. Thus, in addition to providing a multi-user system the *UNIX* System provides each user with the capability of running several programs at once. This feature is called multi-programming.

UNIX System V/VME also includes device drivers for third-party system controller and floppy/hard disk controller boards (and SMD disk-system controller, serial I/O, and network cards for Release 3.0). Additional device drivers are available from AT&T for Release 2.0, including a hard disk system controller and intelligent serial I/O. Drivers may also be developed by users, following the guidelines provided in the **UNIX® System V/VME Block/Character Interface Driver Development Guide** or the **UNIX® System V/VME Driver Design Block/Character Reference Manual.**

UNIX System V/VME users who want to develop software can use the *WE* 321SG C-Software Generation Programs, which include an AT&T enhanced C compiler and utilities. The *WE* 321SG C-Software and Cross C-Software Generation Programs are supported either in a stand-alone or cross development environment. Compilers for other languages are available as well.

Many *UNIX* System based products are available from AT&T and other vendors. Descriptions of these products may be found in the **AT&T Computer Software Catalog.** Products offered through the AT&T *UNIX* System Toolchest are available to customers who have a *UNIX* System source license.

Chapter 6

Training

Training

To meet the learning needs of the *WE*® 32-Bit Microprocessor and Peripheral users, AT&T offers courses in the use and operation of these and related products.

Educational Programs

AT&T offers courses related to the use and operation of 32-bit microprocessors at:

- Corporate Education Center in Hopewell, New Jersey

For information on available courses, pricing, locations, and lodging contact the Registrar/Computer Systems Training at **1-800-247-1212**. AT&T employees may call **(609) 639-4357** or **CORNET 257-4357**.

The following courses discuss the *WE* 32-Bit Microprocessor architectures, support devices, and hardware and software development tools and systems for designers and users of 32-bit microcomputer systems:

- **UC2201 – WE® 32100 Microsystem Overview** (2 days)

  Provides an overview of the AT&T *WE* 32-Bit Microprocessors and peripheral devices, with emphasis on the *WE* 32100 and 32200 Microprocessors, *WE* 32101 Memory Management Unit, *WE* 321EB Microprocessor Evaluation Board, *WE* 321SG Software Generation Programs, and support devices.

- **UC2202 – WE® 32100 Microprocessor** (5 days)

  Provides an in-depth study of the architecture and operation of the 32-bit *WE* 32100 and 32200 Microprocessors. Topics include architecture, programming, system calls, process switching, interrupt handling, fault and exception handling, timing, software and hardware debugging, and peripherals. The *WE* 32101 and 32201 Memory Management Units are also covered in this course. Hands-on experience is acquired through laboratory exercises that reinforce classroom lecture material.

- **UC2203 – WE® 321DS Microprocessor Development System** (3½ days)

  Provides an understanding of the architecture, operation, and software development tools of the *WE* 321DS Microprocessor Development System. The development system is used to develop, test, and debug *WE* 32100 Microprocessor and *WE* 321012 Memory Management Unit based applications. The student will study the architecture and operation of the *WE* 321AP Microprocessor Analysis Pod, including Ferret, the C Language Symbolic Debugger. The student will also be able to debug software downloaded to a target system using symbolic debugging and logic analysis.

- **UC2204 – WE® 321SB VMEbus Single Board Computer and Software Development Tools** (3 days)

  This fast-paced course provides a thorough coverage of the *WE* 321SB VMEbus Single Board Computer and software development tools. VME system designers and integrators will acquire the information needed to:

  - ☐ Write stand-alone programs using the *UNIX* Operating System on the VMEbus.
  - ☐ Use the firmware to boot programs from disk, execute, and return to firmware upon program completion.
  - ☐ Download programs to a PROM burner, burn EPROMs, and use the firmware to execute the programs the EPROMs.
  - ☐ Recognize the various system configurations (and constraints) and real time operating system available.

- **UC2230 – WE® 32100 Microprocessor Basics** (4½ days)

  Provides an understanding of the *WE* 32100 and 32200 Microprocessor 32-bit architectures. Principle CPU elements such as registers, buses, and controllers are covered. Data types, data handling, addressing capabilities, and I/O devices are covered. C Language constructs are presented with their equivalent assembly language. Labs provide experience with hardware and software using a microcomputer system based on the *WE* 32100 Microprocessor.

- **UC2231 – WE® 32100 Microprocessor Architecture and Software** (7½ days)

  This course covers, at a slower pace, the same material as course UC2202. The course provides an in-depth study of the architectures and operation of the *WE* 32100 and 32200 Microprocessors. Topics include architecture, programming, system calls, process switching, interrupt handling, fault and exception handling, timing, software and hardware debugging, and peripherals. The *WE* 32101 and 32201 Memory Management Units are an integral part of the course. Hands-on experience is acquired through laboratory exercises that reinforce classroom lecture materials.

- **UC6010 – UNIX® System V/VME Device Drivers** (3 days)

  Provides an in-depth study of techniques and strategies used to interface a VMEbus device to the *UNIX* Operating System through a *UNIX* Operating System Device Driver. The course also gives the VME system designer an understanding of:

  - ☐ Device driver operating environments
  - ☐ Installation and testing of firmware

The following courses are prerequisites for some of the above courses. These courses are offered at the locations mentioned above, and at most AT&T training facilities located throughout the country. Contact the Registrar for more information concerning locations.

- **UC1070* – Fundamentals of the UNIX® Operating System for Users** (4½ days)

  Introduces how to use the *UNIX* System. Topics include the files system hierarchy, commonly used *UNIX* System commands, the *UNIX* System editor vi, shell and editor metacharacters, protecting files, some shell programming, and communicating with and transferring files to other *UNIX* Systems.

- **UC1007 – Fundamentals of the UNIX® Operating System for Programmers** (2½ days)

  Provides an understanding and working knowledge of the *UNIX* System with emphasis on how to logon to a *UNIX* System, the basic use of general-purpose commands, the file system, and the editor.

- **UC1001 – C Language Programming** (5 days)

  Provides a basic working knowledge of C language programming. Topics include identifiers, the full complement of C operators, program flow control constructs, functions, storage classes, pointers to various data types, arrays, structures, unions, and standard input and output. Also discussed are the *UNIX* System interface and *UNIX* System routines.

- **UC1633 – Introduction to C Language for Nonexperienced Programmers** (10 days)

  Contains the same topics that are covered in course UC1001. This course is presented over a longer time frame to allow the nonexperienced student additional time for classroom interaction and laboratory experimentation.

*Course numbers beginning with UC are for commercial customers. Courses for AT&T employees start with CS instead of UC. In all other regards, the courses are identical.

Chapter 7

Ordering Information

Ordering Information

All AT&T *WE* 32-Bit Microprocessors and related products can be ordered by contacting your AT&T Account Manager, or by calling **1-800-372-2447**. The following tables list the products available.

Microprocessors and Peripherals	
Device	**Available Frequencies (MHz)**
WE 32100 Microprocessor	10, 14, 18
WE 32200 Microprocessor	24
WE 32101 Memory Management Unit	10, 14, 18
WE 32201 Memory Management Unit	24
WE 32102 Clock	10, 14, 18, 24
WE 32103 DRAM Controller	10, 14, 18
WE 32104 DMA Controller	10, 14, 18
WE 32106 Math Acceleration Unit	10, 14, 18
WE 32206 Math Acceleration Unit	24

Microprocessors and Peripherals – Extended Temperature	
Device	**Available Frequencies (MHz)**
WE 32100 Microprocessor	10, 14
WE 32101 Memory Management Unit	10, 14
WE 32102 Clock	10, 14
WE 32103 DRAM Controller	10, 14
WE 32104 DMA Controller	10, 14
WE 32106 Math Acceleration Unit	10, 14

Software, Board Level, and Support Tool Products
WE 321SG-NAT Native Software Generation Programs Version 1.1
WE 321SG-CUX Cross *UNIX* System Software Generation Programs Version 1.1
WE 321SG-CRS Cross Software Generation Programs Version 1.1
WE 321SG C-Software Generation Programs Version 2.0
WE 321SG Cross C-Software Generation Programs Version 2.0
UNIX System V Release 2
UNIX System V Release 3
WE 321SB VMEbus Single Board Computer (14 and 18 MHz)
UNIX System V/VME Release 2.1
UNIX System V/VME Release 3.1
WE 321SB-VFG Firmware Generation Programs
WE 321SB-VSD Software Debug Utility Programs
WE 321DM CPU & MMU Device Monitor
WE 322DM CPU & MMU Device Monitor
WE 321EB Microprocessor Evaluation Board
WE 321DS Microprocessor Development System

Ordering Information

Documentation

The following manuals are available:

Title	Select Code
WE 32100 Microprocessor Information Manual	307-730
WE 32101 Memory Management Unit Information Manual	307-731
WE 32103 DRAM Controller Information Manual	307-732
WE 32104 DMA Controller Information Manual	307-733
WE 32106 Math Acceleration Unit Information Manual	307-734
WE 32200 Microprocessor Information Manual	307-705
WE 32201 Memory Management Unit Information Manual	307-706
WE 32206 Math Acceleration Unit Information Manual	307-707
WE 321DM CPU & MMU Device Monitor User Manual	307-763
WE 321DS Microprocessor Development System User Manual	451-012
WE 321EB Microprocessor Evaluation Board User Manual	451-019
WE 321SB VMEbus Single Board Computer User Manual	307-794
WE 321SG Software Generation Programs User Guide (Version 1.1)	307-760
WE 321SG C-Software and Cross C-Software Generation Programs User Guide (Version 2.0)	307-701

The following manuals and guides are available for the AT&T *UNIX* System V/VME operating system. Consult the *UNIX* System V/VME Documentation Roadmap (Select Code 307-782 for Release 2.1, and Select Code 307-061 for Release 3.1) to determine your documentation needs.

Title	Release 2.1 Select Code	Release 3.1 Select Code
AT&T Computer Systems Documentation Catalog	300-000	—
UNIX System V Interface Definition Issue 2 Volume I	307-011	307-011
UNIX System V Interface Definition Issue 2 Volume II	307-012	307-012
UNIX System V Interface Definition Issue 2 Volume III	—	307-013
UNIX System V User Guide	307-118	307-118
UNIX System V/VME Application Design Guide	307-774	—
UNIX System V/VME Crash Analysis Guide	307-772	—
UNIX System V/VME DEMON Guide	307-773	307-773
UNIX System V/VME Directory and File Management Utilities Guide	307-776	—
UNIX System V/VME Documentation Roadmap	307-782	307-061
UNIX System V/VME Driver Design Guide	307-771	—
UNIX System V/VME Editing Guide	307-126	—
UNIX System V/VME Error Message Manual	307-770	307-049
UNIX System V/VME Inter-Process Communication Utilities Guide	307-775	—
UNIX System V/VME OEM Porting Manual	307-780	—
UNIX System V/VME Product Overview	307-781	
UNIX System V/VME Programming Guide	307-791	307-791
UNIX System V/VME Programmer Reference Manual	307-788	307-053
UNIX System V/VME Release Notes	307-783	307-059
UNIX System V/VME SCCS Utilities Guide	307-778	—
UNIX System V/VME Security Administration Utilities Guide	307-779	
UNIX System V/VME Support Tools Guide	307-785	—
UNIX System V/VME System Administration Guide	307-784	307-047

UNIX System V/VME System Administration Reference Manual	307-789	307-056
UNIX System V/VME Terminal Information Utilities Guide	307-777	—
UNIX System V/VME Tuning and Configuration Guide	307-786	307-043
UNIX System V/VME User Reference Manual	307-790	307-057
UNIX System V/VME OEM Systems Manual	307-792	—
UNIX System V/VME Block/Character Interface Driver Development Guide	—	307-191
UNIX System V/VME Driver Design Block/Character Reference Manual	—	307-192
UNIX System V/VME System Builders Reference Guide	—	307-068

In addition, refer to the **AT&T Computer Software Catalog** (Select Code 311-024) for specific *UNIX* System V software products. The catalog contains over 1500 software applications written for the *UNIX* Operating System.

Contact your AT&T Account Manager for single copies of the above manuals. For more than one copy, contact the AT&T Customer Information Center on **1-800-432-6600.**

Additional documentation in the form of product descriptions, data sheets, and application notes are available through your AT&T Account Manager, or call **1-800-372-2447.**

Notes

Notes

Notes

For additional information contact
your AT&T Account Manager
or call:

☐ AT&T
Dept. 51AL603140
555 Union Boulevard
Allentown, PA 18103
1-800-372-2447

☐ AT&T Microelectronics
Freischützstrasse 92
8000 Munich 81
West Germany
Tel. 0 89/95 97 0 Telex 5 216 884

☐ AT&T Microelectronics Pte. Ltd.
745 Larong 5 Toa Payoh
Singapore 1231
Tel. 250 8422 / 253 3722
Telex RS 21473 /RS 55038

☐ AT&T International Japan Ltd.
Nippon Press Center Bldg.
2-1, Uchisaiwai-cho, 2-Chome,
Chiyoda-ku, Tokyo 100 Japan
Tel. (03) 593 3301
Telex J32562 ATTIJ

AT&T reserves the right to make changes to the product(s), including any hardware, software, and/or firmware contained therein, described herein without notice. No liability is assumed as a result of the use or application of this product(s). No rights under any patent accompany the sale of any such product(s).

© 1987 AT&T
All Rights Reserved
Printed in USA

August 1987

CA87-20MCP

AT&T
The right choice.