

Am29C327

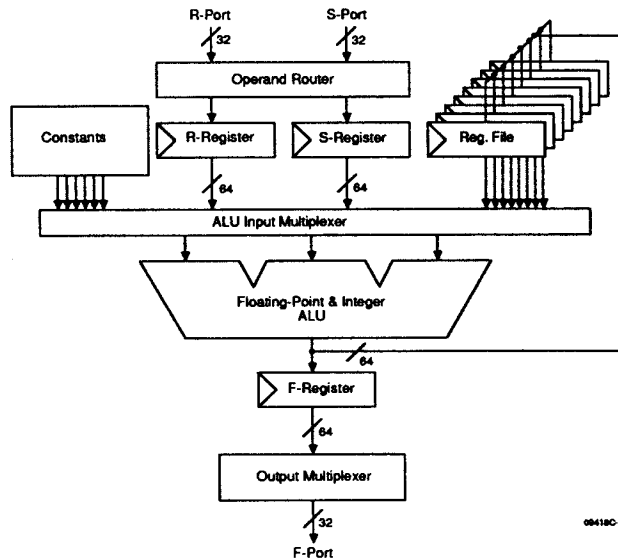
Double-Precision Floating-Point Processor

FINAL

DISTINCTIVE CHARACTERISTICS

- High-performance double-precision floating-point processor
- Comprehensive floating-point and integer instruction sets
- Single VLSI device performs single-, double-, and mixed-precision operations
- Performs conversions between precisions and between data formats
 - ANSI/IEEE standard 754-1985; single- and double-precision
 - DEC F, DEC D, DEC G
 - IBM single- and double-precision
- Exact IEEE compliance for denormalized numbers with no speed penalty
- Eight-deep register file for intermediate results and on-chip 64-bit data path facilitates compound operations; e.g., Newton-Raphson division, sum-of-products, and transcendental
- Supports pipelined or flow-through operation
- Fabricated with Advanced Micro Devices' 1.2 micron CMOS process

SIMPLIFIED BLOCK DIAGRAM



Publication # Rev. Amendment
09418 C /0
Issue Date: December 1988

Am29C327

2-59

GENERAL DESCRIPTION

The Am29C327 double-precision floating-point processor is a single VLSI device that implements an extensive floating-point and integer instruction set. The three most widely used floating-point standards – IEEE, DEC, and IBM – are supported for both single- and double-precision operations. IEEE operations comply with the ANSI/IEEE Standard 754, with direct implementation of special features such as gradual underflow and handling of traps and denormalized numbers.

The Am29C327 consists of a 64-bit ALU, a 64-bit datapath, and a control unit. The ALU has three data input ports, and can perform single-operand, two-operand, and three-operand operations. The data path comprises two 64-bit input operand registers, an 8-by-64-bit register file for storage of intermediate results, three operand-selection multiplexers that provide for orthogonal selection of input operands, a 64-bit output regis-

ter, and an output multiplexer that allows access to the 32 MSBs or 32 LSBs of the result data. Control signals determine the operation to be performed, the source of operands, the operand precisions, the rounding mode, and other aspects of device operation.

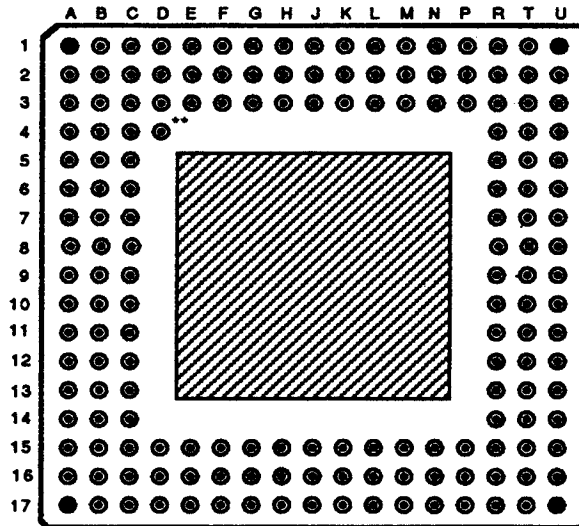
Operations can be performed in either of two modes: flow-through or pipelined. In the flow-through mode, the ALU is completely combinatorial; this mode is best suited for scalar operations. Pipelined mode divides the ALU into one or two pipeline stages, for use in vector operations, as often found in graphics or signal processing.

Fabricated with AMD's 1.2 micron CMOS technology, the Am29C327 is housed in a 169-lead pin-grid-array (PGA) package.

RELATED AMD PRODUCTS

Part No.	Description
Am29C10A	CMOS 12-Bit Sequencer
Am29C111	CMOS 16-Bit Sequencer
Am29C116	CMOS 16-Bit Microprocessor
Am29CPL141	CMOS 64 x 16 EPROM Programmable Controller
Am29CPL142	CMOS 128 x 16 EPROM Programmable Controller
Am29CPL144	CMOS 512 x 16 EPROM Programmable Controller
Am29CPL151	CMOS 64 x 16 EPROM Programmable Controller-Slim DIP
Am29CPL152	CMOS 128 x 16 EPROM Programmable Controller-Slim DIP
Am29CPL154	CMOS 512 x 16 EPROM Programmable Controller-Slim DIP
Am29C323	CMOS 32-Bit Parallel Multiplier
Am29C325	CMOS 32-Bit Floating-Point Processor
Am29C331	CMOS 16-Bit Sequencer
Am29C332	CMOS 32-Bit Arithmetic Logic Unit
Am29C334	CMOS Four-Port, Dual-Access Register File

CONNECTION DIAGRAM
169-Lead PGA*
Bottom View



09418C-5A

CD011601

*Pinout observed from pin side of package.

**Alignment pin (not connected internally).

PGA PIN DESIGNATIONS
(Sorted by Pin No.)

PIN NO.	PIN NAME	PIN NO.	PIN NAME	PIN NO.	PIN NAME	PIN NO.	PIN NAME
A-1	S ₃₁	C-9	V _{CCO}	J-15	TSEL ₁	R-10	V _{CC}
A-2	F ₄	C-10	F ₂₀	J-16	TSEL ₀	R-11	CLOCK
A-3	F ₆	C-11	V _{CCO}	J-17	TSEL ₂	R-12	ENF
A-4	F ₈	C-12	GNDO	K-1	S ₉	R-13	ENI
A-5	F ₁₀	C-13	F ₂₉	K-2	S ₁₀	R-14	FSEL
A-6	F ₁₂	C-14	GNDO	K-3	GND	R-15	RFSEL ₁
A-7	F ₁₄	C-15	V _{CCO}	K-15	QSEL ₁	R-16	PSEL ₃
A-8	F ₁₆	C-16	I ₂	K-16	QSEL ₀	R-17	PSEL ₀
A-9	F ₁₈	C-17	I ₆	K-17	TSEL ₃	T-1	R ₂₈
A-10	F ₂₁	D-1	S ₂₄	L-1	S ₈	T-2	R ₂₃
A-11	F ₂₂	D-2	S ₂₅	L-2	S ₇	T-3	R ₂₁
A-12	F ₂₄	D-3	S ₂₉	L-3	S ₆	T-4	R ₁₈
A-13	F ₂₇	D-15	I ₀	L-15	GNDO	T-5	R ₁₆
A-14	F ₂₈	D-16	I ₃	L-16	QSEL ₃	T-6	R ₁₃
A-15	F ₃₁	D-17	I ₈	L-17	QSEL ₂	T-7	R ₁₀
A-16	SLAVE	E-1	S ₂₁	M-1	S ₅	T-8	R ₇
A-17	I ₁	E-2	S ₂₃	M-2	S ₄	T-9	R ₅
B-1	S ₃₀	E-3	S ₂₆	M-3	S ₂	T-10	R ₃
B-2	F ₁	E-15	I ₄	M-15	V _{CCO}	T-11	R ₀
B-3	F ₃	E-16	I ₇	M-16	FLAG ₂	T-12	RM ₁
B-4	F ₅	E-17	I ₉	M-17	FLAG ₁	T-13	ENS
B-5	F ₇	F-1	S ₁₈	N-1	S ₃	T-14	ÖES
B-6	F ₉	F-2	S ₂₀	N-2	S ₁	T-15	ENRF
B-7	F ₁₃	F-3	S ₂₂	N-3	R ₃₀	T-16	RFSEL ₀
B-8	F ₁₅	F-15	V _{CC}	N-15	FLAG ₆	T-17	PSEL ₁
B-9	F ₁₇	F-16	I ₁₀	N-16	FLAG ₄	U-1	R ₂₅
B-10	F ₁₉	F-17	I ₁₂	N-17	FLAG ₃	U-2	R ₂₂
B-11	F ₂₃	G-1	S ₁₅	P-1	S ₀	U-3	R ₁₉
B-12	F ₂₅	G-2	S ₁₇	P-2	R ₂₉	U-4	R ₁₇
B-13	F ₂₆	G-3	S ₁₉	P-3	R ₂₆	U-5	R ₁₅
B-14	F ₃₀	G-15	GND	P-15	PSEL ₂	U-6	R ₁₄
B-15	V _{CC}	G-16	I ₁₁	P-16	SIGN	U-7	R ₁₁
B-16	MSERR	G-17	S/ $\overline{D}$ F	P-17	FLAG ₅	U-8	R ₉
B-17	I ₅	H-1	S ₁₃	R-1	R ₃₁	U-9	R ₆
C-1	S ₂₇	H-2	S ₁₄	R-2	R ₂₇	U-10	R ₄
C-2	S ₂₈	H-3	S ₁₆	R-3	R ₂₄	U-11	R ₂
C-3	F ₀	H-15	S/ $\overline{D}$ S	R-4	R ₂₀	U-12	R ₁
C-4	F ₂	H-16	I ₁₃	R-5	V _{CC}	U-13	RM ₀
C-5	V _{CCO}	H-17	S/ $\overline{D}$ R	R-6	GND	U-14	RM ₂
C-6	GNDO	J-1	S ₁₁	R-7	R ₁₂	U-15	ENR
C-7	F ₁₁	J-2	S ₁₂	R-8	R ₈	U-16	ÖEF
C-8	GNDO	J-3	V _{CC}	R-9	GND	U-17	RFSEL ₂

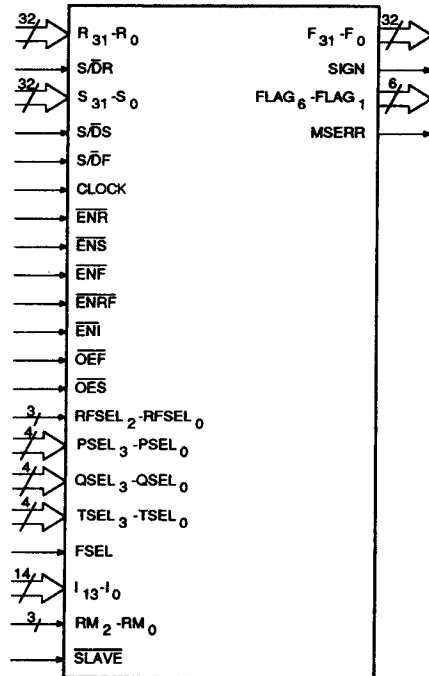
Note: Pin number D-4 = Alignment Pin
V_{CCO} and GNDO are power and ground pins for the output buffers.
V_{CC} and GND are power and ground pins for the rest of the logic.

PGA PIN DESIGNATIONS (Cont'd.)
(Sorted by Pin Name)

PIN NO.	PIN NAME	PIN NO.	PIN NAME	PIN NO.	PIN NAME	PIN NO.	PIN NAME
R-11	CLOCK	P-17	FLAG ₅	T-9	R ₅	K-1	S ₉
R-12	ENF	N-15	FLAG ₆	U-9	R ₆	K-2	S ₁₀
R-13	ENI	R-14	FSEL	T-8	R ₇	J-1	S ₁₁
U-15	ENR	G-15	GND	R-8	R ₈	J-2	S ₁₂
T-15	ENRF	K-3	GND	U-8	R ₉	H-1	S ₁₃
T-13	ENS	R-6	GND	T-7	R ₁₀	H-2	S ₁₄
C-3	F ₀	R-9	GND	U-7	R ₁₁	G-1	S ₁₅
B-2	F ₁	C-6	GNDO	R-7	R ₁₂	H-3	S ₁₆
C-4	F ₂	C-8	GNDO	T-6	R ₁₃	G-2	S ₁₇
B-3	F ₃	C-12	GNDO	U-6	R ₁₄	F-1	S ₁₈
A-2	F ₄	C-14	GNDO	U-5	R ₁₅	G-3	S ₁₉
B-4	F ₅	L-15	GNDO	T-5	R ₁₆	F-2	S ₂₀
A-3	F ₆	D-15	I ₀	U-4	R ₁₇	E-1	S ₂₁
B-5	F ₇	A-17	I ₁	T-4	R ₁₈	F-3	S ₂₂
A-4	F ₈	C-16	I ₂	U-3	R ₁₉	E-2	S ₂₃
B-6	F ₉	D-16	I ₃	R-4	R ₂₀	D-1	S ₂₄
A-5	F ₁₀	E-15	I ₄	T-3	R ₂₁	D-2	S ₂₅
C-7	F ₁₁	B-17	I ₅	U-2	R ₂₂	E-3	S ₂₆
A-6	F ₁₂	C-17	I ₆	T-2	R ₂₃	C-1	S ₂₇
B-7	F ₁₃	E-16	I ₇	R-3	R ₂₄	C-2	S ₂₈
A-7	F ₁₄	D-17	I ₈	U-1	R ₂₅	D-3	S ₂₉
B-8	F ₁₅	E-17	I ₉	P-3	R ₂₆	B-1	S ₃₀
A-8	F ₁₆	F-16	I ₁₀	R-2	R ₂₇	A-1	S ₃₁
B-9	F ₁₇	G-16	I ₁₁	T-1	R ₂₈	G-17	S/DF
A-9	F ₁₈	F-17	I ₁₂	P-2	R ₂₉	H-17	S/DR
B-10	F ₁₉	H-16	I ₁₃	N-3	R ₃₀	H-15	S/DS
C-10	F ₂₀	B-16	MSERR	R-1	R ₃₁	P-16	SIGN
A-10	F ₂₁	U-16	OE _F	T-16	RFSEL ₀	A-16	SLAVE
A-11	F ₂₂	T-14	OE _S	R-15	RFSEL ₁	J-16	TSEL ₀
B-11	F ₂₃	R-17	PSEL ₀	U-17	RFSEL ₂	J-15	TSEL ₁
A-12	F ₂₄	T-17	PSEL ₁	U-13	RM ₀	J-17	TSEL ₂
B-12	F ₂₅	P-15	PSEL ₂	T-12	RM ₁	K-17	TSEL ₃
B-13	F ₂₆	R-16	PSEL ₃	U-14	RM ₂	B-15	VCC
A-13	F ₂₇	K-16	QSEL ₀	P-1	S ₀	F-15	VCC
A-14	F ₂₈	K-15	QSEL ₁	N-2	S ₁	J-3	VCC
C-13	F ₂₉	L-17	QSEL ₂	M-3	S ₂	R-5	VCC
B-14	F ₃₀	L-16	QSEL ₃	N-1	S ₃	R-10	VCC
A-15	F ₃₁	T-11	R ₀	M-2	S ₄	C-5	VCCO
M-17	FLAG ₁	U-12	R ₁	M-1	S ₅	C-9	VCCO
M-16	FLAG ₂	U-11	R ₂	L-3	S ₆	C-11	VCCO
N-17	FLAG ₃	T-10	R ₃	L-2	S ₇	C-15	VCCO
N-16	FLAG ₄	U-10	R ₄	L-1	S ₈	M-15	VCCO

Note: Pin number D-4 = Alignment Pin
VCCO and GNDO are power and ground pins for the output buffers.
VCC and GND are power and ground pins for the rest of the logic.

LOGIC SYMBOL



00418C-6A

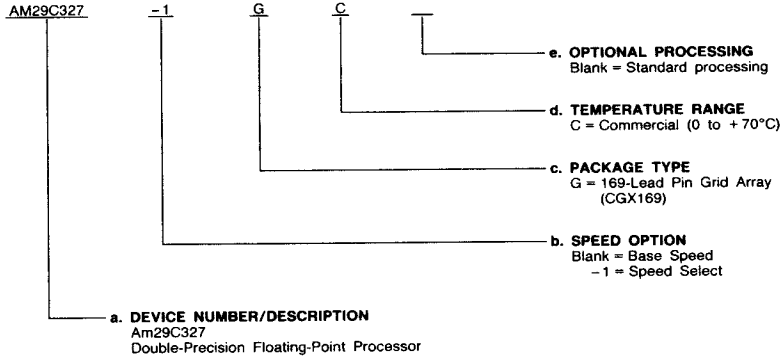
LS003281

ORDERING INFORMATION

Standard Products

AMD standard products are available in several packages and operating ranges. The order number (Valid Combination) is formed by a combination of:

- a. **Device Number**
- b. **Speed Option** (if applicable)
- c. **Package Type**
- d. **Temperature Range**
- e. **Optional Processing**



Valid Combinations	
AM29C327	GC
AM29C327-1	

Valid Combinations

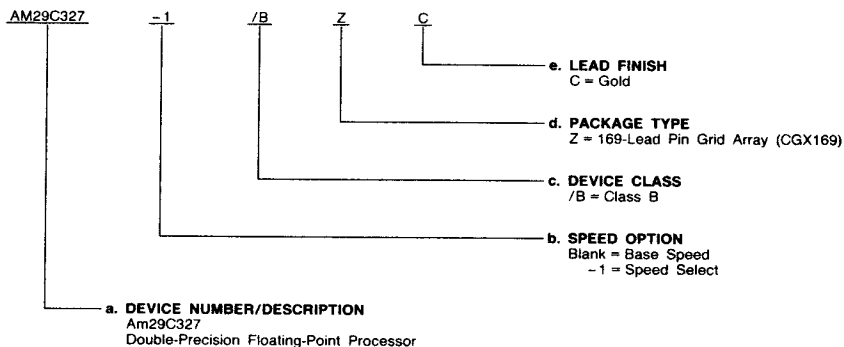
Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations, to check on newly released combinations, and to obtain additional data on AMD's standard military grade products.

MILITARY ORDERING INFORMATION

APL Products

AMD standard products for Aerospace and Defence applications are available in several packages and operating ranges. APL (Approved Products List) products are fully compliant with MIL-STD-883C requirements. The order number (Valid Combination) is formed by a combination of:

- a. **Device Number**
- b. **Speed Option** (if applicable)
- c. **Device Class**
- d. **Package Type**
- e. **Lead Finish**



Valid Combinations	
AM29C327	/BZC
AM29C327-1	

Valid Combinations

Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations, to check on newly released combinations, and to obtain additional data on AMD's standard military grade products.

Group A Tests

Group A tests consist of Subgroups
1, 2, 3, 7, 8, 9, 10, 11.

PIN DESCRIPTION

CLOCK Clock (Input)

Clock input to all registers. The Am29C327 is fully static — no data is lost from the internal registers if the clock is stopped for an extended period.

ENF F-Register Enable (Input; Active LOW)

When $\overline{\text{ENF}}$ is HIGH, the contents of the F-register are static. When $\overline{\text{ENF}}$ is LOW, the 64-bit ALU output is clocked into the F-register on the next LOW-to-HIGH transition of the CLOCK input. As described in the Mode Register Description section, the F-register can be made transparent by setting the mode register bit M17 appropriately, in which case $\overline{\text{ENF}}$ has no effect. This input is not clocked into the instruction register, and must be valid at the LOW-to-HIGH CLOCK transition on which the desired data is to be clocked into the F-register.

ENI Instruction Register Enable (Input; Active LOW)

When $\overline{\text{ENI}}$ is HIGH, the contents of the instruction register are static. When $\overline{\text{ENI}}$ is LOW, the 30-bit instruction word, comprising the fields PSEL₃₋₀, QSEL₃₋₀, TSEL₃₋₀, RM₂₋₀, S/DF and I₃₋₀, is clocked into the instruction register on the next LOW-to-HIGH transition of the CLOCK input. This input is not clocked into the instruction register and must be valid at the LOW-to-HIGH CLOCK transition on which the desired data is to be clocked into the instruction register.

ENR R-Register Enable (Input; Active LOW)

When $\overline{\text{ENR}}$ is HIGH, the contents of the R-register are static. When $\overline{\text{ENR}}$ is LOW, a new 64-bit operand, together with the precision control input S/DR, is clocked into the 65-bit R-register on the next LOW-to-HIGH transition of the CLOCK input. As described in the Input Modes section, the user can select from eight different input modes, as appropriate for the system environment. This input is not clocked into the instruction register and must be valid at the LOW-to-HIGH CLOCK transition on which the desired data is to be clocked into the R-register.

ENRF Register File Enable (Input; Active LOW)

When $\overline{\text{ENRF}}$ is HIGH, the contents of the register file are static. When $\overline{\text{ENRF}}$ is LOW, the 64-bit ALU result, together with a "tag" indicating its precision, is clocked into one of the 65-bit registers RF7 to RF0 on the next LOW-to-HIGH transition of the CLOCK input. The inputs RFSEL₂₋₀ determine which of the eight registers in the register file is the destination for the ALU result and its precision tag. This input is not clocked into the instruction register and must be valid at the LOW-to-HIGH CLOCK transition on which the desired data is to be clocked into the register file.

ENS S-Register Enable (Input; Active LOW)

When $\overline{\text{ENS}}$ is HIGH, the contents of the S-register are static. When $\overline{\text{ENS}}$ is LOW, a new 64-bit operand, together with the precision control input S/DS, is clocked into the 65-bit S-register on the next LOW-to-HIGH transition of the CLOCK input. As described in the Input Modes section, the user can select from eight different input modes, as appropriate for the system environment. This input is not clocked into the instruction register and must be valid at the LOW-to-HIGH CLOCK transition on which the desired data is to be clocked into the S-register.

F31-0 Output Bus (Bidirectional)

The 32-bit output bus is bidirectional to support Master/Slave checking.

FLAG₆₋₁ Flag Outputs (Bidirectional)

The six flag outputs FLAG₆-FLAG₁ report the status of the previous ALU operation. The outputs are bidirectional to support Master/Slave checking.

FSEL Output Multiplexer Control (Input)

When FSEL is HIGH, the most-significant 32 bits of the 64-bit F-register are connected to the drivers on the F₃₁₋₀ output bus. When FSEL is LOW, the least-significant 32 bits of the 64-bit F-register are connected to the drivers on the F₃₁₋₀ output bus. The state of this input pin may be changed at twice the rate of the CLOCK input, to allow a full 64-bit result to be output from the Am29C327 in a single clock cycle. This input is not clocked into the instruction register and must be valid while the required data is accessed via the output bus.

I₃₋₀ ALU Instruction (Input)

I₃₋₀ determine the ALU instruction to be executed in the next cycle. The inputs are clocked into the instruction register under the control of the $\overline{\text{ENI}}$ input.

MSERR Master/Slave Error (Output)

The MSERR output is asserted (HIGH) whenever a Master/Slave error is detected on any enabled output, i.e. F₃₁₋₀ if $\overline{\text{OEF}}$ is LOW, SIGN and FLAG₆₋₁ if $\overline{\text{OES}}$ is LOW.

ÖEF F-Output Enable (Input; Active LOW)

When $\overline{\text{OEF}}$ is HIGH, the F₃₁₋₀ output bus assumes a high-impedance state. When $\overline{\text{OEF}}$ is LOW (and SLAVE is HIGH, specifying "Master" mode), the F₃₁₋₀ output drivers are enabled. This input is not clocked into the instruction register and must be valid at all times.

ÖES Flag Output Enable (Input; Active LOW)

When $\overline{\text{OES}}$ is HIGH, the 7 outputs FLAG₆₋₁ and SIGN assume a high-impedance state. When $\overline{\text{OES}}$ is LOW (and SLAVE is HIGH, specifying "Master" mode), the output drivers for the FLAG₆₋₁ and SIGN outputs are enabled. This input is not clocked into the instruction register and must be valid at all times.

PSEL₃₋₀ P-Input Multiplexer Control (Input)

The PSEL₃₋₀ inputs control the P-Input Multiplexer, selecting the source of operands for the P-Input of the ALU. The PSEL₃₋₀ inputs are clocked into the instruction register under the control of the $\overline{\text{ENI}}$ input.

QSEL₃₋₀ Q-Input Multiplexer Control (Input)

The QSEL₃₋₀ inputs control the Q-Input Multiplexer, selecting the source of operands for the Q-Input of the ALU. The QSEL₃₋₀ inputs are clocked into the instruction register under the control of the $\overline{\text{ENI}}$ input.

R₃₁₋₀ R-Input Bus (Input)

The 32-bit R-Input bus, R₃₁₋₀, is used to load operands into one or both of the input registers, R and S. It is also used to load data into the 32-bit mode register.

RFSEL₂₋₀ Register File Destination Select (Input)

The RFSEL₂₋₀ inputs select which of the registers RF7 through RF0 is the destination for the ALU result. Operands are clocked into the selected register only when the $\overline{\text{ENRF}}$ input is LOW. This input is not clocked into the instruction register and must be valid at the LOW-to-HIGH CLOCK transition on which the desired data is to be clocked into the register file.

RM₂₋₀ Rounding Mode Select (Input)

The RM₂₋₀ inputs select which of the six available rounding modes is to be applied to the next ALU operation. Rounding is discussed in Appendix B. The RM₂₋₀ inputs are clocked into the instruction register under the control of the $\overline{\text{ENI}}$ input.

S₃₁₋₀ S-Input Bus (Input)

The 32-bit S-Input bus, S₃₁₋₀, is used to load operands into one or both of the input registers, R and S.

S/DF F-Precision Control (Input)

When S/DF is HIGH, the next ALU operation produces a single-precision (32-bit) result. When S/DF is LOW, the next ALU operation produces a double-precision (64-bit) result. The S/DF input is clocked into the instruction register under the control of the ENI input.

S/DR R-Precision Control (Input)

When S/DR is HIGH, the data clocked into the R-register is treated as single-precision (32-bit) by the processor. When S/DR is LOW, the data clocked into the R-register is treated as double-precision (64-bit) by the processor. The S/DR input is clocked into the 65th bit of the R-register as the "precision tag" for the R-operand, under the control of the ENR input.

S/DS S-Precision Control (Input)

When S/DS is HIGH, the data clocked into the S-register is treated as single-precision (32-bit) by the processor. When S/DS is LOW, the data clocked into the S-register is treated as double-precision (64-bit) by the processor. The S/DS input is clocked into the 65th bit of the S-register as the "precision tag" for the S-operand, under the control of the ENS input.

SIGN Sign Flag (Bidirectional)

If the result of the previous ALU operation was negative, the SIGN output is HIGH (provided that OES is LOW). If the result of the previous ALU operation was not negative, the SIGN output is LOW (provided that OES is LOW). The output is bidirectional to support Master/Slave checking.

SLAVE Master/Slave Mode Select (Input; Active LOW)

When SLAVE is HIGH, the "Master" mode of operation is selected. When SLAVE is LOW, the "Slave" mode of operation is selected and all outputs except MSERR are disabled (high-impedance). This input is not clocked into the instruction register and must be valid at all times.

TSEL3-0 T-Input Multiplexer Control (Input)

The TSEL3-0 inputs control the T-Input Multiplexer, selecting the source of operands for the T-Input of the ALU. The TSEL3-0 inputs are clocked into the instruction register under the control of the ENI input.

VCC GND Power

Power supply pins for the internal logic.

VCCO GND0 Power

Power supply pins for the output buffers.

FUNCTIONAL DESCRIPTION**Overview**

The Am29C327 is a high-performance, single-chip, double-precision floating-point processor.

Architecture

The Am29C327 comprises a high-speed ALU, a 64-bit data path, and control circuitry.

The core of the Am29C327 is a 64-bit floating-point/integer ALU. This ALU takes operands from three 64-bit input ports and performs the selected operation, placing the result on a 64-bit output port. Thirteen ALU flags report operation status via the FLAG6-1 and SIGN outputs. The ALU is completely combinatorial for minimum latency; optional pipelining is available to boost throughput for array operations.

The data path consists of the 32-bit input buses R and S; two 64-bit input operand registers; an 8-by-64-bit register file for storage of intermediate results; three operand-selection multiplexers that provide for orthogonal selection of input operands; a 64-bit output register; and an output multiplexer that permits the selection of 32 MSBs, or 32 LSBs of data. Input operands enter the processor through the R and S buses, and are then demultiplexed and buffered for subsequent storage in registers R and S. The operand selection multiplexers route the operands to the ALU. Operation results are stored in register F, and leave the device on the 32-bit output bus F. The results can also be stored in the register file for use in subsequent operations.

Instruction Set

The Am29C327 implements 58 arithmetic and logical instructions. Thirty-five instructions operate on floating-point numbers; these instructions fall into the following categories:

- Addition/subtraction
- Multiplication
- Multiplication-accumulation
- Comparison
- Selecting the maximum or minimum of two numbers

- Rounding to integral value
- Absolute value, negation, pass
- Reciprocal seed generation
- Conversion between any of the supported floating-point formats
- Conversion of a floating-point number to an integer format, with or without a scale factor

By concatenating these operations, the user can also perform division, square-root extraction, polynomial evaluation, and other functions not implemented directly.

Twenty-two instructions operate on integers, and belong to the following general categories:

- Addition/subtraction
- Multiplication
- Comparison
- Selecting the maximum or minimum of two numbers
- Absolute value, negation, pass
- Logical operations; e.g., AND, OR, XOR, NOT
- Arithmetic, logical, and funnel shifts
- Conversion between single- and double-precision integer formats
- Conversion of an integer number to a floating-point format, with or without a scale factor

One special instruction is provided to move data from the P-Port to the F-Port, and another to load the mode register.

Mixed-Precision Operations

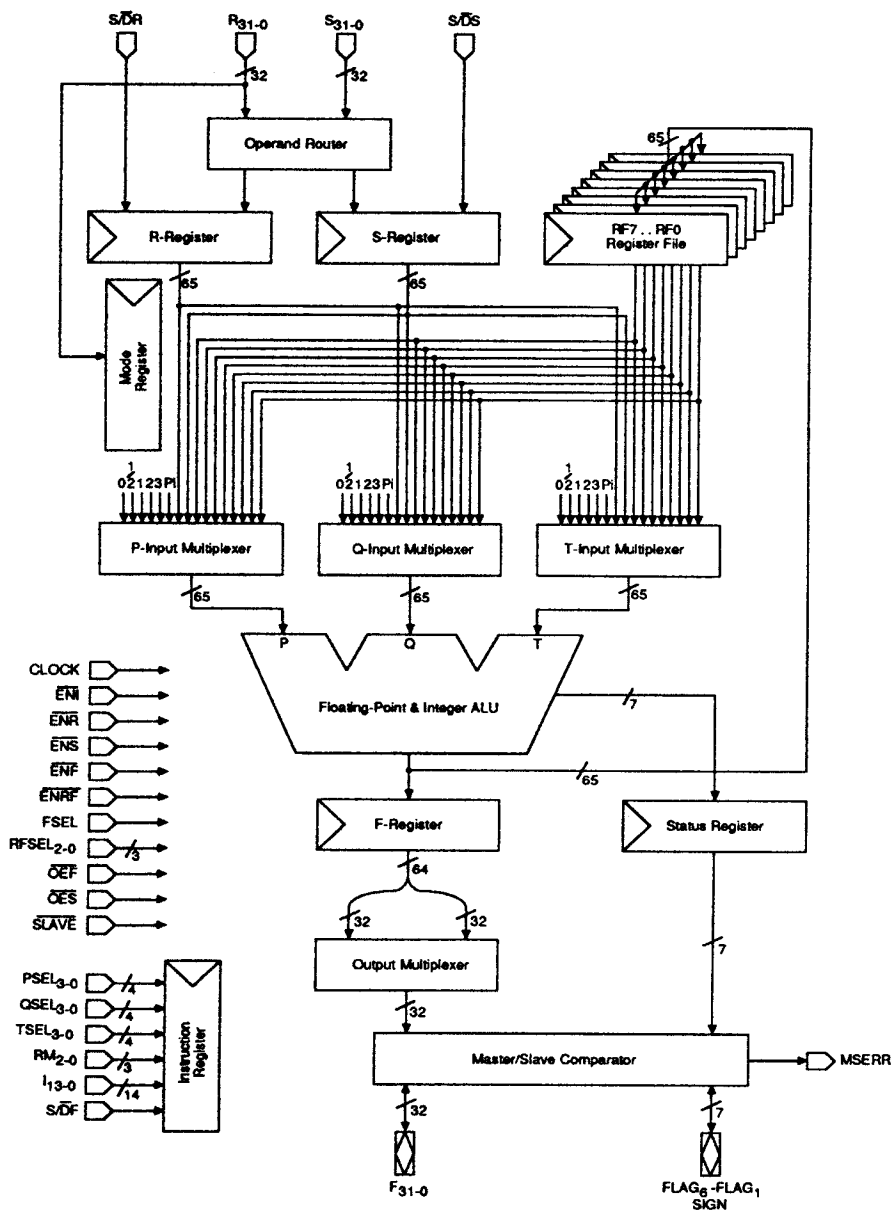
All Am29C327 instructions, floating-point or integer, can be performed with either single- or double-precision operands. In addition, the user can elect to mix precisions within an operation. All operations are performed in double-precision internally; the user specifies the precisions of the input operands and the required precision for the output operand. The necessary precision conversions are made in concert with the selected operation, with no additional cycle-time overhead.

I/O Modes

The Am29C327 supports eight I/O modes that afford flexible interface to a variety of 32- and 64-bit systems.

Fault Detection Features

The Am29C327 contains special comparison hardware to allow the operation of two processors in parallel, with one processor (the slave) checking the results produced by the other (the master). This feature is of particular importance in the design of high-reliability systems.



06418C-7A

BD008101

Figure 1. Block Diagram - Am29C327 Double-Precision Floating-Point Processor

Block Diagram Description

A block diagram of the Am29C327 is shown in Figure 1. The Am29C327 comprises the input registers, the operand selection multiplexers, the instruction register, the ALU, the output register/the register file, the status register, the output selection multiplexer, the mode register, and the master/slave comparator.

Input Registers/Input Modes

Operands are loaded into the processor through the R and S buses, and are then demultiplexed and buffered for subsequent storage in the 65-bit registers R and S. Input operands may be either single-precision (32-bit) or double-precision (64-bit) as specified by S/ $\overline{D}R$ and S/ $\overline{D}S$. Accompanying the input registers are two 32-bit temporary registers, that allow for the overlapping of operand transfers and ALU operations. This arrangement of temporary registers and demultiplexers permits data and corresponding precision bit S/ $\overline{D}R$ or S/ $\overline{D}S$ to be loaded into the 65-bit R register and 65-bit S register via one of the eight input modes:

1. 32-bit-bus, double-cycle, LSWs first
2. 32-bit-bus, double-cycle, MSWs first
3. 32-bit-bus, single-cycle, LSWs first
4. 32-bit-bus, single-cycle, MSWs first
5. 64-bit-bus, double-cycle, R first
6. 64-bit-bus, double-cycle, S first
7. 64-bit-bus, single-cycle, R first
8. 64-bit-bus, single-cycle, S first

These modes are described in detail in the Input Modes section.

Operand Selection Multiplexers

The operand selection multiplexers route operands to the ALU. These multiplexers, as well as selecting operands from input registers R and S and register file locations RF7 – RF0, also have access to a set of constants (0, 0.5, 1, 2, 3, Pi). These constants are double-precision preprogrammed numbers for use in ALU operations, and are automatically provided in the appropriate floating-point or integer format.

Instruction Register

The instruction register stores a 32-bit word specifying the current processor operation. Included in the instruction word are fields that specify the P, Q, and T multiplexer selects, the rounding modes; the core operation to be performed by the ALU; sign-change controls for ALU input and result operands; and the single/double-precision control for the output operand. The multiplexer select controls and the instruction word are described in detail in the Instruction Set section; Rounding modes are described in Appendix B.

ALU

The ALU is a combinatorial arithmetic/logic unit that performs a large repertoire of floating-point and integer operations. The

ALU has three operand inputs. Some operations require a single input operand, for example, conversion operations. Others, such as addition and multiplication, require two input operands. The multiplication-accumulation and funnel shift operations require three input operands. Many ALU arithmetic operations allow for the independent control of operand signs, thus greatly increasing the number of arithmetic expressions that can be evaluated in a single ALU pass.

The ALU can be configured in either a flow-through mode, for which the ALU is completely combinatorial, or a pipelined mode, for which ALU operations incur one or two pipeline delays, but which results in a higher throughput than flow-through mode.

A detailed description of ALU operations appears in the Instruction Set section.

Output Register/Register File

The results of the operations performed by the ALU are stored in the 64-bit output register F. Results can also be stored in the 8-by-64-bit register file for use in subsequent operations. Each register file location contains a 65th bit indicating the precision of the operand stored in that location, thus permitting the ALU to correctly process the operand in subsequent operations.

Status Register

The status register is a 7-bit register that stores flags pertaining to the most recently performed operation. A detailed description is provided in the Instruction Set section.

Output Multiplexer

The output multiplexer routes operation results to the F bus. This multiplexer selects either the 32 MSBs or the 32 LSBs of the data stored in the output register.

Master/Slave Comparator

Each Am29C327 output signal has associated logic that compares that signal with the signal that the processor is providing internally to the output driver; any discrepancies are indicated by assertion of signal MSERR.

For a single processor, this output comparison detects short circuits in output signals or defective output drivers, but does not detect open circuits. It is possible to connect a second processor in parallel with the first, with the second processor's outputs disabled by assertion of signal SLAVE. The second processor detects open-circuit signals, as well as providing a check of the outputs of the first.

Mode Register

The mode register contains processor control parameters that are changed infrequently. The 32-bit mode word is loaded into the register via the R bus. A detailed description of the mode register is provided in the Mode Register Description section.

Mode Register Description

The 32-bit mode register contains parameters specifying the overall operating mode of the Am29C327. These parameters are typically not changed on an instruction-by-instruction basis. The register is loaded from the R-port R₃₁.0, using the "Load Mode Register" instruction. This section provides a comprehensive explanation of the function of each field within the register.

Bits M31-M21

Reserved for factory test and future upgrades. Must be set to logic 0.

Bits M20-M19 — Pipeline Mode Select

This field determines whether the ALU operates in flow-through mode or pipelined mode, and specifies whether pipelined multiply-accumulate operations are single-pipelined or double-pipelined:

M20 M19

- | | | |
|---|---|--|
| 0 | X | The ALU operates in flow-through mode for all operations — all pipeline registers are transparent. |
| 1 | 0 | The ALU operates in single-pipelined mode for all operations. |
| 1 | 1 | The ALU operates in double-pipelined mode for multiply-accumulate operations. The ALU operates in single-pipelined mode for <u>all other</u> operations. |

A detailed description of the pipeline modes is contained in the Pipelining of Operations section.

Bit M18 — Status Register Feedthrough Enable

If M18 is HIGH, the 7-bit status register is made transparent and operates in flow-through mode. This mode is generally used when it is necessary to minimize the overall latency of the processor. If M18 is LOW, the status register operates in clocked mode, status information being clocked into the register from the ALU on every rising edge of the clock input. This mode is generally used when it is necessary to maximize the overall throughput of the processor.

Bit M17 — F-Register Feedthrough Enable

If M17 is HIGH, the 64-bit F-register is made transparent and operates in flow-through mode. This mode is generally used when it is necessary to minimize the overall latency of the processor. If M17 is LOW, the F-register operates in clocked mode, ALU results being clocked into the register from the ALU on every rising edge of the clock input, provided that ENF is LOW. This mode is generally used when it is necessary to maximize the overall throughput of the processor.

Bits M16-M14 — Input Mode Select

This field determines which of the eight available modes is used to input operands to the R-register and S-register:

M16	M15	M14	Input Mode
0	0	0	32-bit bus, single-cycle, LSW first.
0	0	1	32-bit bus, single-cycle, MSW first.
0	1	0	32-bit bus, double-cycle, LSW first.
0	1	1	32-bit bus, double-cycle, MSW first.
1	0	0	64-bit bus, single-cycle, R first.
1	0	1	64-bit bus, single-cycle, S first.
1	1	0	64-bit bus, double-cycle, R first.
1	1	1	64-bit bus, double-cycle, S first.

A detailed description of the input modes is contained in the Input Mode section.

Bits M13-M12 — Integer Multiplication Format Adjust

This field determines the output format for integer multiplication operations, selecting either the MSBs or the LSBs of the product and specifying whether or not format-adjusting (i.e., shifting the product one place left before the MSBs/LSBs selection) is performed:

M13	M12	Integer Multiplication Output
0	0	LSBs selected, no format-adjust performed.
0	1	LSBs selected, format-adjust performed.
1	0	MSBs selected, no format-adjust performed.
1	1	MSBs selected, format-adjust performed.

This field has no effect on operations other than integer multiplications.

Bit M11 — Integer Multiplication Signed/Unsigned

If M11 is HIGH, the input operands for integer multiplications are treated as two's-complement (signed) operands and two's-complement multiplications are performed. If M11 is LOW, the input operands for integer multiplications are treated as unsigned operands and unsigned multiplications are performed.

This bit has no effect on operations other than integer multiplications.

Bit M10

Reserved for factory test and future upgrades. Must be set to logic 0.

Bit M9 — IBM Underflow Mask Enable

If M9 is HIGH, underflowed results in IBM format are output with the (underflowed) exponent increased by 128 to bring its value into the representable range. If M9 is LOW, underflowed results in IBM format are replaced by an IBM "True Zero" (sign = 0, exponent = 0, fraction = 0).

This bit has no effect on operations for which the result format is not IBM.

Bit M8 — IBM Significance Mask Enable

If M8 is HIGH, results in IBM format that contain a non-zero exponent and a zero fraction (known as IBM "Floating-Point Zeros") are output unchanged. If M8 is LOW, results in IBM format that contain a non-zero exponent and a zero fraction are replaced by an IBM "True Zero" (sign = 0, exponent = 0, fraction = 0).

This bit has no effect on operations for which the result format is not IBM.

Bit M7 — IEEE Sudden Underflow Enable

If M7 is HIGH (and M6 is LOW, disabling IEEE traps), results in IEEE format that are denormalized are replaced by a zero of the same sign ("Sudden Underflow" mode). If M7 is LOW (and M6 is LOW, disabling IEEE traps), results in IEEE format that are denormalized are output unchanged as valid denormalized numbers ("Gradual Underflow" mode).

This bit has no effect on operations for which the result format is not IEEE or on operations for which the result format is IEEE but traps are enabled.

Bit M6 — IEEE Trapped Operation Enable

If M6 is HIGH, the exponents of IEEE overflowed results are reduced by 192 (single-precision) or 1536 (double-precision) to bring them into the representable range, the signs and fractions being unchanged, regardless of the setting of M4 (Saturate Enable). Similarly, if M6 is HIGH, the exponents of IEEE underflowed and denormalized results are increased by 192 (single-precision) or 1536 (double-precision) to bring them

into the representable range, the signs and fractions being unchanged, regardless of the setting of M7 (Sudden Underflow Enable). If M6 is LOW, the final results of overflowed and underflowed IEEE operations are determined by bits M4 (Saturate Enable) and M7 (Sudden Underflow Enable) respectively.

This bit has no effect on operations for which the result format is not IEEE.

Bit M5 — IEEE Affine/Projective Mode Select

If M5 is HIGH, IEEE infinities are interpreted in the "Affine" sense for addition, comparison and multiply-accumulate operations. If M5 is LOW, IEEE infinities are interpreted in the "Projective" sense for addition, comparison and multiply-accumulate operations. The difference between affine and projective interpretations is summarized below.

Affine and Projective Interpretations of Infinities

Operation	Affine Result & Flags	Projective Result & Flags
(+ oo) + (+ oo)	+ oo; No flags	Quiet Nan; Flags I, R
(- oo) + (- oo)	- oo; No flags	Quiet Nan; Flags I, R

Flag I is the "Invalid Operation" flag. Flag R is the "Reserved Operand" flag.

This bit has no effect on base operations other than IEEE addition, IEEE comparison and IEEE multiply-accumulate.

Bit M4 — Saturate Enable

If M4 is HIGH, overflowed results are replaced by the largest representable number in the result format, with the same sign as the overflowed result, unless the result format is IEEE and M6 is HIGH, enabling IEEE trapped operation. If M4 is LOW, overflowed results are replaced by infinities (IEEE results, provided that M6 is LOW, disabling trapped operations), replaced by DEC Reserved Operands (DEC results) or left unchanged (IBM and integer results).

When IEEE traps are enabled (M6 is HIGH), this bit has no effect on operations for which the result format is IEEE.

Bits M3-M2 — Alternate Floating-Point Format Select

This field determines the alternate floating-point format:

M3	M2	Alternate Floating-Point Format (Double/Single)
0	0	IEEE (IEEE double-precision/IEEE single-precision)
0	1	DEC D (DEC D/DEC F)
1	0	DEC G (DEC G/DEC F)
1	1	IBM (IBM double-precision/IBM single-precision)

The alternate floating-point format is used for floating-point conversion operations to specify the destination format for the operation "Convert T to Alternate Format" and to specify the source format for the operation "Convert T from Alternate Format".

The floating-point formats are specified according to their double-precision names, since the information regarding the

precisions of operands is provided to the processor via the S/DR, S/DS and S/DF inputs. Note that the two, distinct, double-precision DEC formats, DEC D and DEC G, have the same single-precision format, DEC F.

Bits M1-M0 — Primary Floating-Point Format Select

This field determines the primary floating-point format:

M1	M0	Primary Floating-Point Format (Double/Single)
0	0	IEEE (IEEE double-precision/IEEE single-precision)
0	1	DEC D (DEC D/DEC F)
1	0	DEC G (DEC G/DEC F)
1	1	IBM (IBM double-precision/IBM single-precision)

The primary floating-point format is the format in which all floating-point inputs are assumed to be represented and in which all floating-point results are generated, taking into account specified precisions, with the following exceptions: the T-input for the operation "Convert T from Alternate Format" is assumed to be the alternate floating-point format and the result of the operation "Convert T to Alternate Format" is generated in the alternate floating-point format, again taking into account specified precisions.

The floating-point formats are specified according to their double-precision names, since the information regarding the precisions of operands is provided to the processor via the S/DR, S/DS and S/DF inputs. Note that the two, distinct, double-precision DEC formats, DEC D and DEC G, have the same single-precision format, DEC F.

The mode register bits, which affect the arithmetic result of an operation rather than controlling data movement within the processor, are bits M13 through M4. The table below indicates the settings that should be used for these 10 bits to ensure strict conformance with the IEEE, DEC, and IBM standards. As regards IEEE traps and IBM masks, the relevant standards specify that these bits be set as dictated by the application. In all cases, of course, bits M1 and M0 must be set to specify the desired primary format.

It should be noted that the IEEE, DEC, and IBM standards do not explicitly define conversion operations between floating-point formats. When executing these operations, the appropriate settings for the mode register are, in general, determined by the system environment and the algorithms to be executed. Bits M3 and M2 must be set to specify the required alternate floating-point format.

Mode Settings for Exact Compliance with Standards

Bit	Function	IEEE	DEC	IBM
13-12	Int FMTADJ	X	X	X
11	Int SIGNSSEL	X	X	X
9	IBM UNFMASK	X	X	As Req'd.
8	IBM SIGMASK	X	X	As Req'd.
7	IEEE SUEN	0	X	X
6	IEEE TRAPEN	As Req'd.	X	X
5	IEEE AFF/PROJ	1	X	X
4	SATEN	0	0	0

"0" = set to logic 0.

"1" = set to logic 1.

"X" = don't care.

Input Modes

The Am29C327 supports a total of eight input modes for loading data into the R and S registers.

The 32-bit bus modes allow the user to connect each input port ($R_{31} - R_0$ and $S_{31} - S_0$) to a separate 32-bit bus. 64-bit operands can then be loaded by placing the MSBs and LSBs alternately on the appropriate ports. In the 64-bit bus modes, the two input ports are configured internally as a single 64-bit port. The Am29C327 may then be connected directly to a 64-bit bus, and 64-bit operands may be loaded in a single operation. Either the 32-bit bus modes or the 64-bit bus modes may be used regardless of the precision of the operands being transferred — the choice of input modes will in practice be determined by the system into which the Am29C327 is to be integrated.

Single-cycle input modes allow two 64-bit operands to be loaded in a single clock cycle. This necessitates driving the input buses at twice the speed of the Am29C327. In systems where this is not practical, the double-cycle modes allow the loading of one 64-bit operand (or two 32-bit operands) per clock cycle.

Data may be loaded from the input buses to the R register and S register using one of the eight input modes:

1. 32-Bit Bus, Single-Cycle, LSWs First
2. 32-Bit Bus, Single-Cycle, MSWs First
3. 32-Bit Bus, Double-Cycle, LSWs First
4. 32-Bit Bus, Double-Cycle, MSWs First
5. 64-Bit Bus, Single-Cycle, R First
6. 64-Bit Bus, Single-Cycle, S First
7. 64-Bit Bus, Double-Cycle, R First
8. 64-Bit Bus, Double-Cycle, S First

The choice of input mode is determined by mode register bits M16 – M14.

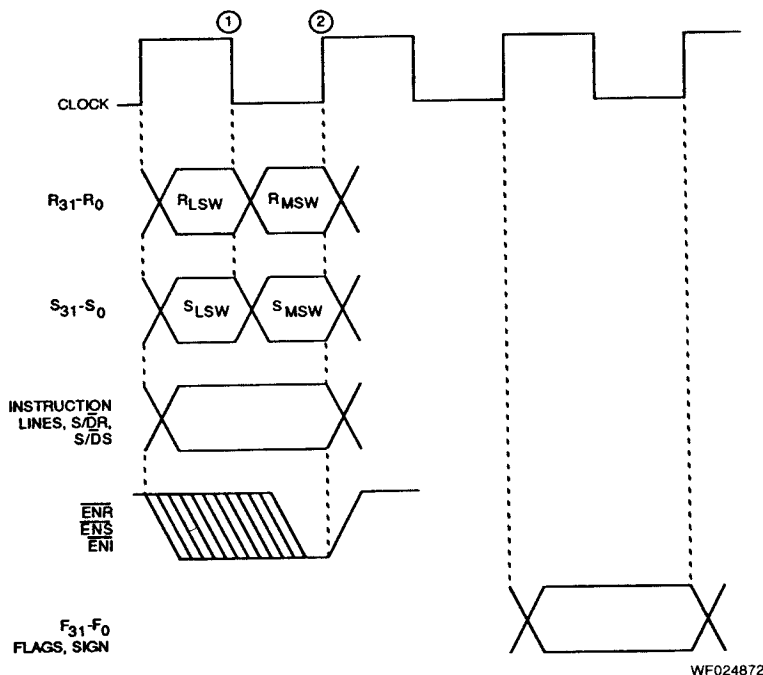
In order to permit the loading of new operands to be overlapped with the execution of a current operation, temporary registers are provided within the "operand router" block (shown in Figure 1). The operation of these temporary registers is transparent to the user. The conditions under which they are loaded depends on the input mode selected.

The eight input modes are described in the following pages.

32-Bit Bus, Single-Cycle, LSW First (M16 = 0, M15 = 0, M14 = 0)

In this mode, the two halves of the 64-bit R operand are placed on the R-port in successive half-cycles, with the S

operand similarly placed on the S-port. After one complete cycle, the R and S registers contain the R and S operands, respectively.



Timing of Operations with Input Mode 1 (32-Bit Bus, Single-Cycle, LSW First)*

*Assumes processor flow-through mode, F register, and status register clocked.

The temporary registers are clocked on every HIGH-to-LOW clock transition.

At 1, the least-significant 32 bits of the R operand are loaded from the R-input port into the R-temp register, and the least-significant 32 bits of the S operand are loaded from the S-input port into the S-temp register. Both words are loaded on the HIGH-to-LOW transition of the clock.

At 2, the most-significant 32 bits of the R operand are loaded from the R-input port into the most-significant half of the R register, and the most-significant 32 bits of the S operand are loaded from the S-input port into the most-significant half of the S register.

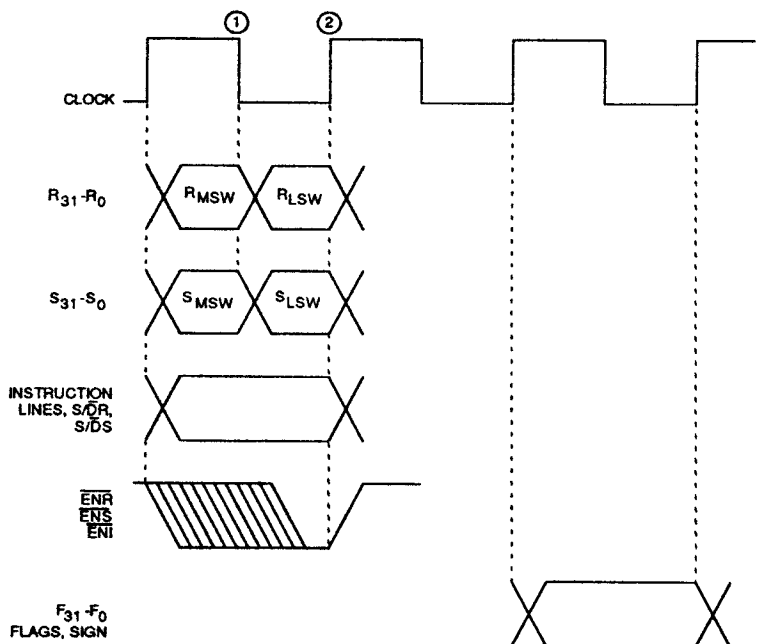
At the same time, at 2, the output of the R-temp register is loaded into the least-significant half of the R register, and the output of the S-temp register is loaded into the least-significant half of the S register.

If the R-operand is a single-precision (32-bit) floating-point operand, it is input in the R-MSW position (with S/DR = HIGH) and the data in the R-LSW position is ignored by the processor. If the R-operand is a single-precision (32-bit) integer operand, it is input in the R-LSW position (with S/DR = HIGH) and the data in the R-MSW position is ignored by the processor. The same procedure applies to the S-operand, if it is a single-precision operand.

32-Bit Bus, Single-Cycle, MSW First (M16 = 0, M15 = 0, M14 = 1)

In this mode, the two halves of the 64-bit R operand are placed on the R-port in successive half-cycles, with the S

operand similarly placed on the S-port. After one complete cycle, the R and S registers contain the R and S operands, respectively.



WF024892

Timing of Operations with Input Mode 2 (32-Bit Bus, Single-Cycle, MSW First)*

*Assumes processor flow-through mode, F register, and status register clocked.

The temporary registers are clocked on every HIGH-to-LOW clock transition.

At 1, the most-significant 32 bits of the R operand are loaded from the R-input port into the R-temp register, and the most-significant 32 bits of the S operand are loaded from the S-input port into the S-temp register. Both words are loaded on the HIGH-to-LOW transition of the clock.

At 2, the least-significant 32 bits of the R operand are loaded from the R-input port into the least-significant half of the R register, and the least-significant 32 bits of the S operand are loaded from the S-input port into the least-significant half of the S register.

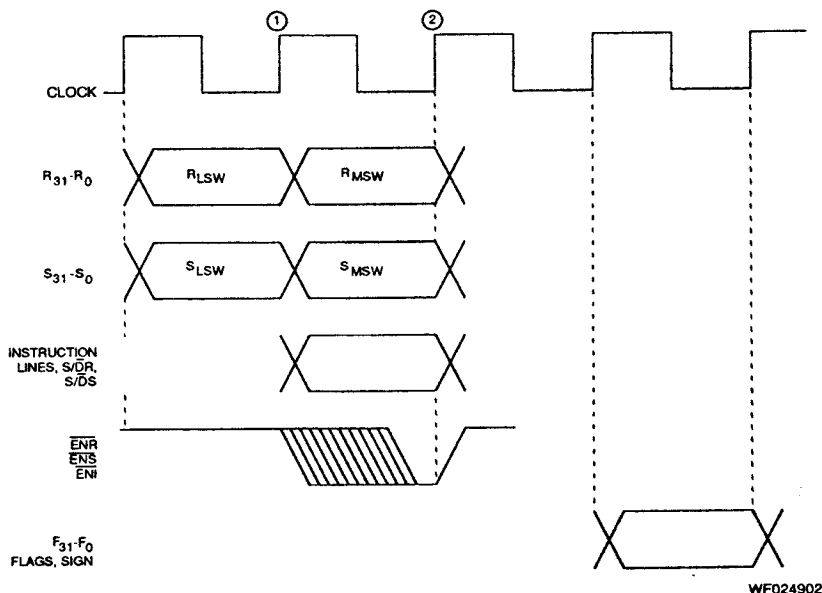
At the same time, at 2, the output of the R-temp register is loaded into the most-significant half of the R register, and the output of the S-temp register is loaded into the most-significant half of the S register.

If the R-operand is a single-precision (32-bit) floating-point operand, it is input in the R-MSW position (with S/DR = HIGH) and the data in the R-LSW position is ignored by the processor. If the R-operand is a single-precision (32-bit) integer operand, it is input in the R-LSW position (with S/DR = HIGH) and the data in the R-MSW position is ignored by the processor. The same procedure applies to the S-operand, if it is a single-precision operand.

32-Bit Bus, Double-Cycle, LSW First (M16 = 0, M15 = 1, M14 = 0)

In this mode, the two halves of the 64-bit R operand are placed on the R-port in successive cycles, with the S operand

similarly placed on the S-port. After two cycles, the R and S registers contain the R and S operands, respectively.



WF024902

**Timing of Operations with Input Mode 3
(32-Bit Bus, Double-Cycle, LSW First)***

*Assumes processor flow-through mode, F register, and status register clocked.

The temporary registers are clocked on every LOW-to-HIGH clock transition.

At 1, the least-significant 32 bits of the R operand are loaded from the R-input port into the R-temp register, and the least-significant 32 bits of the S operand are loaded from the S-input port into the S-temp register.

At 2, the most-significant 32 bits of the R operand are loaded from the R-input port into the most-significant half of the R register, and the most-significant 32 bits of the S operand are loaded from the S-input port into the most-significant half of the S register.

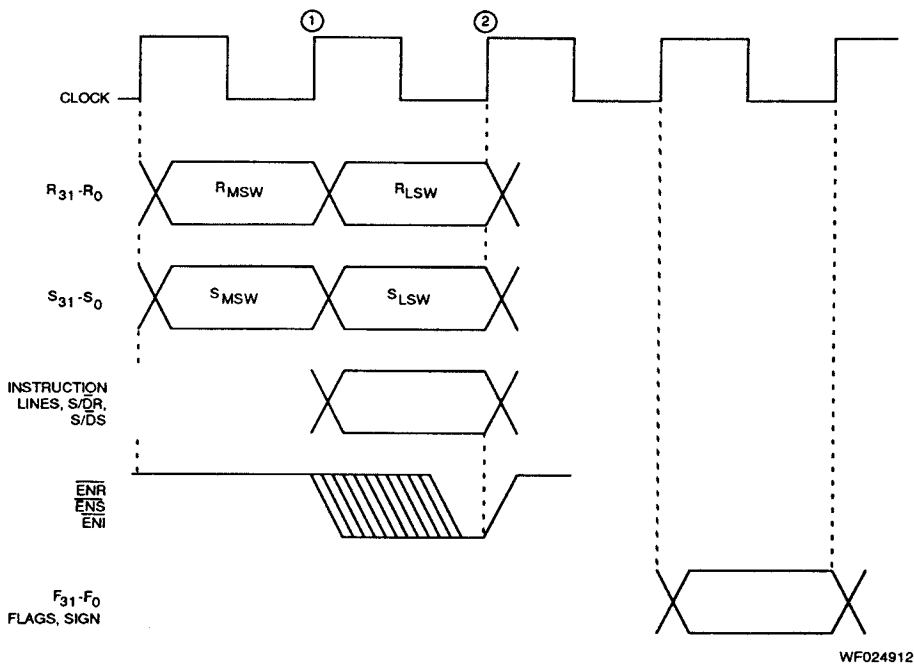
At the same time, at 2, the output of the R-temp register is loaded into the least-significant half of the R register, and the output of the S-temp register is loaded into the least-significant half of the S register.

If the R-operand is a single-precision (32-bit) floating-point operand, it is input in the R-MSW position (with S/DR = HIGH) and the data in the R-LSW position is ignored by the processor. If the R-operand is a single-precision (32-bit) integer operand, it is input in the R-LSW position (with S/DR = HIGH) and the data in the R-MSW position is ignored by the processor. The same procedure applies to the S-operand, if it is a single-precision operand.

32-Bit Bus, Double-Cycle, MSW First (M16 = 0, M15 = 1, M14 = 1)

In this mode, the two halves of the 64-bit R operand are placed on the R-port in successive cycles, with the S operand

similarly placed on the S-port. After two cycles, the R and S registers contain the R and S operands, respectively.



WF024912

**Timing of Operations with Input Mode 4
(32-Bit Bus, Double-Cycle, MSW First)***

*Assumes processor flow-through mode, F register, and status register clocked.

The temporary registers are clocked on every LOW-to-HIGH clock transition.

At 1, the most-significant 32 bits of the R operand are loaded from the R-input port into the R-temp register, and the most-significant 32 bits of the S operand are loaded from the S-input port into the S-temp register.

At 2, the least-significant 32 bits of the R operand are loaded from the R-input port into the least-significant half of the R register, and the least-significant 32 bits of the S operand are loaded from the S-input port into the least-significant half of the S register.

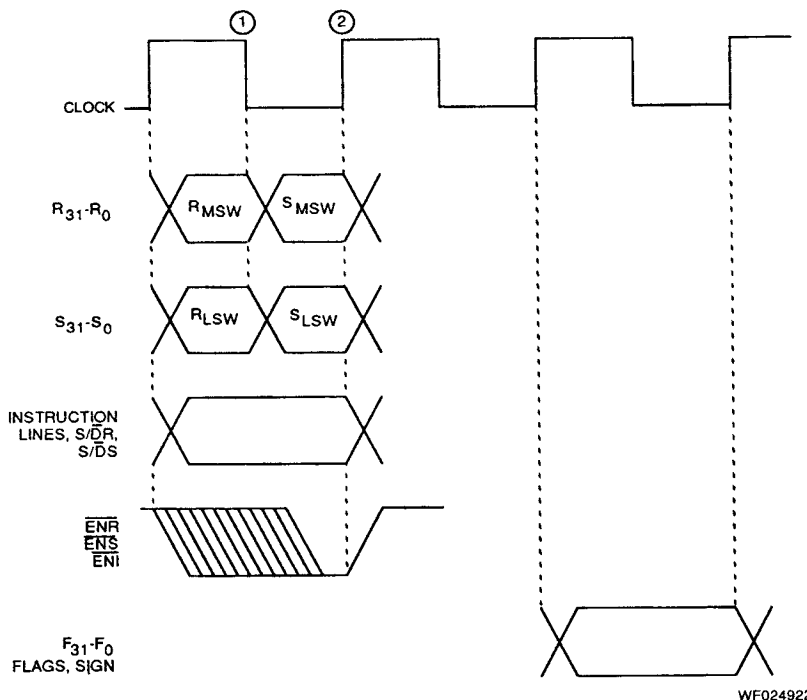
At the same time, at 2, the output of the R-temp register is loaded into the most-significant half of the R register, and the output of the S-temp register is loaded into the most-significant half of the S register.

If the R-operand is a single-precision (32-bit) floating-point operand, it is input in the R-MSW position (with S/DR = HIGH) and the data in the R-LSW position is ignored by the processor. If the R-operand is a single-precision (32-bit) integer operand, it is input in the R-LSW position (with S/DR = HIGH) and the data in the R-MSW position is ignored by the processor. The same procedure applies to the S-operand, if it is a single-precision operand.

64-Bit Bus, Single-Cycle, R First (M16 = 1, M15 = 0, M14 = 0)

In this mode, the MSW of the 64-bit R operand is placed on the R-port and the LSW on the S-port. Both halfwords are

loaded in the first half cycle. Similarly, the two halves of the S operand are loaded in the second half cycle. After one full cycle, the R and S registers contain the R and S operands, respectively.



**Timing of Operations with Input Mode 5
(64-Bit Bus, Single-Cycle, R First)***

*Assumes processor flow-through mode, F register, and status register clocked.

The temporary registers are clocked on every HIGH-to-LOW clock transition.

At 1, the most-significant 32 bits of the R operand are loaded from the R-input port into the R-temp register, and the least-significant 32 bits of the R operand are loaded from the S-input port into the S-temp register.

At 2, the most-significant 32 bits of the S operand are loaded from the R-input port into the most-significant half of the S register, and the least-significant 32 bits of the S operand are loaded from the S-input port into the least-significant half of the S register.

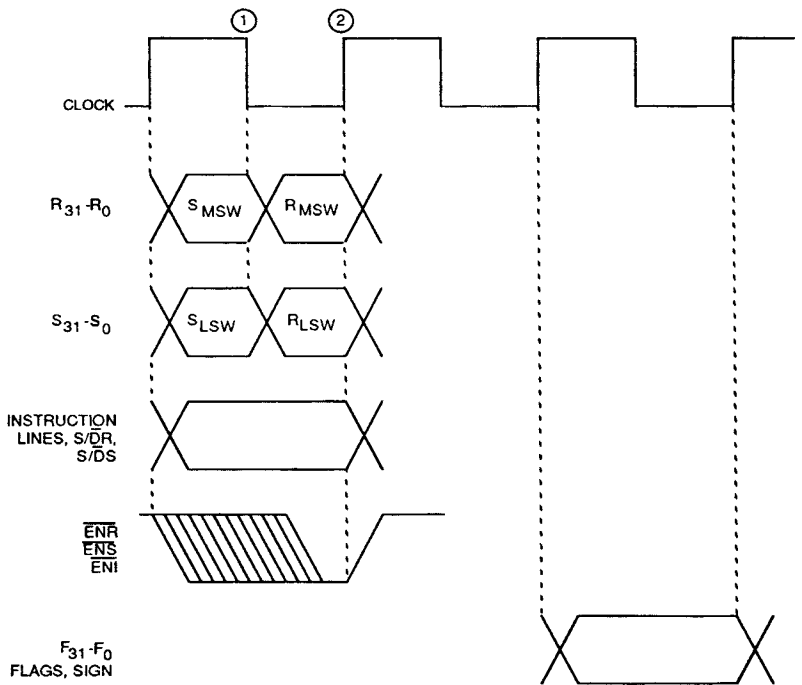
At the same time, at 2, the output of the R-temp register is loaded into the most-significant half of the R register, and the output of the S-temp register is loaded into the least-significant half of the R register.

If the R-operand is a single-precision (32-bit) floating-point operand, it is input in the R-MSW position (with S/D, R = HIGH) and the data in the R-LSW position is ignored by the processor. If the R-operand is a single-precision (32-bit) integer operand, it is input in the R-LSW position (with S/D, R = HIGH) and the data in the R-MSW position is ignored by the processor. The same procedure applies to the S-operand, if it is a single-precision operand.

64-Bit Bus, Single-Cycle, S First (M16 = 1, M15 = 0, M14 = 1)

In this mode, the MSW of the 64-bit S operand is placed on the R-port and the LSW on the S-port. Both halfwords are loaded

in the first half cycle. Similarly, the two halves of the R operand are loaded in the second half cycle. After one full cycle, the R and S registers contain the R and S operands, respectively.



WF024932

**Timing of Operations with Input Mode 6
(64-Bit Bus, Single-Cycle, S First)***

*Assumes processor flow-through mode, F register, and status register clocked.

The temporary registers are clocked on every HIGH-to-LOW clock transition.

At 1, the most-significant 32 bits of the S operand are loaded from the R-input port into the R-temp register, and the least-significant 32 bits of the S operand are loaded from the S-input port into the S-temp register.

At 2, the most-significant 32 bits of the R operand are loaded from the R-input port into the most-significant half of the R register, and the least-significant 32 bits of the R operand are loaded from the S-input port into the least-significant half of the R register.

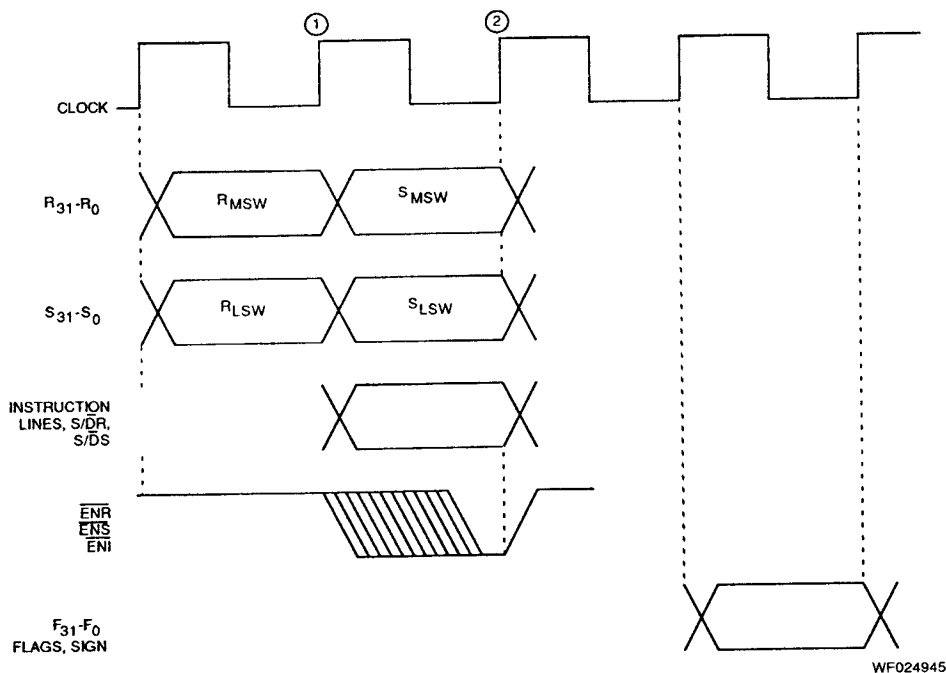
At the same time, at 2, the output of the R-temp register is loaded into the most-significant half of the S register, and the output of the S-temp register is loaded into the least-significant half of the S register.

If the R-operand is a single-precision (32-bit) floating-point operand, it is input in the R-MSW position (with S/DR = HIGH) and the data in the R-LSW position is ignored by the processor. If the R-operand is a single-precision (32-bit) integer operand, it is input in the R-LSW position (with S/DR = HIGH) and the data in the R-MSW position is ignored by the processor. The same procedure applies to the S-operand, if it is a single-precision operand.

64-Bit Bus, Double-Cycle, R First (M16 = 1, M15 = 1, M14 = 0)

In this mode, the MSW of the 64-bit R operand is placed on the R-port and the LSW on the S-port. Both halfwords are

loaded in the first cycle. Similarly, the two halves of the S operand are loaded in the second cycle. After two cycles, the R and S registers contain the R and S operands, respectively.



Timing of Operations with Input Mode 7 (64-Bit Bus, Double-Cycle, R First)*

*Assumes processor flow-through mode, F register, and status register clocked.

The temporary registers are clocked on every LOW-to-HIGH clock transition.

At 1, the most-significant 32 bits of the R operand are loaded from the R-input port into the R-temp register, and the least-significant 32 bits of the R operand are loaded from the S-input port into the S-temp register.

At 2, the most-significant 32 bits of the S operand are loaded from the R-input port into the most-significant half of the S register, and the least-significant 32 bits of the S operand are loaded from the S-input port into the least-significant half of the S register.

At the same time, at 2, the output of the R-temp register is loaded into the most-significant half of the R register, and the output of the S-temp register is loaded into the least-significant half of the R register.

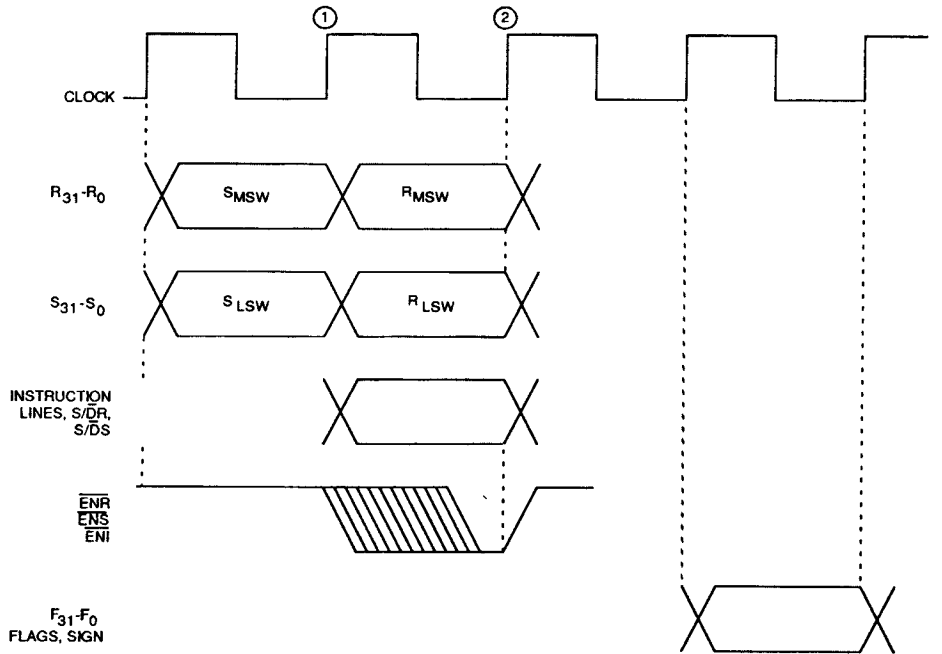
If the R-operand is a single-precision (32-bit) floating-point operand, it is input in the R-MSW position (with S/DR = HIGH) and the data in the R-LSW position is ignored by the processor. If the R-operand is a single-precision (32-bit) integer operand, it is input in the R-LSW position (with S/DR = HIGH) and the data in the R-MSW position is ignored by the processor. The same procedure applies to the S-operand, if it is a single-precision operand.

2

64-Bit Bus, Double-Cycle, S First (M16 = 1, M15 = 1, M14 = 1)

In this mode, the MSW of the 64-bit S operand is placed on the R-port and the LSW on the S-port. Both halfwords are loaded

in the first cycle. Similarly, the two halves of the R operand are loaded in the second cycle. After two cycles, the R and S registers contain the R and S operands, respectively.



WF024946

**Timing of Operations with Input Mode 8
(64-Bit Bus, Double-Cycle, S First)***

*Assumes processor flow-through mode, F register, and status register clocked.

The temporary registers are clocked on every LOW-to-HIGH clock transition.

At 1, the most-significant 32 bits of the S operand are loaded from the R-input port into the R-temp register, and the least-significant 32 bits of the S operand are loaded from the S-input port into the S-temp register.

At 2, the most-significant 32 bits of the R operand are loaded from the R-input port into the most-significant half of the R register, and the least-significant 32 bits of the R operand are loaded from the S-input port into the least-significant half of the R register.

At the same time, at 2, the output of the R-temp register is loaded into the most-significant half of the S register, and the output of the S-temp register is loaded into the least-significant half of the S register.

If the R-operand is a single-precision (32-bit) floating-point operand, it is input in the R-MSW position (with S/DR = HIGH) and the data in the R-LSW position is ignored by the processor. If the R-operand is a single-precision (32-bit) integer operand, it is input in the R-LSW position (with S/DR = HIGH) and the data in the R-MSW position is ignored by the processor. The same procedure applies to the S-operand, if it is a single-precision operand.

Pipelining of Operations

The floating-point ALU of the Am29C327 may be operated in one of three modes:

1. Flow-Through Mode
2. Single-Pipelined Mode
3. Double-Pipelined Mode

Flow-Through Mode

In this mode the floating-point ALU acts as a purely combinatorial device.

Single-Pipelined Mode

In this mode the floating-point ALU contains a single pipeline delay for all operations; throughput is roughly double that for unpipelined mode. Simplified diagrams of the ALU configuration for single-pipelined mode are shown in Figure 2.

Double-Pipelined Mode

In this mode, which applies only to the multiply-accumulate operation, the ALU contains two pipeline delays; throughput is roughly triple that for the unpipelined multiplication-accumulation operation. A simplified block diagram is shown in Figure 3.

Figures 4 and 5 provide timing diagrams for all operations except multiply-accumulate, illustrating flow-through mode and pipelined mode, respectively. Figures 6, 7, and 8 provide timing diagrams for multiply-accumulate, illustrating flow-through mode, single-pipelined mode, and double-pipelined mode, respectively.

The choice of pipelining mode affects only the floating-point ALU. Operations of other parts of the Am29C327, such as the input registers, the output register, the mode register, and the instruction register are not affected by the choice of pipelining mode. However, the instruction bits are pipelined as they pass through the ALU. This permits instructions to be interleaved in pipelined mode.

The desired pipelined mode or modes can be invoked by setting mode register bits M20 and M19 to the appropriate values.

When using the Am29C327 in either single-pipelined or double-pipelined mode, two conditions must be observed:

1. The "load mode register" instruction is not pipelined, nor are any of the mode register bits. When the mode register is loaded, any differences between the current mode and the previous mode take effect immediately. In single-pipelined mode, the user should separate the last valid ALU instruction and the "load mode register" instruction with one "NO-OP" instruction. In double-pipelined mode, the user should separate them with two "NO-OP" instructions. A NO-OP instruction is any instruction whose result is not stored in register F, or the register file.
2. A multiplication-accumulation instruction cannot be immediately followed by any other type of instruction. This problem can be avoided by inserting a "dummy" multiplication-accumulation instruction at the end of a multiplication-accumulation instruction. This "dummy" instruction is any instruction whose results are not stored in register F or the register file.

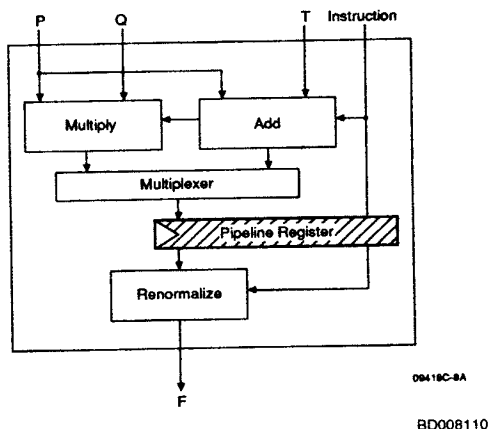


Figure 2.1. ALU in Pipelined Mode — All Operations Except Multiply-Accumulate

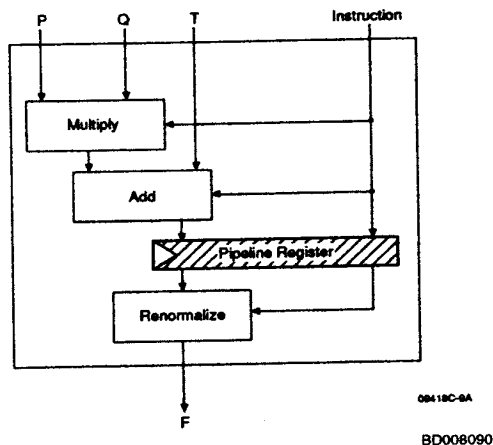


Figure 2.2. ALU in Single-Pipelined Mode — Multiply-Accumulate Only

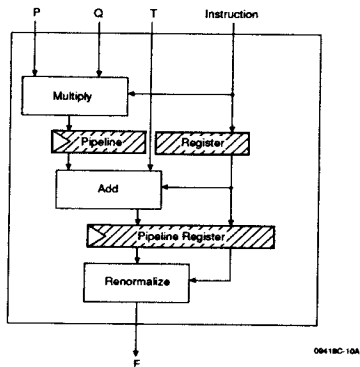


Figure 3. ALU Double-Pipelined Mode — Multiply-Accumulate Only

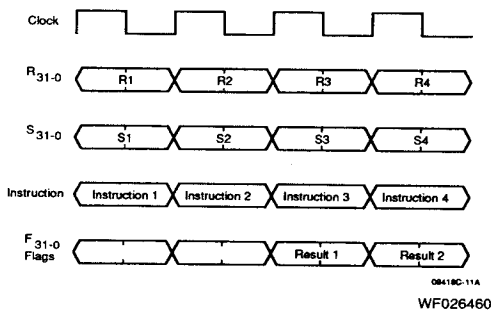


Figure 4. Flowthrough Mode — All Operations Except Multiply-Accumulate*

*Assumes F-register and status register clocked

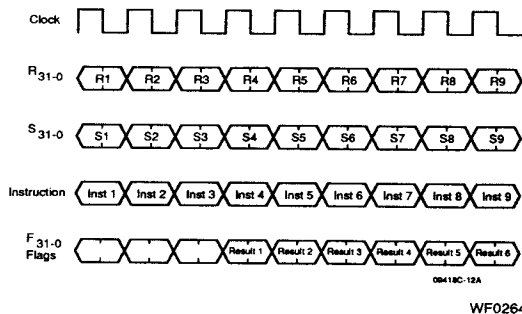


Figure 5. Pipelined Mode — All Operations Except Multiply-Accumulate*

*Assumes F-register and status register clocked

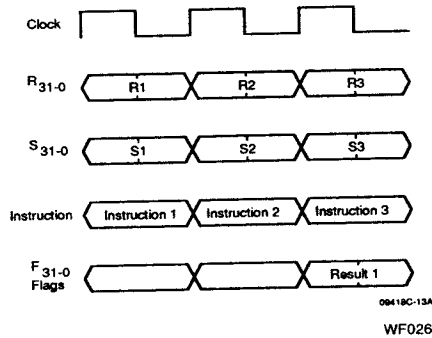


Figure 6. Flow-Through Mode — Multiply-Accumulate Operations Only*

*Assumes F-register and status register clocked.

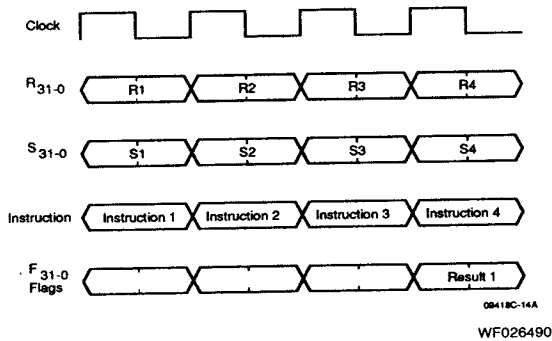


Figure 7. Single-Pipelined Mode — Multiply-Accumulate Operations Only*

*Assumes F-register and status register clocked.

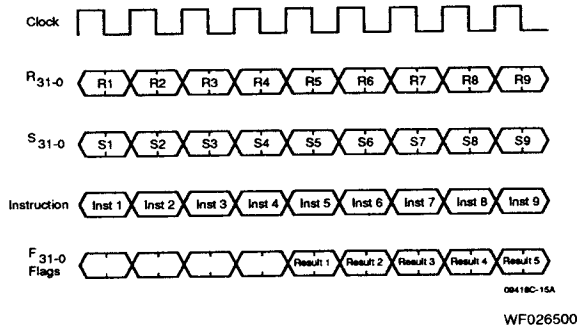


Figure 8. Double-Pipelined Mode — Multiply-Accumulate Operations Only*

*Assumes F-register and status register clocked.

Instruction Set

Instruction Register Format

The 14-bit instruction word $I_{13} - I_0$ comprises the sign-change controls, the integer/floating-point select bit, and the opcode.

I_{13}	$I_{12} I_{11}$	$I_{10} I_9$	$I_8 I_7$	I_6	I_5	$I_4 I_3 I_2 I_1 I_0$
SIGN (P)	SIGN (Q)	SIGN (T)	SIGN (F)	INT/FP		OPCODE

The opcode field, $I_4 - I_0$, specifies the base operation to be performed by the ALU; instruction bit I_5 selects between

floating-point and integer operations. The base operations and their corresponding opcodes are listed in Table 1.

TABLE 1. BASE OPERATIONS PERFORMED BY THE Am29C327

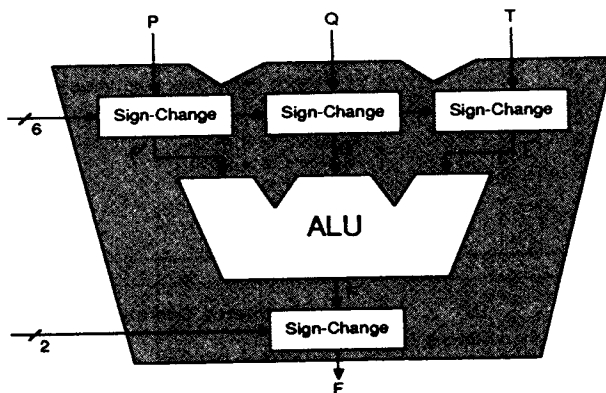
5		15-0 4 3 2 1 0	Mnemonic	Base Operation (Floating-Point)
0	0	0 0 0 0 0	FPAS	$F' = P$
0	0	0 0 0 0 1	FADD	$F' = P' + T'$
0	0	0 0 0 1 0	FMUL	$F' = P' * Q'$
0	0	0 0 0 1 1	FCMP	Compare P, T
0	0	0 0 1 0 0	FMAX	Maximum P, T
0	0	0 0 1 0 1	FMIN	Minimum P, T
0	0	0 0 1 1 0	FCTI	Convert T to Integer
0	0	0 0 1 1 1	FSTI	Scale T to Integer by Q
0	0	1 0 0 0 0	FMAC	$F' = (P' * Q') + T'$
0	0	1 0 0 0 1	FRND	Round T to Integral Value
0	0	1 0 1 0 0	FRCP	Reciprocal Seed of P
0	0	1 0 1 0 1	FCTA	Convert T to Alternate FP Format
0	0	1 1 0 0 0	FCFA	Convert T from Alternate FP Format
5		15-0 4 3 2 1 0	Mnemonic	Base Operation (Integer)
1	0	0 0 0 0 0	IPAS	$F = P$
1	0	0 0 0 0 1	IADD	$F = P + T$
1	0	0 0 0 1 0	IMUL	$F = P * Q$
1	0	0 0 0 1 1	ICMP	Compare P, T
1	0	0 1 0 0 0	IMAX	Maximum P, T
1	0	0 1 0 0 1	IMIN	Minimum P, T
1	0	0 1 1 0 0	ICTF	Convert T to Floating-Point
1	0	0 1 1 0 1	ISTF	Scale T to Floating-Point by Q
1	1	0 0 0 0 0	ILOR	$F = P \text{ OR } T$
1	1	0 0 0 0 1	IAND	$F = P \text{ AND } T$
1	1	0 0 0 1 0	IXOR	$F = P \text{ XOR } T$
1	1	0 0 0 1 1	ILSH	Shift P Logical by Q Places
1	1	0 1 0 0 0	IASH	Shift P Arithmetic by Q Places
1	1	0 1 0 0 1	IFSH	Funnel Shift PT by Q Places
5		15-0 4 3 2 1 0	Mnemonic	Base Operation (Format-Independent)
X	1	1 0 0 0 0	MOVE	Move P
X	1	1 1 1 1 1	LMRG	Load Mode Register

Sign-Change Selects

Each ALU input and output operand has associated hardware that can be used to modify operand signs (see Figure 9). These sign-change blocks, when applied to base operations, greatly increase the number of available operations. A base operation of $P + T$, for example, can be used to perform operations such as $P - T$, $ABS(P + T)$, $ABS(P) + ABS(T)$, and others, simply by modifying the signs of the input and output operands.

Using the sign-change blocks, the sign of an input operand may be left unchanged, complemented, set to zero, or set to one; the sign of the output operand may be left unchanged, complemented, set to zero, set to one, set to the sign of the P input operand, or set to the sign of the T input operand. Select decodes for the P, Q, T, and F operand sign-change blocks are shown in Table 2-1, 2-2, 2-3, and 2-4, respectively.

Note: The P-sign change block has no effect on the P input operand for the base operation $F = P$.



BD007602

Figure 9. ALU Sign-Change Blocks

TABLE 2-1. SELECT DECODE FOR P OPERAND SIGN-CHANGE BLOCK

I ₁₃	I ₁₂	Sign (P')
0	0	SIGN (P)
0	1	SIGN (P)
1	0	0
1	1	1

Note: The P-sign change block has no effect on the P input operand for the base operation $F = P$.

TABLE 2-2. SELECT DECODE FOR Q OPERAND SIGN-CHANGE BLOCK

I ₁₁	I ₁₀	Sign (Q')
0	0	SIGN (Q)
0	1	SIGN (Q)
1	0	0
1	1	1

TABLE 2-3. SELECT DECODE FOR T OPERAND SIGN-CHANGE BLOCK

I ₉	I ₈	Sign (T')
0	0	SIGN T
0	1	SIGN T
1	0	0
1	1	1

TABLE 2-4. SELECT DECODE FOR F OPERAND SIGN-CHANGE BLOCK

Base Operation	I ₁₁	I ₁₀	I ₇	I ₆	Sign (F)
$F = P$	0	x	0	0	SIGN (F')
or	0	x	0	1	SIGN (F')
Maximum P, T	0	x	1	0	0
or	0	x	1	1	1
Minimum P, T	1	0	x	x	SIGN (P)
	1	1	x	x	SIGN (T)
All other base operations	x	x	0	0	SIGN (F')
	x	x	0	1	SIGN (F')
	x	x	1	0	0
	x	x	1	1	1

Operand Multiplexer Selects

The instruction fields PSEL₃ – PSEL₀, QSEL₃ – QSEL₀, and TSEL₃ – TSEL₀ specify the select codes for the P, Q, and T

operand multiplexers, respectively; the codes are summarized in Table 3.

TABLE 3. OPERAND MULTIPLEXER SELECT CODES

PSEL ₃ QSEL ₃ TSEL ₃	PSEL ₂ QSEL ₂ TSEL ₂	PSEL ₁ QSEL ₁ TSEL ₁	PSEL ₀ QSEL ₀ TSEL ₀	P Q T
0	0	0	0	R - Register
0	0	0	1	S - Register
0	0	1	0	0
0	0	1	1	0.5 (Floating Point); -1 (Integer)
0	1	0	0	1
0	1	0	1	2
0	1	1	0	3
0	1	1	1	Pi (Floating Point); Max Neg. Value (Integer)
1	0	0	0	Register File 0 (RF0)
1	0	0	1	Register File 1 (RF1)
1	0	1	0	Register File 2 (RF2)
1	0	1	1	Register File 3 (RF3)
1	1	0	0	Register File 4 (RF4)
1	1	0	1	Register File 5 (RF5)
1	1	1	0	Register File 6 (RF6)
1	1	1	1	Register File 7 (RF7)

Operand Precisions

The Am29C327 supports mixed-precision operations, so that it is possible, for example, for an operation to have single-precision inputs and a double-precision output, or one single- and one double-precision input, or any other combination.

Precision of the operands in registers R and S is specified by signals S/DR and S/DS at the time the operands are loaded.

The precision of an operation result is specified by signal S/DF.

Operands stored in the register file are each accompanied by a bit indicating that operand's precision; this precision information is automatically supplied to the ALU when a register file location is used as an input operand to an operation.

Processor Operations

Table 4 illustrates a number of possible ALU instructions comprising the opcode, integer/floating-point select, and sign-change fields. Note that the remaining instruction bits — P, Q, and T operand multiplexer selects; the rounding modes; and the output operand precision — can be specified independently.

The user may create instructions using instruction words other than those listed in Table 4. For some base operations, sign-change control settings are completely arbitrary; for others, only the sign-change field values shown in Table 4 are valid. Table 5 summarizes permissible sign-change field values for each core operation.

TABLE 4. INSTRUCTION WORDS

Operation	Sign				I/ $\bar{F}$	Opcode
	P	Q	T	F		
FP $F = P$	xx	0x	xx	00	0	00000
FP $F = -P$	xx	0x	xx	01	0	00000
FP $F = \text{ABS}(P)$	xx	0x	xx	10	0	00000
FP $F = \text{Sign}(T) * \text{ABS}(P)$	xx	11	xx	xx	0	00000
FP $F = P + T$	00	xx	00	00	0	00001
FP $F = P - T$	00	xx	01	00	0	00001
FP $F = T - P$	01	xx	00	00	0	00001
FP $F = -P - T$	01	xx	01	00	0	00001
FP $F = \text{ABS}(P + T)$	00	xx	00	10	0	00001
FP $F = \text{ABS}(P - T)$	00	xx	01	10	0	00001
FP $F = \text{ABS}(P) + \text{ABS}(T)$	10	xx	10	00	0	00001
FP $F = \text{ABS}(P) - \text{ABS}(T)$	10	xx	11	00	0	00001
FP $F = \text{ABS}(\text{ABS}(P) - \text{ABS}(T))$	10	xx	11	10	0	00001
FP $F = P * Q$	00	00	xx	00	0	00010
FP $F = (-P) * Q$	01	00	xx	00	0	00010
FP $F = \text{ABS}(P * Q)$	00	00	xx	10	0	00010
FP Compare P, T	00	xx	01	00	0	00011
FP Max P, T	00	00	01	00	0	00100
FP Max ABS(P), ABS(T)	10	00	11	00	0	00100
FP Min P, T	01	00	00	00	0	00101
FP Min ABS(P), ABS(T)	11	00	10	00	0	00101
FP Limit P to Magnitude T	11	10	10	xx	0	00101
FP Convert T to Integer	xx	xx	00	00	0	00110
FP Scale T to Integer by Q	xx	00	00	00	0	00111
FP $F = (P * Q) + T$	00	00	00	00	0	01000
FP $F = T - (P * Q)$	01	00	00	00	0	01000
FP $F = (P * Q) - T$	00	00	01	00	0	01000
FP $F = -T - P * Q$	01	00	01	00	0	01000
FP $F = \text{ABS}(T) + \text{ABS}(P * Q)$	10	10	10	00	0	01000
FP $F = \text{ABS}(T) - \text{ABS}(P * Q)$	11	10	10	00	0	01000
FP $F = \text{ABS}(P * Q) - \text{ABS}(T)$	10	10	11	00	0	01000
FP Round T to Integral Value	xx	xx	00	00	0	01001
FP Reciprocal Seed (P)	00	xx	xx	00	0	01010
FP Convert T to Alternate Floating-point Format	xx	xx	00	00	0	01011
FP Convert T from Alternate Floating-point Format	xx	xx	00	00	0	01100
Int $F = P$	00	00	00	00	1	00000
Int $F = -P$	00	00	00	01	1	00000
Int $F = \text{ABS}(P)$	00	00	00	10	1	00000
Int $F = \text{sign}(T) * \text{ABS}(P)$	00	11	00	xx	1	00000
Int $F = P + T$	00	xx	00	00	1	00001
Int $F = P - T$	00	xx	01	00	1	00001
Int $F = T - P$	01	xx	00	00	1	00001
Int $F = \text{ABS}(P + T)$	00	xx	00	10	1	00001
Int $F = \text{ABS}(P - T)$	00	xx	01	10	1	00001
Int $F = P * Q$	00	00	xx	00	1	00010
Int Compare P, T	00	xx	01	00	1	00011
Int Max P, T	00	00	01	00	1	00100
Int Min P, T	01	00	00	00	1	00101

TABLE 4. INSTRUCTION WORDS (Cont'd.)

Operation	Sign				I/F	Opcode
	P	Q	T	F		
Int Convert T to Float	xx	xx	00	00	1	00110
Int Scale T to Float by Q	xx	00	00	00	1	00111
Int P OR T	xx	xx	xx	xx	1	10000
Int P AND T	xx	xx	xx	xx	1	10001
Int P XOR T	xx	xx	xx	xx	1	10010
Int NOT T (see Note 1)	xx	xx	xx	xx	1	10010
Int Shift P Logical by Q Places	00	00	xx	00	1	10011
Int Shift P Arithmetic by Q Places	00	00	xx	00	1	10100
Int Funnel Shift PT by Q Places	00	00	00	00	1	10101
Move P	xx	xx	xx	xx	x	11000
Load Mode Register	xx	xx	xx	xx	x	11111

Notes: 1. NOT T is performed by XORING T with a word containing all 1's (integer - 1). When invoking NOT T the user must set PSEL₃-PSEL₀ to 0011₂, thus selecting integer constant - 1.

TABLE 5. ALLOWABLE SIGN-CHANGE/BASE OPERATION COMBINATIONS

	I 11111 5 43210	Base Operation	Sign-Change Fields			
			Sign (P)	Sign (Q)	Sign (T)	Sign (F)
Floating-Point Operations	0 00000	FP F' = P*	V	V	x	V
	0 00001	FP F' = P' + T'	V	x	V	V
	0 00010	FP F' = P'*Q'	V	V	x	V
	0 00011	FP Compare P, T	F	x	F	F
	0 00100	FP Max P, T	F	F	F	F
	0 00101	FP Min P, T	F	F	F	F
	0 00110	FP Cvt T to Int	x	x	F	F
	0 00111	FP Scale T to Int	x	F	F	F
	0 01000	FP F' = (P'*Q') + T'	V	V	V	V
	0 01001	FP Round T	x	x	F	F
	0 01010	FP Recip Seed P	F	x	x	F
	0 01011	FP Cvt T to Alt Fmt	x	x	F	F
	0 01100	FP Cvt T fm Alt Fmt	x	x	F	F
Integer Operations	1 00000	Int F = P	F	F	F	F
	1 00001	Int F = P + T	F	x	F	F
	1 00010	Int F = P*Q	F	F	x	F
	1 00011	Int Compare P, T	F	x	F	F
	1 00100	Int Max P, T	F	F	F	F
	1 00101	Int Min P, T	F	F	F	F
	1 00110	Int Cvt T to f.p.	x	x	F	F
	1 00111	Int Scale T to f.p.	x	F	F	F
	1 10000	Int F = P OR T	x	x	x	x
	1 10001	Int F = P AND T	x	x	x	x
	1 10010	Int F = P XOR T	x	x	x	x
	1 10011	Int Shift P Logical	F	F	x	F
	1 10100	Int Shift P Arith	F	F	x	F
	1 10101	Int Funnel Shift PT	F	F	F	F
	x 11000	Move P	x	x	x	x
	x 11111	Load Mode Reg	x	x	x	x

Key: V = Variable; user can specify arbitrary sign change.

F = Fixed; user is restricted to sign change combinations shown in Table 4.

x = Don't care; this field does not affect the operation or its result.

*Note: The P-sign change block has no effect on the P-input operand for the base operation F' = P.

Base Operation Code Description

F' = P (Floating-point) FPAS: The P-operand is passed through the ALU unchanged, except for any specified precision conversions. If the user specifies different input and output precisions, the operation may be used to perform single-to-double or double-to-single conversions. Instructions such as negation, absolute value extraction and sign-transfer may be executed by setting the sign-change controls appropriately while executing this base operation.

*Note: The P-sign change block has no effect on the P-input operand for the base operation F' = P.

F' = P' + T' (Floating-point) FADD: The two operands P' and T' are added, taking into account any specified precision conversions. Instructions such as subtraction, sum-of-absolute-values, difference-of-absolute-values, absolute-value-of-sum, and absolute-value-of-difference may be executed by setting the sign-change controls appropriately while executing this base operation.

F' = P' * Q' (Floating-point) FMUL: The Operands P' and Q' are multiplied, taking into account any specified precision conversions. Instructions such as negative-product and absolute-value-of-product may be executed by setting the sign-change controls appropriately while executing this base operation.

Compare P, T (Floating-point) FCMP: The two operands P and T are compared, taking into account any specified precision conversions. The output of the operation is the result of the subtraction (P-T). The flags are set appropriately to indicate the result of the comparison, conforming to the relevant parts of the floating-point standards. For IEEE and DEC operations, one of four flags (greater than, less than, equals or unordered) is set for any given compare operation. For IBM operations, the unordered flag does not apply since the format does not support any reserved operands.

Maximum P, T (Floating-point) FMAX: The two operands P and T are compared, taking into account any specified precision conversions. The most-positive operand is selected as the output. The "Winner" flag indicates which of the operands is selected. Additionally, the operation maximum-of-absolute-value may be performed by setting the appropriate sign-change controls.

Minimum P, T (Floating-point) FMIN: The two operands P and T are compared, taking into account any specified precision conversions. The most-negative operand is selected as the output. The "Winner" flag indicates which of the two operands is selected. Additionally, the operations minimum-of-absolute-values and limit-P-to-magnitude-T may be performed by setting the appropriate sign-change controls. The limit-P-to-magnitude-T operation is useful for "clipping" a sequence of operands to ensure that their magnitude never exceeds a preset limit.

Convert T to Integer (Floating-point) FCTI: The operand T is converted from floating-point representation to two's complement integer representation, taking into account the specified precision of the floating-point operand. If the output precision is specified as single, the result is a 32-bit integer. If the output precision is specified as double, the result is a 64-bit integer.

Scale T to Integer by Q (Floating-point) FSTI: The operand T is converted from floating-point representation to two's complement integer representation, using the exponent of the floating-point operand Q as a scale factor and taking into account the specified precision of the floating-point operands. The unbiased exponent of the operand Q is added to the exponent of the operand T, permitting IEEE and DEC oper-

ands to be multiplied by any power of 2 and IBM operands by any power of 16, before the conversion is performed. If the output precision is specified as single, the result is a 32-bit integer. If the output precision is specified as double, the result is a 64-bit integer.

F' = (P' * Q') + T' (Floating-point) FMAC: The operands P' and Q' are multiplied, producing a double-precision product. This product is added to the operand T', taking into account any specified precision conversions. Instructions such as "P*Q = T'", "T = P*Q'", "ABS(P*Q) + ABS(T)" and "ABS (P*Q + T)" may be executed by setting the sign-change controls appropriately while executing this base operation.

Round T to Integral Value (Floating-point) FRND: The floating-point operand T is rounded to an integer-valued floating-point operand, using the specified rounding mode and taking into account any specified precision conversions. As an example, the operation converts a floating-point representation of Pi (3.14159...) to a floating-point representation of 3.0 or 4.0, depending on the rounding mode selected. The final result of the operation is a floating-point number.

Reciprocal Seed of P (Floating-point) FRCP: An approximation to the reciprocal of the operand P is evaluated, taking into account any specified precision conversions. The reciprocal seed comprises an accurate sign, a fully-accurate exponent and a mantissa which is accurate to only one place. This operation can be used as the initial step in performing Newton-Raphson division; optionally, an external seed look-up table can be used for faster convergence.

Convert T to Alternate Floating-point Format (Floating-point) FCTA: The floating-point operand T, assumed to be in the "primary" floating-point format is converted to a floating-point operand in the "Alternate" floating-point format, taking into account any specified precision conversions.

Convert T from Alternate Floating-point Format (Floating-point) FCFA: The floating-point operand T, assumed to be in the "Alternate" floating-point format is converted to a floating-point operand in the "primary" floating-point format, taking into account any specified precision conversions.

F = P (Integer) IPAS: The P-operand is passed through the ALU unchanged, except for any specified precision conversions. If the user specifies different input and output precisions, the operation may be used to perform single-to-double or double-to-single conversions. Instructions such as negation, absolute value extraction and sign transfer may be performed by setting the sign-change control appropriately while executing this base operation.

F = P + T (Integer) IADD: The two operands P and T are added, taking into account any specified precision conversions. Instructions such as subtraction, absolute-value-of-sum and absolute-value-of-difference may be performed by setting the sign-change controls appropriately while executing this base operation.

F = P * Q (Integer) IMUL: The two operands P and Q are multiplied, taking into account any specified precision conversions. Either 32-bit multiplication or 64-bit multiplication may be performed, and the user may select either the MSBs or the LSBs of the product as the final result. In addition, format-adjusting may be implemented if required, and the operands may be considered as signed (two's complement) or unsigned.

Compare P, T (Integer) ICMP: The two operands P and T are compared, taking into account any specified precision conversions. The output of the operation is the result of the subtraction (P - T). The flags are set appropriately to indicate the result of the comparison, one of three flags (greater than,

less than or equals) being set for any given compare operation.

Maximum P, T (Integer) IMAX: The two operands P and T are compared, taking into account any specified precision conversions. The most-positive operand is selected as the output. The "Winner" flag indicates which of the two operands is selected.

Minimum P, T (Integer) IMIN: The two operands P and T are compared, taking into account any specified precision conversions. The most-negative operand is selected as the output. The "Winner" flag indicates which of the two operands is selected.

Convert T to Floating-point (Integer) ICTF: The operand T is converted from two's complement integer representation to floating-point representation, taking into account the specified precision of the integer operand. If the output precision is specified as single, the result is a 32-bit floating-point operand. If the output precision is specified as double, the result is a 64-bit floating-point operand.

Scale T to Floating-point by Q (Integer) ISTF: The operand T is converted from two's complement integer representation to floating-point representation, using the exponent of the floating-point operand Q as a scale factor and taking into account the specified precision of the integer operand. The unbiased exponent of the operand Q is added to the exponent of the floating-point result, permitting IEEE and DEC operands to be multiplied by any power of 2, and IBM operands by any power of 16, after the conversion is performed. If the output precision is specified as single, the result is a 32-bit floating-point operand. If the output precision is specified as double, the result is a 64-bit floating-point operand.

F = P OR T (Integer) ILOR: The operand P is logically ORed with the operand T. Before the operation is performed, the inputs, if 32-bit, are sign-extended to 64-bits.

F = P AND T (Integer) IAND: The operand P is logically ANDed with the operand T. Before the operation is performed, the inputs, if 32-bit, are sign-extended to 64-bits.

F = P XOR T (Integer) IXOR: The operand P is logically exclusive-ORed with the operand T. Before the operation is performed, the inputs, if 32-bit, are sign-extended to 64-bits. This operation may be used to invert an operand by selecting the second operand to be the integer constant -1, so that all bits of this second operand are 1. Exclusive-ORing an operand with -1 is equivalent to inverting each bit in the operand.

Shift P Logical by Q Places (Integer) ILSH: This operation cannot be performed in mixed-precision mode. The precision of the result is the same as the precision of the input operand P. A two's-complement shift length in the range -64 to +63 (double-precision) or -32 to +31 (single-precision) is extracted from the LSBs of the operand Q. The operand P is logically right-shifted by the number of places specified by the shift length. A negative shift length therefore produces a left-shift. If a right-shift is performed, zeros fill vacated bit positions to the left of the input operand. If a left-shift is performed, zeros fill vacated bit positions to the right of the input operand.

Shift P Arithmetic by Q Places (Integer) IASH: This operation cannot be performed in mixed-precision mode. The precision of the result is the same as the precision of the input

operand P. A two's-complement shift length in the range -64 to +63 (double-precision) or -32 to +31 (single-precision) is extracted from the LSBs of the operand Q. The operand P is arithmetically right-shifted by the number of places specified by the shift length. A negative shift length therefore produces a left-shift. If a right-shift is performed, the MSB (bit 63 or 31) is replicated to fill vacated bit positions to the left of the input operand. If a left-shift is performed, zeros fill vacated bit positions to the right of the input operand.

Funnel Shift PT by Q Places (Integer) IFSH: This operation cannot be performed in mixed-precision mode. The operand T is interpreted as having the same precision as the input operand P and the precision of the result is also the same as the precision of the input operand P. A two's-complement shift length in the range -64 to +63 (double-precision) or -32 to +31 (single-precision) is extracted from the LSBs of the operand Q. A triple-width operand (96-bit or 192-bit) is formed by concatenating the input operands into the arrangement P-T-P, with the 32-bit or 64-bit result field initially aligned with the T-operand. The triple-width operand is logically right-shifted by the number of places specified by the shift length. A negative shift length therefore produces a left-shift.

Move P (Format Independent) MOVE: The 64-bit operand P is passed unchanged through the ALU. No exceptions are detected or signalled.

Load Mode Register (Format Independent) LMRG: The 32-bit operand on the R31.0 input port is loaded into the mode register on the rising edge of the clock input. No exceptions are detected or signalled.

Operation Flags

For each operation, the ALU produces thirteen flags that indicate operation status. Of the flags produced, a maximum of seven are relevant to any given operation. The relevant flags are clocked into the status register, and the other flags are discarded.

The ALU flags are:

C — CARRY: Carry-out bit produced by integer addition, subtraction, or comparison.

I — INVALID OPERATION: Input operands are unsuitable for the operation specified (e.g., $\infty * 0$).

R — RESERVED OPERAND: Reserved operand detected/generated.

S — SIGN: Result sign.

U — UNDERFLOW: Result underflowed the destination format.

V — OVERFLOW: Result overflowed the destination format.

W — WINNER: Indicates which of the two operands was selected when performing Max/Min operations.

X — INEXACT RESULT: Result had to be rounded to fit the destination format.

Z — ZERO: Zero result.

>, =, <, # — GREATER THAN, EQUAL, LESS THAN, UNORDERED: Used to report the result of a comparison operation.

Table 6 lists the flags reported for each operation.

TABLE 6. ORGANIZATION OF FLAGS APPLICABLE TO EACH BASE OPERATION

Format	Operation	Mnemonic	SIGN	Flag6	Flag5	Flag4	Flag3	Flag2	Flag1
IEEE	$F' = P$	FPAS	S	Z	X	U	V	R	I
IEEE	$F' = P' + T'$	FADD	S	Z	X	U	V	R	I
IEEE	$F' = P' \times Q'$	FMUL	S	Z	X	U	V	R	I
IEEE	Compare P, T	FCMP	S	=	>	<	#	R	I
IEEE	Maximum P, T	FMAX	S	Z		W		R	I
IEEE	Minimum P, T	FMIN	S	Z		W		R	I
IEEE	Convert T to Integer	FCTI	S	Z	X		V	R	I
IEEE	Scale T to Integer	FSTI	S	Z	X		V	R	I
IEEE	$F' = (P' \times Q') + T'$	FMAC	S	Z		U	V	R	I
IEEE	Round T to Integral Value	FRND	S	Z	X		V	R	I
IEEE	Reciprocal Seed of P	FRCP	S	Z		U	V	R	I
IEEE	Convert T to Alt FP Format	FCTA	S	Z	X	U	V	R	I
IEEE	Convert T from Alt FP Format	FCFA	S	Z	X	U	V	R	I
DEC D	$F' = P$	FPAS	S	Z	X		V	R	
DEC D	$F' = P' + T'$	FADD	S	Z	X	U	V	R	
DEC D	$F' = P' \times Q'$	FMUL	S	Z	X	U	V	R	
DEC D	Compare P, T	FCMP	S	=	>	<	#	R	
DEC D	Maximum P, T	FMAX	S	Z		W		R	
DEC D	Minimum P, T	FMIN	S	Z		W		R	
DEC D	Convert T to Integer	FCTI	S	Z	X		V	R	I
DEC D	Scale T to Integer	FSTI	S	Z	X		V	R	I
DEC D	$F' = (P' \times Q') + T'$	FMAC	S	Z		U	V	R	
DEC D	Round T to Integral Value	FRND	S	Z	X		V	R	
DEC D	Reciprocal Seed of P	FRCP	S	Z		U	V	R	I
DEC D	Convert T to Alt FP Format	FCTA	S	Z	X	U	V	R	I
DEC D	Convert T from Alt FP Format	FCFA	S	Z	X	U	V	R	I
DEC G	$F' = P$	FPAS	S	Z	X	U	V	R	
DEC G	$F' = P' + T'$	FADD	S	Z	X	U	V	R	
DEC G	$F' = P' \times Q'$	FMUL	S	Z	X	U	V	R	
DEC G	Compare P, T	FCMP	S	=	>	<	#	R	
DEC G	Maximum P, T	FMAX	S	Z		W		R	
DEC G	Minimum P, T	FMIN	S	Z		W		R	
DEC G	Convert T to Integer	FCTI	S	Z	X		V	R	I
DEC G	Scale T to Integer	FSTI	S	Z	X		V	R	I
DEC G	$F' = (P' \times Q') + T'$	FMAC	S	Z		U	V	R	
DEC G	Round T to Integral Value	FRND	S	Z	X		V	R	
DEC G	Reciprocal Seed of P	FRCP	S	Z		U	V	R	I
DEC G	Convert T to Alt FP Format	FCTA	S	Z	X	U	V	R	I
DEC G	Convert T from Alt FP Format	FCFA	S	Z	X	U	V	R	I
IBM	$F' = P$	FPAS	S	Z	X		V		
IBM	$F' = P' + T'$	FADD	S	Z	X	U	V		
IBM	$F' = P' \times Q'$	FMUL	S	Z	X	U	V		
IBM	Compare P, T	FCMP	S	=	>	<			
IBM	Maximum P, T	FMAX	S	Z		W			
IBM	Minimum P, T	FMIN	S	Z		W			
IBM	Convert T to Integer	FCTI	S	Z	X		V		I
IBM	Scale T to Integer	FSTI	S	Z	X		V		I
IBM	$F' = (P' \times Q') + T'$	FMAC	S	Z		U	V		
IBM	Round T to Integral Value	FRND	S	Z	X		V		
IBM	Reciprocal Seed of P	FRCP	S	Z			V		I
IBM	Convert T to Alt FP Format	FCTA	S	Z	X	U	V	R	
IBM	Convert T from Alt FP Format	FCFA	S	Z	X	U	V	R	I
Integer	$F = P$	IPAS	S	Z			V		C
Integer	$F = P + T$	IADD	S	Z			V		
Integer	$F = P \times Q$	IMUL	S	Z			V		C
Integer	Compare P, T	ICMP	S	=	>	<			
Integer	Maximum P, T	IMAX	S	Z		W			
Integer	Minimum P, T	IMIN	S	Z		W			
Integer	Convert T to Floating-Point	ICTF	S	Z	X				
Integer	Scale T to Floating-Point	ISTF	S	Z	X	U	V	R	
Integer	$F = P \text{ OR } T$	ILOR	S	Z					
Integer	$F = P \text{ AND } T$	IAND	S	Z					
Integer	$F = P \text{ XOR } T$	IXOR	S	Z					
Integer	Logical Shift P by Q Places	ILSH	S	Z					
Integer	Arithmetic Shift P by Q Places	IASH	S	Z			V		
Integer	Funnel Shift P T by Q Places	IFSH	S	Z					
	Move P	MOVE	S						
	Load Mode Register	LMRG							

Master/Slave Operation

Two Am29C327 processors can be tied together in master/slave configuration, with the slave checking the results produced by the master. All input and output signals of the slave, with the exception of SLAVE and MSEERR, are tied to the corresponding signals of the master. The master is selected by asserting signal SLAVE LOW; the slave, by asserting signal SLAVE HIGH.

The slave processor, by comparing its outputs to the outputs of the master processor, performs a comprehensive check of the operation of the master processor. In addition, the slave processor may detect open circuits and other faults in the electrical path between the master processor and the system. Note that the master processor still performs the comparison between its outputs and its own internally generated results, and is therefore able to detect faults in its output drivers.

APPLICATIONS

Suggestions for Power and Ground Pin Connections

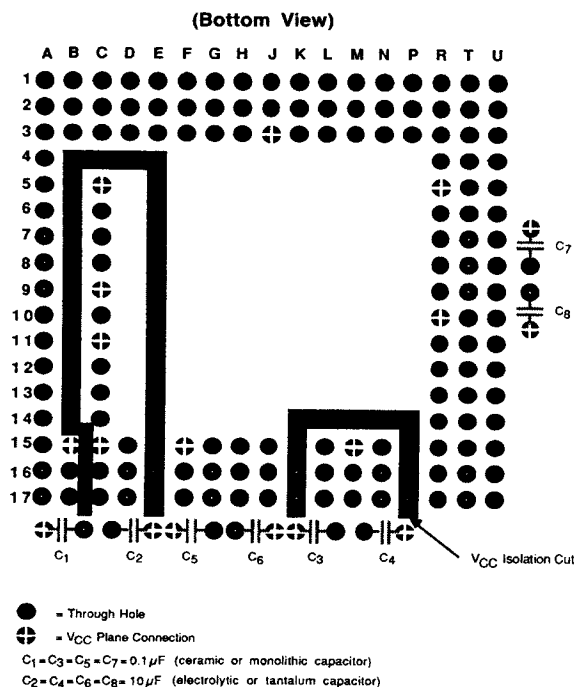
The Am29C327 operates in an environment of fast signal rise times and substantial switching currents. Therefore, care must be exercised during circuit board design and layout, as with any high-performance component. The following is a suggested layout, but since systems vary widely in electrical configuration, an empirical evaluation of the intended layout is recommended.

The V_{CCO} and G_{NDO} pins carry output driver switching currents and can be electrically noisy. The V_{CC} and G_{ND} pins, which supply the logic core of the device, tend to produce less noise, and the circuits they supply may be adversely affected by noise spikes on the V_{CC} plane. For this reason, it is best to provide isolation between the V_{CC} and V_{CCO} pins, as well as independent decoupling for each. Isolating the G_{ND} and G_{NDO} pins is not required.

Printed Circuit-Board Layout Suggestions

1. Use of a multi-layer PC board with separate power, ground, and signal planes is highly recommended.
2. All V_{CC} and V_{CCO} pins should be connected to the V_{CC} plane. V_{CC} pins should be isolated from V_{CCO} pins by means of an isolation slot which is cut in the V_{CC} plane; see Figure 10. By physically separating the V_{CC} and V_{CCO} pins, coupled noise will be reduced.
3. All G_{ND} and G_{NDO} pins should be connected directly to the ground plane.
4. The V_{CCO} pins should be decoupled to ground with a 0.1- μ F ceramic capacitor and a 10- μ F electrolytic capacitor, placed as closely to the Am29C327 as is practical. V_{CC} pins should be decoupled to ground in a similar manner.

A suggested layout is shown in Figure 10.



CD011712

Figure 10. Suggested Printed Circuit-Board Layout

ABSOLUTE MAXIMUM RATINGS

Storage Temperature -65 to +150°C
Case Temperature Under Bias -55 to +125°C
Supply Voltage to
Ground Potential Continuous -0.3 V to +7.0 V
DC Voltage Applied to Outputs for
HIGH Output State -0.3 V to +V_{CC} + 0.3 V
DC Input Voltage -0.3 V to +V_{CC} + 0.3 V
DC Output Current, Into LOW Outputs 30 mA
DC Input Current -10 mA to +10 mA

Stresses above those listed under ABSOLUTE MAXIMUM RATINGS may cause permanent device failure. Functionality at or above these limits is not implied. Exposure to absolute maximum ratings for extended periods may affect device reliability.

OPERATING RANGES

Commercial (C) Devices

Case Temperature (T_C) 0 to +70°C
Supply Voltage (V_{CC}) +4.75 V to +5.25 V

Military* (M) Devices

Case Temperature (T_C) -55 to +125°C
Supply Voltage (V_{CC}) +4.5 V to +5.5 V

Operating ranges define those limits between which the functionality of the device is guaranteed.

*Military Product 100% tested at T_C = +25°C, +125°C, and -55°C.

Thermal Resistance (Typical)

Symbol	CGX169	Unit
θ _{JA}	20	°C/W
θ _{JC}	4	°C/W

DC CHARACTERISTICS over operating ranges unless otherwise specified (for APL Products, Group A, Subgroups 1, 2, 3 are tested unless otherwise noted)

2

Parameter Symbol	Parameter Description	Test Conditions (Note 1)		Min.	Max.	Unit	
V _{OH}	Output HIGH Voltage	V _{CC} = Min. V _{IN} = V _{IH} or V _{IL}	I _{OH} = -4.0 mA	2.4		V	
V _{OL}	Output LOW Voltage	V _{CC} = Min. V _{IN} = V _{IH} or V _{IL}	I _{OL} = 4.0 mA		0.5	V	
V _{IH}	Input Logical HIGH Voltage (Note 2)			2.0		V	
V _{IL}	Input Logical LOW Voltage (Note 2)				0.8	V	
V _{IH} (F)	Guaranteed Input Logical HIGH Voltage (Note 2)	F Bus, Slave Operation only		V _{CC} - 0.5		V	
V _{IL} (F)	Guaranteed Input Logical LOW Voltage (Note 2)	F Bus, Slave Operation only			0.5	V	
I _{IL}	Input LOW Current	V _{CC} = Max. V _{IN} = 0.5 V			-10	μA	
I _{IH}	Input HIGH Current	V _{CC} = Max. V _{IN} = V _{CC} - 0.5 V			10	μA	
I _{OZH}	Off State (High Impedance) Output Current	V _{CC} = Max. V _O = 2.4 V			10	μA	
I _{OZL}	Off State (High Impedance) Output Current	V _{CC} = Max. V _O = 0.5 V			-10	μA	
I _{CC} Static	Static Power Supply Current	V _{CC} = Max. I _O = 0 μA	COM'L T _C = 0 to +70°C	(Note 3) CMOS V _{IN} = V _{CC} or GND		240	mA
				(Note 3) TTL V _{IN} = 0.5 V or 2.4 V		275	
			V _{IL} = 0.5 V to +125°C	(Note 3) CMOS V _{IN} = V _{CC} or GND		285	
				(Note 3) TTL V _{IN} = 0.5 V or 2.4 V		335	
C _{PD}	Power Dissipation Capacitance (Note 4)	V _{CC} = Max. No Load			12,500	pF	

Notes: 1. V_{CC} conditions shown as Min. or Max. refer to the applicable device type Operating Range.
2. These input levels provide zero-noise immunity and should only be statically tested in a noise-free environment (not functionally tested).
3. Use CMOS Static I_{CC} when the device is driven by CMOS circuits and TTL Static I_{CC} when the device is driven by TTL circuits.
4. C_{PD} determines the dynamic current consumption:

$$I_{CC}(\text{Total}) = I_{CC}(\text{Static}) + (C_{PD} + nCL) \frac{f}{2}, \text{ where } f \text{ is the clock frequency, } CL \text{ output load capacitance, and } n \text{ number of loads.}$$

This is tested on a sample basis.

CAPACITANCE*

Parameter Symbol	Parameter Description	Test Conditions	Min.	Max.	Units
C _I	Input Capacitance	f = 1 MHz		12	pF
C _O	Output Capacitance			20	
C _{I/O}	I/O Pin Capacitance			20	

*These capacitances are tested on a sample basis.

SWITCHING CHARACTERISTICS over **COMMERCIAL** operating operating range unless otherwise specified

No.	Parameter Description	Test Conditions	Am29C327		Am29C327-1		Am29C327-2		Unit
			Min.	Max.	Min.	Max.	Min.	Max.	
1	CLOCK Period Flow-Through Mode Multiply-Accumulate All Other Operations Single-Pipelined Mode Multiply-Accumulate All Other Operations Double-Pipelined Mode Multiply-Accumulate	(Note 1)	240 190	DC DC	200 160	DC DC	160 130	DC DC	ns ns
2	CLOCK LOW Time		15		12		10		ns
3	CLOCK HIGH Time		15		12		10		ns
4	Instruction Setup Time	(Note 2)	19		17		16		ns
5	Instruction Hold Time	(Note 2)	0		0		0		ns
6	Data Setup Time	(Note 3)	19		17		16		ns
7	Data Hold Time	(Note 3)	0		0		0		ns
8	Control Lines Setup Time	(Note 4)	18		17		16		ns
9	Control Lines Hold Time	(Note 4)	0		0		0		ns
10	F ₃₁₋₀ CLOCK-to-Output-Valid F Register Clocked			22		18		15	ns
11	FLAG ₆₋₁ SIGN CLOCK-to-Output-Valid Status Register Clocked			17		15		13	ns
12	F ₃₁₋₀ CLOCK-to-Output-Valid F Register Transparent Flow-Through Mode Multiply-Accumulate All Other Operations Single-Pipelined Mode Multiply-Accumulate All Other Operations Double-Pipelined Mode Multiply-Accumulate			250 200 190 135 135		175 170 170 110 110		172 136 136 88 88	ns ns ns ns ns
13	FLAG ₆₋₁ SIGN CLOCK-to-Output-Valid Status Register Transparent Flow-Through Mode Multiply-Accumulate All Other Operations Single-Pipelined Mode Multiply-Accumulate All Other Operations Double-Pipelined Mode Multiply-Accumulate			250 195 190 115 115		170 170 160 110 100		168 136 128 80 80	ns ns ns ns ns
14	OE _F , OE _S , Disable Time HIGH to Z			18		16		14	ns
15	OE _F , OE _S , Disable Time LOW to Z			18		16		14	ns
16	OE _F , OE _S , Enable Time Z to HIGH			22		19		16	ns
17	OE _F , OE _S , Enable Time Z to LOW			22		19		16	ns
18	FSEL to F ₃₁₋₀			22		18		15	ns
19	MSERR Data-to-Valid Delay			29		27		25	ns

Notes: 1. CLOCK switching characteristics are made relative to 1.5 V.

2. Instruction signals include S/DR, S/D_S, S/D_F, RM₂₋₀, PSEL₃₋₀, QSEL₃₋₀, TSEL₃₋₀, and I₁₃₋₀.

3. Data signals include R₃₁₋₀ and S₃₁₋₀.

4. Control signals include ENR, ENS, ENF, ENRF, RFSEL₂₋₀, and ENI.

Conditions: A. All inputs/outputs except CLOCK are TTL-compatible for V_{IH}, V_{IL}, and V_{OL}.

B. All outputs are driving 80 pF unless otherwise noted.

C. All setup, hold, and delay times are measured relative to CLOCK at V_{CC}/2 volts unless otherwise noted.

Advance Information

PRELIMINARY

SWITCHING CHARACTERISTICS over **MILITARY** operating range unless otherwise specified

No.	Parameter Description	Test Conditions	Am29C327		Am29C327-1		Am29C327-2		Unit
			Min.	Max.	Min.	Max.	Min.	Max.	
1	CLOCK Period Flow-Through Mode Multiply-Accumulate All Other Operations Single-Pipelined Mode Multiply-Accumulate All Other Operations Double-Pipelined Mode Multiply-Accumulate	(Note 1)	290 230	DC DC	245 195	DC DC	200 160	DC DC	ns ns
2	CLOCK LOW Time		18		15		12		ns
3	CLOCK HIGH Time		18		15		12		ns
4	Instruction Setup Time	(Note 2)	23		21		17		ns
5	Instruction Hold Time	(Note 2)	0		0		0		ns
6	Data Setup Time	(Note 3)	23				17		ns
7	Data Hold Time	(Note 3)	0				0		ns
8	Control Lines Setup Time	(Note 4)	22		21		17		ns
9	Control Lines Hold Time	(Note 4)	0		0		0		ns
10	F ₃₁₋₀ CLOCK-to-Output-Valid F Register Clock					23		19	ns
11	FLAG ₆₋₁ SIGN CLOCK-to-Output-Valid Status Register Clock					19		16	ns
12	F ₃₁₋₀ CLOCK-to-Output-Valid F Register Transparent Flow-Through Mode Multiply-Accumulate All Other Operations Single-Pipelined Mode Multiply-Accumulate All Other Operations Double-Pipelined Mode Multiply-Accumulate			300 240		260 205		215 170	ns ns
				230 165		205 135		170 110	ns ns
				165		135		110	ns
13	FLAG ₆₋₁ SIGN CLOCK-to-Output-Valid Status Register Transparent Flow-Through Mode Multiply-Accumulate All Other Operations Single-Pipelined Mode Multiply-Accumulate All Other Operations Double-Pipelined Mode Multiply-Accumulate			300 235		255 205		210 170	ns ns
				230 140		195 125		160 100	ns ns
				140		125		100	ns
14	OEF, OES, Disable Time HIGH to Z			22		20		16	ns
15	OEF, OES, Disable Time LOW to Z			22		20		16	ns
16	OEF, OES, Enable Time Z to HIGH			27		23		19	ns
17	OEF, OES, Enable Time Z to LOW			27		23		19	ns
18	FSEL to F ₃₁₋₀			27		22		18	ns
19	MSERR Data-to-Valid Delay			34		33		27	ns

Notes: 1. CLOCK switching characteristics are made relative to 1.5 V.

2. Instruction signals include S/DR, S/DS, S/DF, RM₂₋₀, PSEL₃₋₀, QSEL₃₋₀, TSEL₃₋₀, and I₁₃₋₀.

3. Data signals include R₃₁₋₀ and S₃₁₋₀.

4. Control signals include ENR, ENS, ENF, ENRF, RFSEL₂₋₀, and ENI.

Conditions: A. All inputs/outputs except CLOCK are TTL-compatible for V_{IH}, V_{IL}, and V_{OL}.

B. All outputs are driving 80 pF unless otherwise noted.

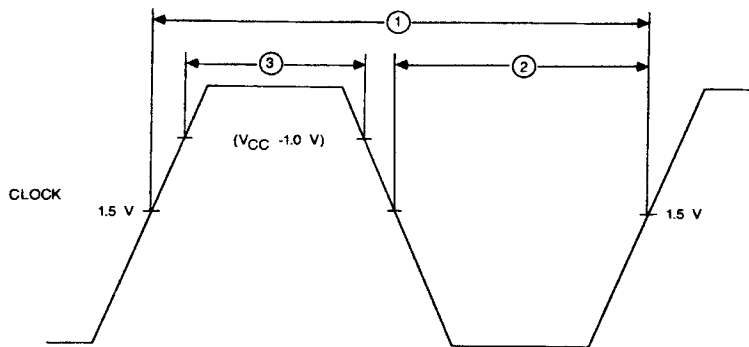
C. All setup, hold, and delay times are measured relative to CLOCK at V_{CC}/2 volts unless otherwise noted.

SWITCHING WAVEFORMS

KEY TO SWITCHING WAVEFORMS

WAVEFORM	INPUTS	OUTPUTS
	MUST BE STEADY	WILL BE STEADY
	MAY CHANGE FROM H TO L	WILL BE CHANGING FROM H TO L
	MAY CHANGE FROM L TO H	WILL BE CHANGING FROM L TO H
	DON'T CARE: ANY CHANGE PERMITTED	CHANGING: STATE UNKNOWN
	DOES NOT APPLY	CENTER LINE IS HIGH IMPEDANCE "OFF" STATE

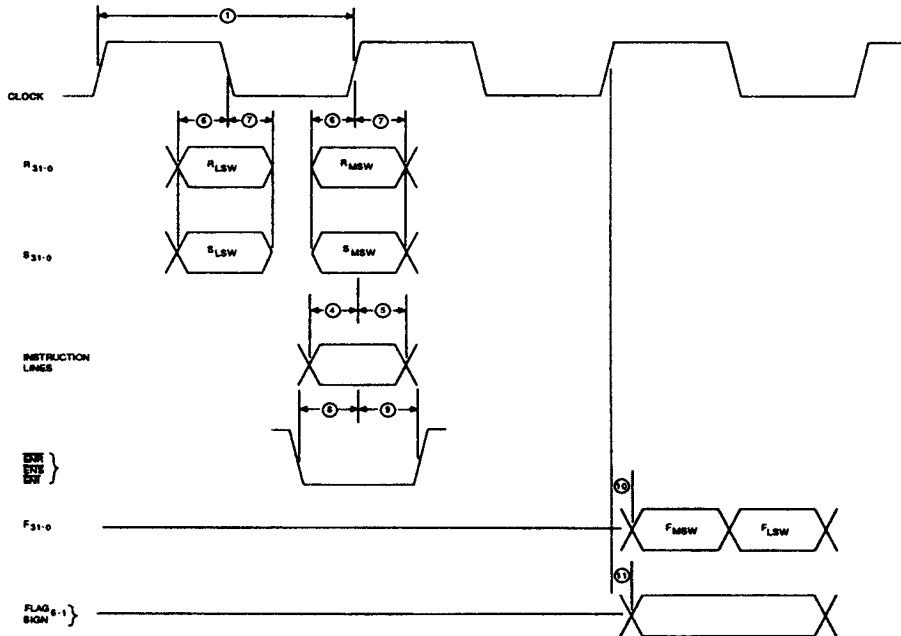
KS000010



WF025013

Input Clock Timing

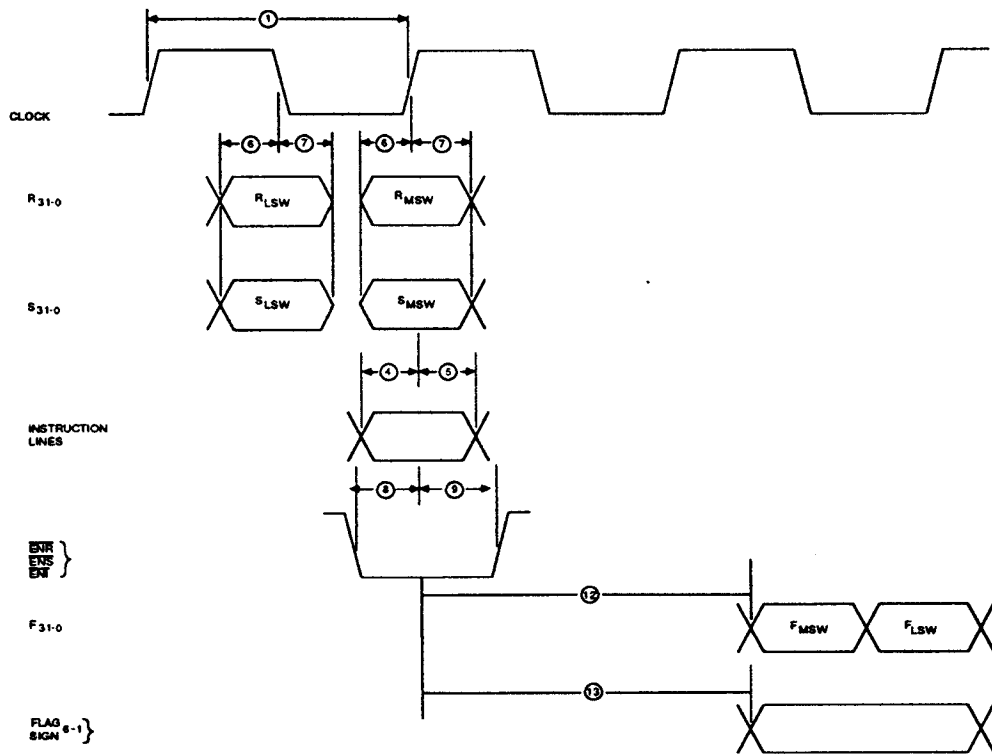
SWITCHING WAVEFORMS (Cont'd.)



WF025023

Timing of Operations with F Register and Status Clocked. Assumes 32-Bit Bus, Single-Cycle, LSW-First Input Mode and Flow-Through Operation

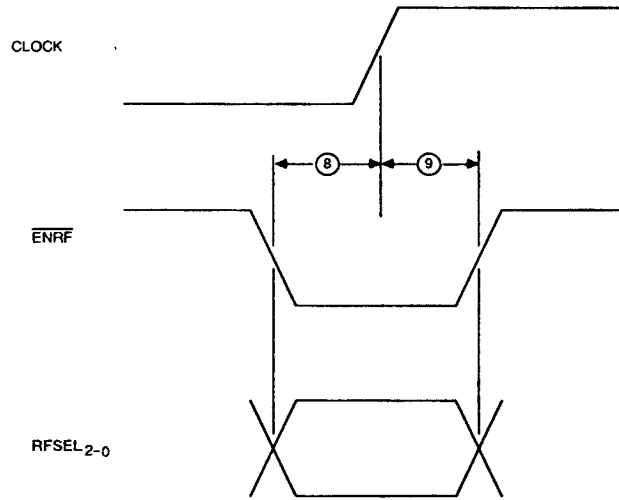
SWITCHING WAVEFORMS (Cont'd.)



WF025033

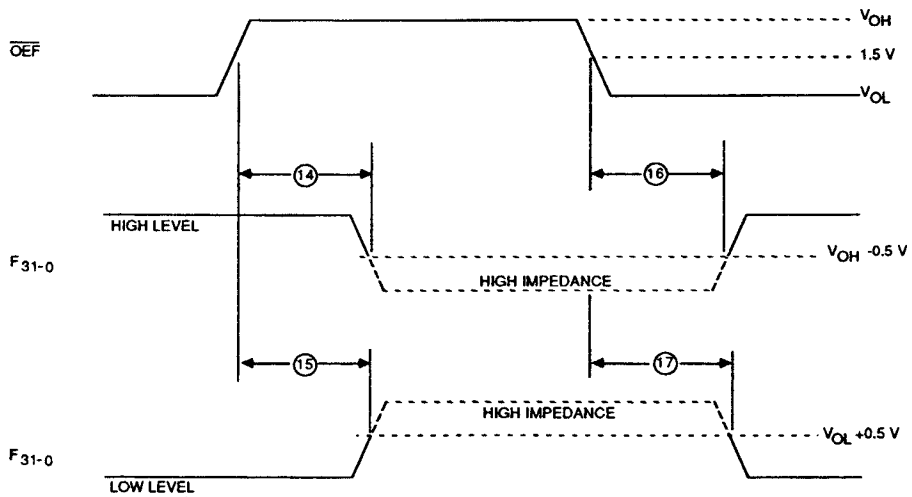
Timing of Operations with F-Register and Status Register in Feedthrough Mode. Assumes 32-Bit Bus, Single-Cycle, LSW-First Input Mode and Flow-Through Operation.

SWITCHING WAVEFORMS (Cont'd.)



WF025042

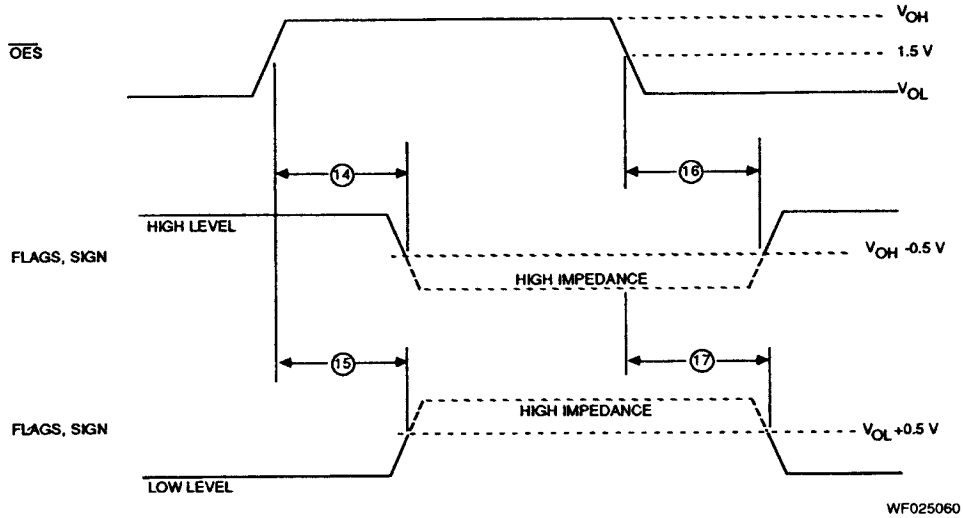
Register File Control Timing



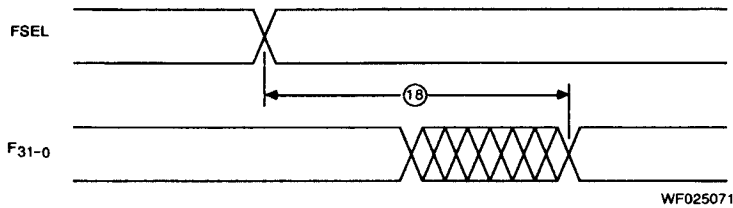
WF025051

Enable/Disable Timing for F₀₋₃₁

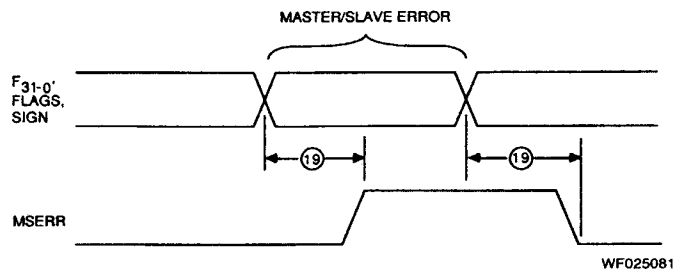
SWITCHING WAVEFORMS (Cont'd.)



Enable/Disable Timing for FLAG₁₋₆ and SIGN

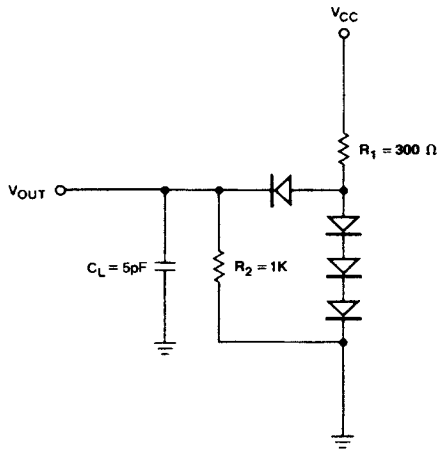


Output Selection Timing



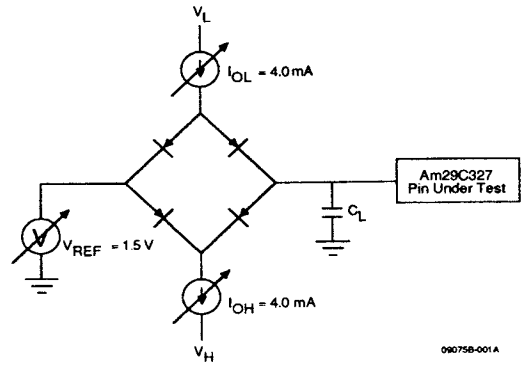
Master/Slave Timing (Assumes SLAVE Mode)

SWITCHING TEST CIRCUITS



A. Three-State Outputs

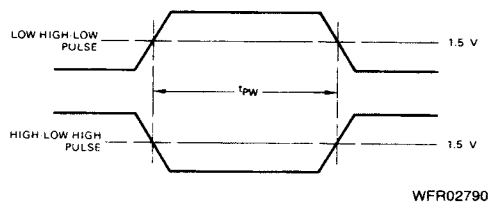
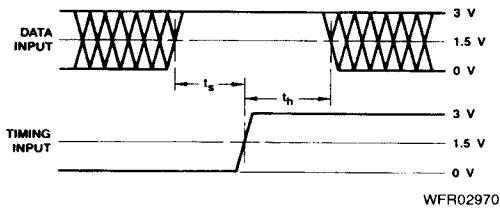
TCR01334



B. All Other Outputs

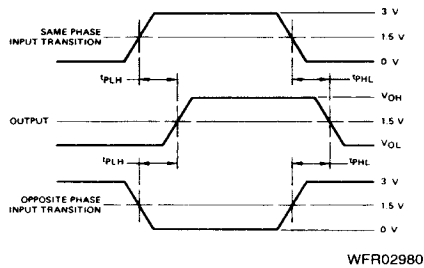
090756-001A
IC001032

SWITCHING TEST WAVEFORMS

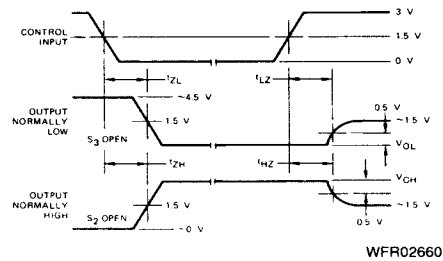


- Notes: 1. Diagram shown for HIGH data only. Output transition may be opposite sense.
2. Cross-hatched area is don't care condition.

Setup, Hold, and Release Times



Pulse Width



- Notes: 1. Diagram shown for Input Control Enable-LOW and Input Control Disable-HIGH.
2. S₁, S₂ and S₃ of Load Circuit are closed except where shown.

Propagation Delay

Enable and Disable Times

TEST PHILOSOPHY AND METHODS

The following eight points describe AMD's philosophy for high volume, high speed automatic testing.

1. Ensure that the part is adequately decoupled at the test head. Large changes in V_{CC} current as the device switches may cause erroneous function failures due to V_{CC} changes.
2. Do not leave inputs floating during any tests, as they may start to oscillate at high frequency.
3. Do not attempt to perform threshold tests at high speed. Following an output transition, ground current may change by as much as 400 mA in 5-8 ns. Inductance in the ground cable may allow the ground pin at the device to rise by hundreds of millivolts momentarily.
4. Use extreme care in defining point input levels for AC tests. Many inputs may be changed at once, so there will be significant noise at the device pins and they may not actually reach V_{IL} or V_{IH} until the noise has settled. AMD recommends using $V_{IL} \leq 0$ V and $V_{IH} \geq 3.0$ V for AC tests.
5. To simplify failure analysis, programs should be designed to perform DC, Function, and AC tests as three distinct groups of tests.
6. Capacitive Loading for AC Testing

Automatic testers and their associated hardware have stray capacitance that varies from one type of tester to another, but is generally around 50 pF. This, of course, makes it impossible to make direct measurements of parameters which call for smaller capacitive load than the associated stray capacitance. Typical examples of this are the so-called "float delays," which measure the propagation delays into the high-impedance state and are usually specified at a load capacitance of 5.0 pF. In these cases, the test is performed at the higher load capacitance (typically 50 pF), and engineering correlations based on data taken with a bench setup are used to predict the result at the lower capacitance.

Similarly, a product may be specified at more than one capacitive load. Since the typical automatic tester is not capable of switching loads in mid-test, it is impossible to make measurements at both capacitances even though they may both be greater than the stray capacitance. In these cases, a measurement is made at one of the two capacitances. The result at the other capacitance is predicted from engineering correlations based on data taken with a bench setup and the knowledge that certain DC measurements (I_{OH} , I_{OL} for example) have already been taken and are within spec. In some cases, special DC tests are performed in order to facilitate this correlation.

7. Threshold Testing

The noise associated with automatic testing (due to the long, inductive cables) and the high gain of the tested device when in the vicinity of the actual device threshold, frequently give rise to oscillations when testing high speed circuits. These oscillations are not indicative of a reject device, but instead of an overtaxed test system. To minimize this problem, thresholds are tested at least once for each input pin. Thereafter, "hard" high and low levels are used for other tests. Generally this means that function and AC testing are performed at "hard" input levels rather than at V_{IL} Max. and V_{IH} Min.

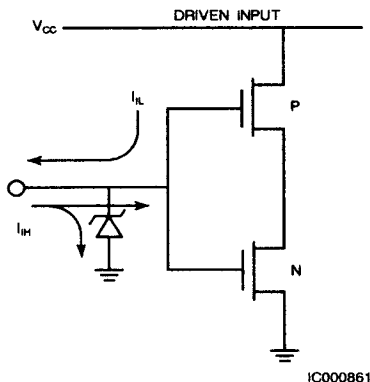
8. AC Testing

Occasionally, parameters are specified that cannot be measured directly on automatic testers because of tester limitations. Data input hold times often fall into this category. In these cases, the parameter in question is guaranteed by correlating these tests with other AC tests that have been performed. These correlations are arrived at by the cognizant engineer by using precise bench measurements in conjunction with the knowledge that certain DC parameters have already been measured and are within spec.

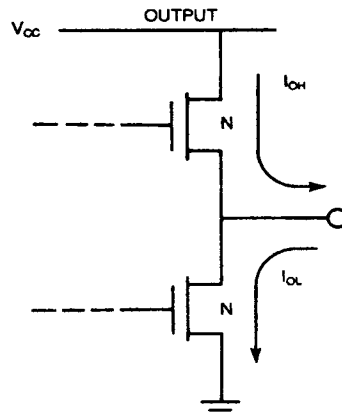
In some cases, certain AC tests are redundant, since they can be shown to be predicted by some other tests which have already been performed. In these cases, the redundant tests are not performed.

2

INPUT/OUTPUT CIRCUIT DIAGRAMS



IC000861



IC000870

APPENDICES

APPENDIX A — DATA FORMATS

The following data formats are supported: 32-bit integer, 64-bit integer, IEEE single-precision, IEEE double-precision, DEC F, DEC D, DEC G, IBM single-precision, and IBM double-precision.

The primary and alternate floating-point formats are selected by mode register bits M3 to M0. The user may select between floating-point operations and integer operations by means of instruction bit I5.

The nine supported formats are described below:

Integer Formats

32-Bit Integer

The 32-bit integer word is arranged as follows:

Bit 31 30 29 28 27 26 25 7 6 5 4 3 2 1 0

2^{31}	2^{30}	2^{29}	2^{28}	2^{27}	2^{26}	2^{25}		2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
----------	----------	----------	----------	----------	----------	----------	-----------	-------	-------	-------	-------	-------	-------	-------	-------

TB001030

The 32-bit word is interpreted as a two's-complement integer. For integer multiplications, the user has the option of interpreting integers as unsigned. An unsigned single-precision integer

has a format similar to that of the two's-complement integer, but with an MSB weight of 2^{31} .

64-Bit Integer

The 64-bit integer word is arranged as follows:

Bit 63 62 61 60 59 58 57 7 6 5 4 3 2 1 0

2^{63}	2^{62}	2^{61}	2^{60}	2^{59}	2^{58}	2^{57}		2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
----------	----------	----------	----------	----------	----------	----------	-----------	-------	-------	-------	-------	-------	-------	-------	-------

TB001040

The 64-bit word is interpreted as a two's-complement integer. For integer multiplications, the user has the option of interpreting integers as unsigned. An unsigned double-precision integer

has a format similar to that of the two's-complement integer, but with an MSB weight of 2^{63} .

IEEE Formats

IEEE Single-Precision

The IEEE single-precision word is 32 bits wide and is arranged in the format as follows:

31	30	29	28	27	26	25	24	23	22	21	20	19	18	. . .	3	2	1	0
s	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^1	2^2	2^3	2^4	2^5	. . .	2^{20}	2^{21}	2^{22}	2^{23}
sign	biased exponent (e)								fraction (f)									

TB001050

The floating-point word is divided into three fields: a single-bit sign, an 8-bit biased exponent, and a 23-bit fraction.

The sign bit is 0 for positive numbers and 1 for negative numbers. Zero may have either sign.

The biased exponent is an 8-bit unsigned integer representing a multiplicative factor of some power of two. The bias value is 127. If, for example, the multiplicative value for a floating-point

number is to be 2^a , the value of the biased exponent is $a + 127$, where "a" is the true exponent.

The fraction is a 23-bit unsigned fractional field containing the 23 least-significant bits of the floating-point number's 24-bit mantissa. The weight of the fraction's most-significant bit is 2^{-1} . The weight of the least-significant bit is 2^{-23} .

An IEEE floating-point number is evaluated or interpreted as follows:

If $e = 255$ and $f \neq 0$	value = NaN	Not-a-Number
If $e = 255$ and $f = 0$	value = $(-1)^s \infty$	Infinity
If $0 < e < 255$	value = $(-1)^s 2^{e-127} (1.f)$	Normalized number
If $e = 0$ and $f \neq 0$	value = $(-1)^s 2^{-126} (0.f)$	Denormalized number
If $e = 0$ and $f = 0$	value = $(-1)^s 0$	Zero

Infinity: Infinity can have either a positive or negative sign. The interpretation of infinities is determined by the Affine/Projective select input AFF/PROJ.

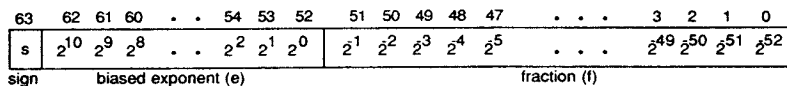
NaN: A NaN is interpreted as a signal or symbol. NaNs are used to indicate invalid operations, and as a means of passing process status through a series of calculations. They arise in two ways: either generated by the Am29C327 to indicate an

invalid operation, or provided by the user as an input. A signaling NaN has the MSB of its fraction set to 0 and at least one of the remaining fraction bits set to 1. A quiet NaN has the MSB of its fraction set to 1.

The IEEE format is fully described in ANSI/IEEE Standard 754-1985.

IEEE Double-Precision

The IEEE double-precision word is 64 bits wide and is arranged in the format shown below:



TB001060

The floating-point word is divided into three fields: a single-bit sign, an 11-bit biased exponent, and a 52-bit fraction.

The sign bit is 0 for positive numbers and 1 for negative numbers; zero may have either sign.

The biased exponent is an 11-bit unsigned integer representing a multiplicative factor of some power of two. The bias value is 1023. If, for example, the multiplicative value for a

floating-point number is to be 2^a , the value of the biased exponent is $a + 1023$, where "a" is the true exponent.

The fraction is a 52-bit unsigned fractional field containing the 52 least-significant bits of the floating-point number's 53-bit mantissa. The weight of the fraction's most-significant bit is 2^{-1} . The weight of the least-significant bit is 2^{-52} .

An IEEE floating-point number is evaluated or interpreted as follows:

If $e = 2047$ and $f \neq 0$	value = Reserved operand	Not-a-Number
If $e = 2047$ and $f = 0$	value = $(-1)^s \infty$	Infinity
If $0 < e < 2047$	value = $(-1)^s 2^{e-1023} (1.f)$	Normalized number
If $e = 0$ and $f \neq 0$	value = $(-1)^s 2^{-1022} (0.f)$	Denormalized number
If $e = 0$ and $f = 0$	value = $(-1)^s 0$	Zero

Infinity: Infinity can have either a positive or negative sign. The interpretation of infinities is determined by the Affine/Projective select input AFF/PROJ.

NaN: A NaN is interpreted as a signal or symbol. NaNs are used to indicate invalid operations, and as a means of passing process status through a series of calculations. They arise in two ways: either generated by the Am29C327 to indicate an

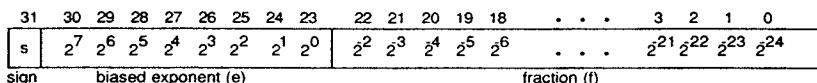
invalid operation, or provided by the user as an input. A signaling NaN has the MSB of its fraction set to 0 and at least one of the remaining fraction bits set to 1. A quiet NaN has the MSB of its fraction set to 1.

The IEEE format is fully described in ANSI/IEEE Standard 754-1985.

DEC Formats

DEC F

The DEC F word is 32 bits wide and is arranged in the format shown below:



TB001070

The floating-point word is divided into three fields: a single-bit sign, an 8-bit biased exponent, and a 23-bit fraction.

The sign bit is 0 for positive numbers and 1 for negative numbers; zero has a positive sign.

The biased exponent is an 8-bit unsigned integer representing a multiplicative factor of some power of two. The bias value is 128. If, for example, the multiplicative value for a floating-point number is to be 2^a , the value of the biased exponent is $a + 128$, where "a" is the true exponent.

The fraction is a 23-bit unsigned fractional field containing the 23 least-significant bits of the floating-point number's 24-bit mantissa. The weight of the fraction's most-significant bit is 2^{-2} . The weight of the least-significant bit is 2^{-24} .

A DEC F floating-point number is evaluated or interpreted as follows:

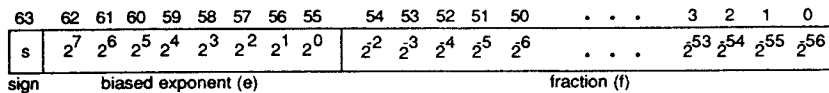
If $e \neq 0$	value $\neq (-1)^s 2^{e-128} (0.1f)$
If $s = 0$ and $e = 0$	value = 0
If $s = 1$ and $e = 0$	value = DEC-Reserved Operand

DEC-Reserved Operand: A DEC-Reserved Operand is interpreted as a signal or symbol. DEC-Reserved Operands are used to indicate invalid operations and operations whose results have overflowed the destination format. They may also be used to pass symbolic information from one calculation to another.

The DEC formats are fully described in the VAX Architecture Manual.

DEC D

The DEC D word is 64 bits wide and is arranged in the format shown below:



TB001080

The floating-point word is divided into three fields: a single-bit sign, an 8-bit biased exponent, and a 55-bit fraction.

The sign bit is 0 for positive numbers and 1 for negative numbers; zero has a positive sign.

The biased exponent is an 8-bit unsigned integer representing a multiplicative factor of some power of two. The bias value is 128. If, for example, the multiplicative value for a floating-point number is to be 2^a , the value of the biased exponent is $a + 128$, where "a" is the true exponent.

The fraction is a 55-bit unsigned fractional field containing the 55 least-significant bits of the floating-point number's 56-bit mantissa. The weight of the fraction's most-significant bit is 2^{-2} . The weight of the least-significant bit is 2^{-56} .

A DEC D floating-point number is evaluated or interpreted as follows:

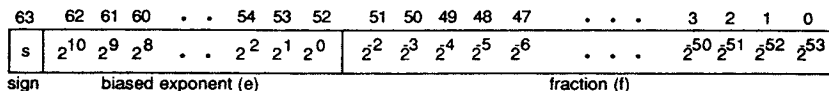
If $e \neq 0$ value = $(-1)^s 2^e - 128 (0.1f)$
 If $s = 0$ and $e = 0$ value = 0
 If $s = 1$ and $e = 0$ value = DEC-Reserved Operand

DEC-Reserved Operand: A DEC-Reserved Operand is interpreted as a signal or symbol. DEC-Reserved Operands are used to indicate invalid operations and operations whose results have overflowed the destination format. They may also be used to pass symbolic information from one calculation to another.

The DEC formats are fully described in the VAX Architecture Manual.

DEC G

The DEC G word is 64 bits wide and is arranged in the format shown below:



TB001090

The floating-point word is divided into three fields: a single-bit sign, an 11-bit biased exponent, and a 52-bit fraction.

The sign bit is 0 for positive numbers and 1 for negative numbers; zero has a positive sign.

The biased exponent is an 11-bit unsigned integer representing a multiplicative factor of some power of two. The bias value is 1024. If, for example, the multiplicative value for a floating-point number is to be 2^a , the value of the biased exponent is $a + 1024$, where "a" is the true exponent.

The fraction is a 52-bit unsigned fractional field containing the 52 least-significant bits of the floating-point number's 53-bit mantissa. The weight of the fraction's most-significant bit is 2^{-2} . The weight of the least-significant bit is 2^{-53} .

A DEC G floating-point number is evaluated or interpreted as follows:

If $e \neq 0$ value = $(-1)^s 2^e - 1024 (0.1f)$
 If $s = 0$ and $e = 0$ value = 0
 If $s = 1$ and $e = 0$ value = DEC-Reserved Operand

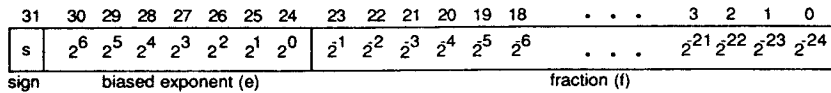
DEC-Reserved Operand: A DEC-Reserved Operand is interpreted as a signal or symbol. DEC-Reserved Operands are used to indicate invalid operations and operations whose results have overflowed the destination format. They may also be used to pass symbolic information from one calculation to another.

The DEC formats are fully described in the VAX Architecture Manual.

IBM Formats

IBM Single-Precision

The IBM single-precision word is 32 bits wide and is arranged in the format shown below:



TB001100

The floating-point word is divided into three fields: a single-bit sign, a 7-bit biased exponent, and a 24-bit fraction.

The sign bit is 0 for positive numbers and 1 for negative numbers; a True-zero has a positive sign.

The biased exponent is a 7-bit unsigned integer representing a multiplicative factor of some power of 16. The bias value is 64. If, for example, the multiplicative value for a floating-point number is to be 16^a , the value of the biased exponent is $a + 64$, where "a" is the true exponent.

The fraction is a 24-bit unsigned fractional field containing the 24 least-significant bits of the floating-point number's 25-bit

mantissa. The weight of the fraction's most-significant bit is 2^{-1} . The weight of the least-significant bit is 2^{-24} .

An IBM floating-point number is evaluated or interpreted as follows:

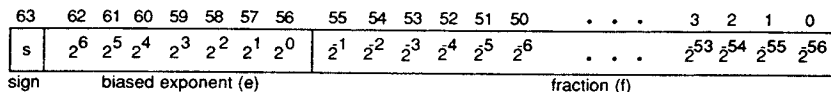
$$\text{value} = (-1)^s 16^e - 64(0.f)$$

Zero: There are two classes of zero. If the sign, biased exponent and fraction are all zero, the operand is known as a "True Zero." If the fraction is zero, but the sign and biased exponent are not both zero, the operand is known as a "Floating-point Zero."

The IBM format is fully described in the IBM System/370 Principles of Operation Manual.

IBM Double-Precision

The IBM double-precision word is 64 bits wide and is arranged in the format shown below:



TB001110

The floating-point word is divided into three fields: a single-bit sign, a 7-bit biased exponent, and a 56-bit fraction.

The sign bit is 0 for positive numbers and 1 for negative numbers; a True-zero has a positive sign.

The biased exponent is a 7-bit unsigned integer representing a multiplicative factor of some power of 16. The bias value is 64. If, for example, the multiplicative value for a floating-point number is to be 16^a , the value of the biased exponent is $a + 64$, where "a" is the true exponent.

The fraction is a 56-bit unsigned fractional field containing the 56 least-significant bits of the floating-point number's 57-bit

mantissa. The weight of the fraction's most-significant bit is 2^{-1} . The weight of the least-significant bit is 2^{-56} .

An IBM floating-point number is evaluated or interpreted as follows:

$$\text{value} = (-1)^s 16^e - 64(0.f)$$

Zero: There are two classes of zero. If the sign, biased exponent and fraction are all zero, the operand is known as a "True Zero." If the fraction is zero, but the sign and biased exponent are not both zero, the operand is known as a "Floating-point Zero."

The IBM format is fully described in the IBM System/370 Principles of Operation Manual.

APPENDIX B — ROUNDING MODES

The Am29C327 provides six rounding modes for floating-point operations, and for integer multiplication. The rounding mode for an operation is selected by the input pins RM₂-RM₀.

RM ₂	RM ₁	RM ₀	Round Mode
0	0	0	Round to Nearest (IEEE)
0	0	1	Round to Minus Infinity
0	1	0	Round to Plus Infinity
0	1	1	Round to Zero
1	0	0	Round to Nearest (DEC)
1	0	1	Round Away From Zero
1	1	X	Illegal Value

Round to Nearest (IEEE)

The infinitely precise result of an operation is rounded to the closest representable value in the destination format. If the infinitely precise result is exactly halfway between two representations, it is rounded to the representation having a least-significant bit of zero.

Round to Minus Infinity (IEEE)

The infinitely precise result of an operation is rounded to the closest representable value in the destination format that is less than or equal to the infinitely precise result. This rounding mode conforms to the "round to minus infinity" mode described in the IEEE Floating-Point Standard.

The IEEE standard specifies that all four "IEEE" modes be available so that the user may select the mode most appropriate for the algorithm being executed. The DEC standard specifies that two rounding modes be available - Round-to-Nearest (DEC) and Round-to-Zero. The IBM standard specifies that all operations be performed using the Round-to-Zero mode.

It should be noted, however, that the Am29C327 permits any of the supported rounding modes to be selected, regardless of the format of the operation. It is permissible to use one of the IEEE rounding modes with an IBM operation, or DEC rounding with an IEEE operation, or any other possible combination. For those integer operations where rounding is performed, any rounding mode may be chosen. This flexibility allows the user to select the mode most appropriate for the arithmetic environment in which the processor is operating.

Round to Plus Infinity (IEEE)

The infinitely precise result of an operation is rounded to the closest representable value in the destination format that is greater than or equal to the infinitely precise result.

Round to Zero (IEEE)

The infinitely precise result of an operation is rounded to the closest representable value in the destination format whose magnitude is less than or equal to the infinitely precise result.

Round to Nearest (DEC)

The infinitely precise result of an operation is rounded to the closest representable value in the destination format. If the infinitely precise result is exactly halfway between two representations, it is rounded to the representation having the greater magnitude.

Round Away from Zero

The infinitely precise result of an operation is rounded to the closest representable value in the destination format whose magnitude is greater than or equal to the infinitely precise result.

A graphical representation of these rounding modes is shown in Figures B1-1 and B1-2.

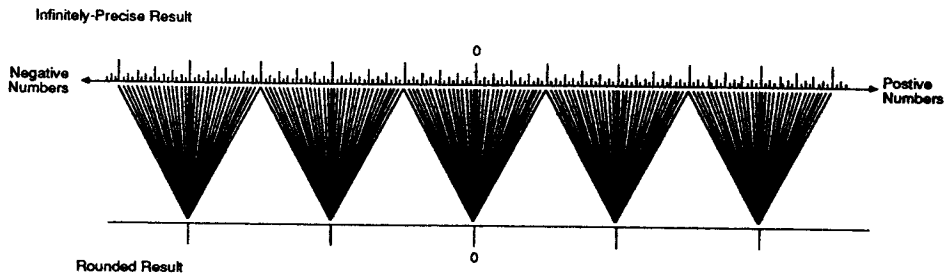


Figure B1. Illustration of IEEE Round-to-Nearest Rounding Mode

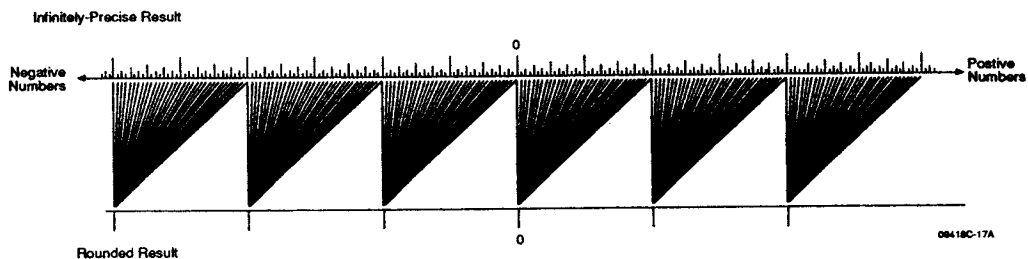


Figure B2. Illustration of IEEE Round-to-Minus-Infinity Rounding Mode

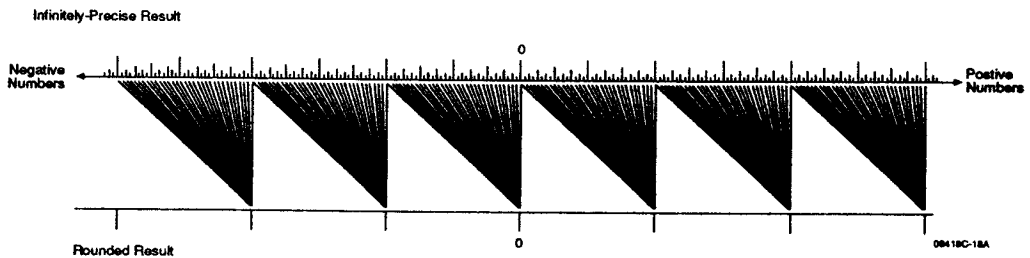


Figure B3. Illustration of IEEE Round-to-Plus-Infinity Rounding Mode

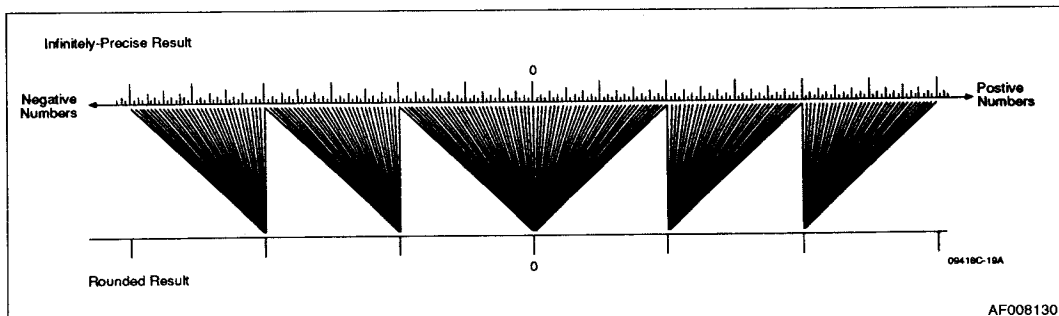


Figure B4. Illustration of IEEE Round-to-Zero Rounding Mode

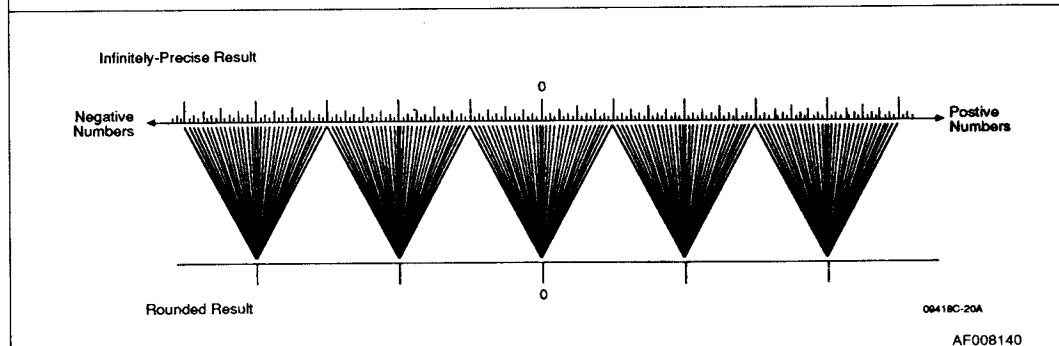


Figure B5. Illustration of DEC Round-to-Nearest Rounding Mode

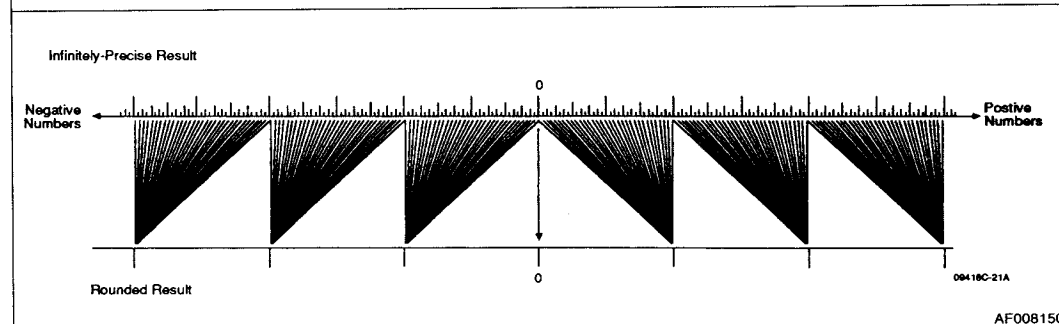


Figure B6. Illustration of Round-Away-From-Zero Rounding Mode

APPENDIX C — DEVIATIONS FROM FLOATING-POINT STANDARDS

There are several cases in which the implementation of the IEEE, DEC and IBM floating-point standards in the Am29C327 differs from the formal definitions of those standards. This appendix describes these deviations from the standards.

Deviations from the IEEE Standard

Section 7.3 of the IEEE-754 standard specifies that "Trapped overflow on conversion from a binary floating-point format shall deliver to the trap handler a result in that or a wider format, possibly with the exponent bias adjusted, but rounded to the destination's precision."

According to the IEEE standard, then, if a double-to-single IEEE operation overflows while traps are enabled, the result is a double-precision operand, rounded to single-precision width (23-bit fraction), together with a correctly-adjusted (double-precision) exponent and the appropriate flags for a trapped overflow.

In the case of an overflow in any IEEE operation, the Am29C327 returns a result in the destination format specified by the user, rounded to that destination format.

In the case of the double-to-single overflow described above, the result from the Am29C327 is a single-precision operand, together with a correctly-adjusted (single-precision) exponent and the appropriate flags for a trapped overflow.

A simple example serves to illustrate the discrepancy, by describing the conversion of the double-precision IEEE number 52B123456789ABCD to single-precision, with traps enabled and the round-to-nearest rounding mode selected. This number is too large to be represented in single-precision format.

According to the IEEE standard, the result of this operation is the double-precision number 52B1234560000000, comprising the double-precision exponent of the input and a fraction truncated to 23 bits, together with flags V and X.

When the operation is performed in the Am29C327, however, using the "F" = "P" operation with appropriate precision controls, the result is the single-precision number 75891A2B,

comprising the single-precision (overflowed) exponent reduced by 192 (decimal) and a single-precision fraction, together with flags V and X.

It should be noted that trapped operation is an optional part of the IEEE standard. Full adherence to the IEEE specification of trapped operation is therefore not necessary to ensure compliance with IEEE-754.

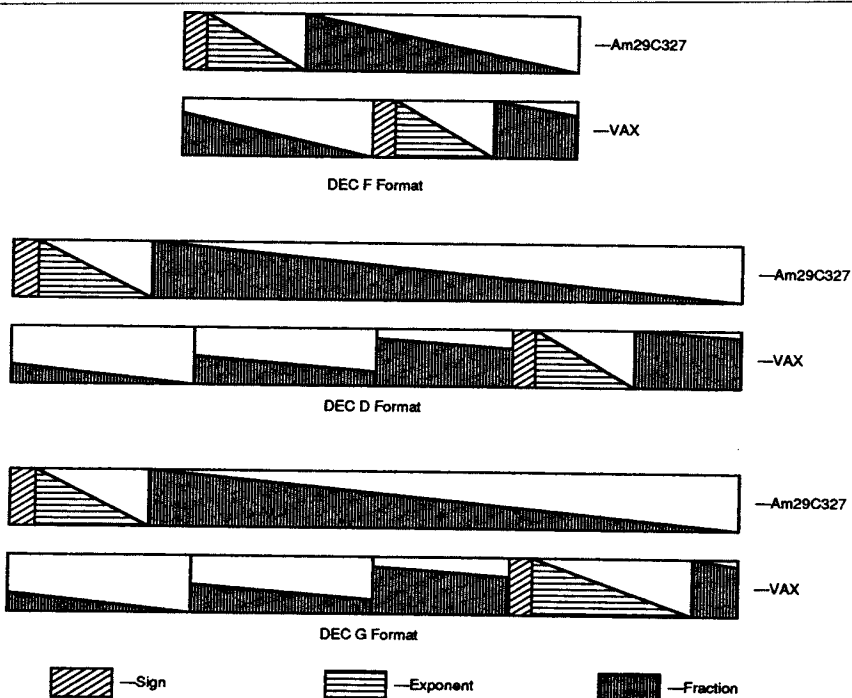
Deviations from the DEC F, DEC D and DEC G Standards

The DEC F, DEC D and DEC G standards, as implemented in the Am29C327, differ from the implementations in a VAX only in the way in which the sub-fields of the floating-point word are arranged. The differences are listed in Table C1.

TABLE C1. DIFFERENCES IN Am29C327 AND DEC FLOATING-POINT FORMATS

	Am29C327 Arrangement	VAX Arrangement
DEC F	sign: bit 31 exponent: bits 30-23 fraction: bits 22-0	sign: bit 15 exponent: bits 14-7 fraction: bits 6-0, bits 31-16
DEC D	sign: bit 63 exponent: bits 62-55 fraction: bits 54-0	sign: bit 15 exponent: bits 14-7 fraction: bits 6-0, bits 31-16, bits 47-32, bits 63-48
DEC G	sign: bit 63 exponent: bits 62-52 fraction: bits 51-0	sign: bit 15 exponent: bits 14-4 fraction: bits 3-0, bits 31-16, bits 47-32, bits 63-48

The discrepancies are shown graphically in Figure C1. Within each exponent and fraction field, the shading illustrates the weighting of the bits, from the MSB to LSB.



08418C-22A

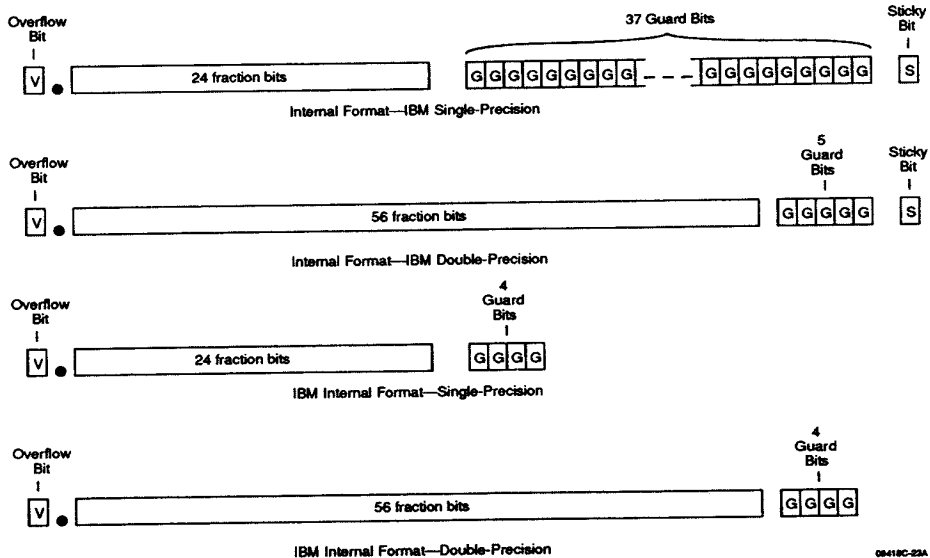
AF008160

**Figure C1. Differences in DEC Representations
Between the Am29C327 and the VAX**

Deviations from the IBM Standard

The Am29C327's deviations from the IBM standard may be summarized as follows, assuming that the user has selected the round-to-nearest rounding mode:

1. The Am29C327 provides more guard bits in its internal format than specified by the IBM standard. With certain combinations of input operands, the Am29C327 produces more accurate results than a standard IBM processor for instructions based on addition operations and comparisons (i.e. IBM instructions with an op-code, 14-0, of 00001 or 00011).
2. The discrepancies are much larger for single-precision operations than double-precision operations, because the difference in the number of guard bits is much greater (33 more for single, one more for double).
3. There is no universal rule for determining whether a given set of input operands will result in a discrepancy. Provided the conditions in (1) above are met, the user must examine each operation on a case-by-case basis, taking into account the input operands and the internal formats discussed in this section.
4. The Am29C327 does not produce unnormalized results from additions. The results of all addition operations are renormalized.



08-41/IC-23A

TB001230

Figure C2. Differences in Internal Mantissa Formats of an IBM CPU and the Am29C327

It should be mentioned that the discrepancies due to the above effects, for both additions and multiplications, are typically insignificant when compared to the magnitude of the result itself. They become an important issue only when the user requires exact compatibility with an existing implementation of the IBM standard.

The term "accuracy" in this discussion is used to describe the difference between the final result of an operation and its infinitely-precise result. In the context of adherence to a standard, however, "accuracy" might better be defined as the difference between the final result of an operation executed in the Am29C327 and the final result of the same operation executed in a CPU which exactly meets that standard.