Am29C325

CMOS 32-Bit Floating-Point Processor

FINAL

DISTINCTIVE CHARACTERISTICS

- Single VLSI device performs high-speed single-precision floating-point arithmetic
  Floating-point addition, subtraction, and multiplication in a single clock cycle
  Internal architecture supports sum-of-products, Newton-Raphson division
- 32-bit, three-bus flow-through architecture
  Programmable I/O allows interface to 32- and 16-bit systems
- IEEE and DEC formats
  Performs conversions between formats
  Performs integer ↔ floating-point conversions
- Input and output registers may be made transparent independently
- Pin and functionally compatible with the bipolar Am29325
- The Am29C325 uses less than one-quarter the power of the Am29325
- 145 PGA requires no heatsink

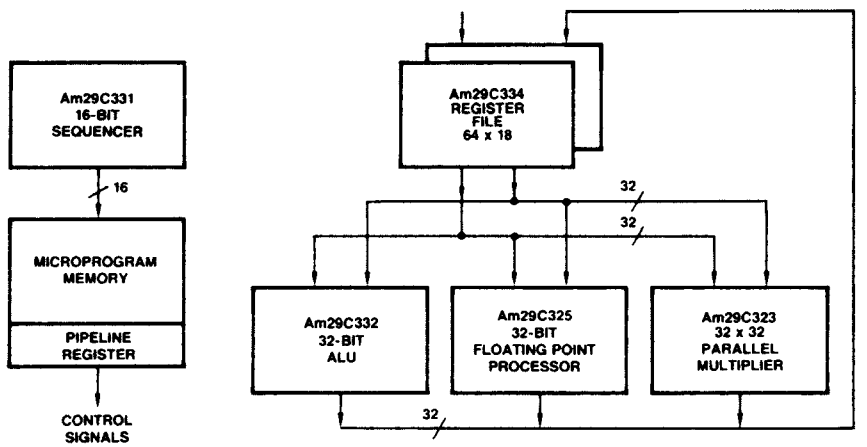
GENERAL DESCRIPTION

The Am29C325 is a high-speed floating-point processor unit. It performs 32-bit single-precision floating-point addition, subtraction, and multiplication operations in a single VLSI circuit, using the format specified by the IEEE floating-point standard, 754. The DEC single-precision floating-point format is also supported. Operations for conversion between 32-bit integer format and floating-point format are available, as are operations for converting between the IEEE and DEC floating-point formats. Any instruction can be performed in a single clock cycle. Six flags — invalid operation, inexact result, zero, not-a-number, overflow, and underflow — monitor the status of operations.

The Am29C325 has a three-bus, 32-bit architecture, with two input buses and one output bus. This configuration provides high I/O bandwidth, allows access to all buses, and affords a high degree of flexibility when connecting this device in a system. All buses are registered, with each register having a clock enable. Input and output registers may be made transparent independently. Two other I/O configurations, a 32-bit, two-bus architecture and a 16-bit, three-bus architecture, are user-selectable, easing interface with a wide variety of systems. Thirty-two-bit internal feedforward datapaths support accumulation operations, including sum-of-products and Newton-Raphson division.

Fabricated using Advanced Micro Devices' 1.2 micron CMOS process, the Am29C325 is powered by a single 5-volt supply. The device is housed in a 145-lead pin-grid-array package.

Am29C300 FAMILY HIGH-PERFORMANCE SYSTEM BLOCK DIAGRAM



AF004651

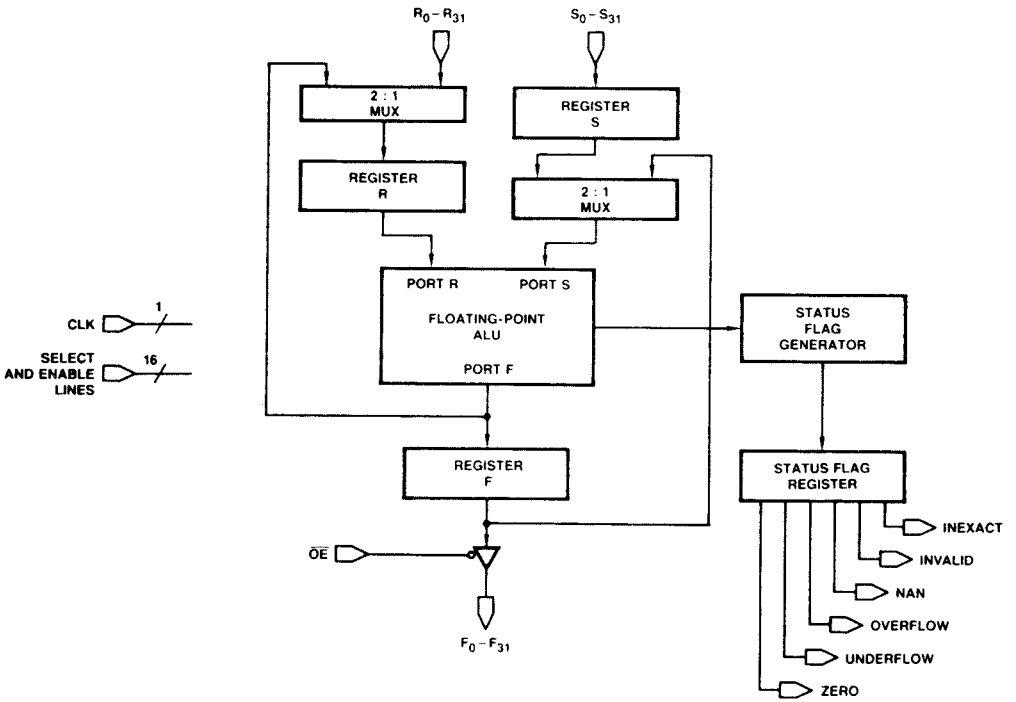
Publication #	Rev.	Amendment
07783	C	/0

Issue Date: March 1989

RELATED AMD PRODUCTS

Part No.	Description
Am29027	Am29K Arithmetic Accelerator
Am29116	High-Performance Bipolar 16-Bit Microprocessor
Am29C116	High-Performance CMOS 16-Bit Microprocessor
Am29CPL141	CMOS 64 x 16 EPROM Field Programmable Controller
Am29CPL142	CMOS 128 x 16 EPROM Field Programmable Controller
Am29CPL144	CMOS 512 x 16 EPROM Field Programmable Controller
Am29CPL151	CMOS 64 x 16 EPROM Field Programmable Controller — Slim DIP
Am29CPL152	CMOS 128 x 16 EPROM Field Programmable Controller — Slim DIP
Am29CPL154	CMOS 512 x 16 EPROM Field Programmable Controller — Slim DIP
Am29C323	CMOS 32-Bit Parallel Multiplier
Am29C327	CMOS Double-Precision Floating-Point Processor
Am29331	16-Bit Microprogram Sequencer
Am29C331	CMOS 16-Bit Microprogram Sequencer
Am29332	32-Bit Extended Function ALU
Am29C332	CMOS 32-Bit Extended Function ALU
Am29334	64 x 18 Four-Port, Dual-Access Register File
Am29C334	CMOS 64 x 18 Four-Port, Dual-Access Register File

BLOCK DIAGRAM



BD007080

CONNECTION DIAGRAM
Top View

PGA

	A	B	C	D	E	F	G	H	J	K	L	M	N	P	R
1	INEX	I2	I1	$\overline{ENF}$	I4	OBUS	$\overline{OE}$	VCC	CLK	R31	R30	R25	R24	R21	R20
2	INVA	NAN	I0	I/$\overline{D}$	FT0	FT1	VCC	VCC	RND0	RND1	R27	R28	R23	R22	R17
3	F29	ZERO	GNDO	$\overline{ENR}$	$\overline{ENS}$	16/$\overline{32}$	VCC	VCC	VCC	R29	R26	GND	GND	R19	R18
4	F30	F31	GNDO	*									R15	R16	R13
5	F23	OVFL	UNFL										R14	R11	R12
6	F26	F27	F28										R9	R10	R7
7	F21	F24	F25										R8	R5	R6
8	F22	F19	VCCO										R3	R4	R1
9	F17	F20	VCCO										R0	I3	R2
10	F18	F15	F16										S28	S31	S30
11	F13	F14	F11										S27	S26	S29
12	F12	F9	F10										VCC	S25	S24
13	F7	F6	GNDO	GNDO	GNDO	GNDO	GND	GND	GND	S8	S13	S14	VCC	S22	S23
14	F8	F3	F2	GNDO	F0	S1	S2	GND	S4	S9	S10	S15	S18	S21	S20
15	F5	F4	F1	GNDO	P/$\overline{AFF}$	S0	S3	S5	S7	S6	S11	S12	S17	S16	S19

CD010492

Key:
16/$\overline{32}$ = S16/$\overline{32}$
I/$\overline{D}$ = IEEE/$\overline{DEC}$
INEX = INEXACT
INVA = INVALID
OBUS = ONEBUS
OVFL = OVERFLOW
P/$\overline{AFF}$ = PROJ/$\overline{AFF}$
UNFL = UNDERFLOW

*D4 is an alignment pin (not connected internally).

PIN DESIGNATIONS

(Sorted by Pin No.)

PIN NO.	PIN NAME	PIN NO.	PIN NAME	PIN NO.	PIN NAME	PIN NO.	PIN NAME
A-1	Inexact	**C-7**	F_{25}	**H-13**	GND	**N-10**	S_{28}
A-2	Invalid	**C-8**	V_{CCO}	**H-14**	GND	**N-11**	S_{27}
A-3	F_{29}	**C-9**	V_{CCO}	**H-15**	S_5	**N-12**	V_{CC}
A-4	F_{30}	**C-10**	F_{16}	**J-1**	CLK	**N-13**	V_{CC}
A-5	F_{23}	**C-11**	F_{11}	**J-2**	RND_0	**N-14**	S_{18}
A-6	F_{26}	**C-12**	F_{10}	**J-3**	V_{CC}	**N-15**	S_{17}
A-7	F_{21}	**C-13**	GNDO	**J-13**	GND	**P-1**	R_{21}
A-8	F_{22}	**C-14**	F_2	**J-14**	S_4	**P-2**	R_{22}
A-9	F_{17}	**C-15**	F_1	**J-15**	S_7	**P-3**	R_{19}
A-10	F_{18}	**D-1**	$\overline{ENF}$	**K-1**	R_{31}	**P-4**	R_{16}
A-11	F_{13}	**D-2**	IEEE/$\overline{DEC}$	**K-2**	RND_1	**P-5**	R_{11}
A-12	F_{12}	**D-3**	$\overline{ENR}$	**K-3**	R_{29}	**P-6**	R_{10}
A-13	F_7	**D-13**	GNDO	**K-13**	S_8	**P-7**	R_5
A-14	F_8	**D-14**	GNDO	**K-14**	S_9	**P-8**	R_4
A-15	F_5	**D-15**	GNDO	**K-15**	S_6	**P-9**	I_3
B-1	I_2	**E-1**	I_4	**L-1**	R_{30}	**P-10**	S_{31}
B-2	NAN	**E-2**	FT_0	**L-2**	R_{27}	**P-11**	S_{26}
B-3	ZERO	**E-3**	$\overline{ENS}$	**L-3**	R_{26}	**P-12**	S_{25}
B-4	F_{31}	**E-13**	GNDO	**L-13**	S_{13}	**P-13**	S_{22}
B-5	OVERFLOW	**E-14**	F_0	**L-14**	S_{10}	**P-14**	S_{21}
B-6	F_{27}	**E-15**	PROJ/$\overline{AFF}$	**L-15**	S_{11}	**P-15**	S_{16}
B-7	F_{24}	**F-1**	ONEBUS	**M-1**	R_{25}	**R-1**	R_{20}
B-8	F_{19}	**F-2**	FT_1	**M-2**	R_{28}	**R-2**	R_{17}
B-9	F_{20}	**F-3**	S16/$\overline{32}$	**M-3**	GND	**R-3**	R_{18}
B-10	F_{15}	**F-13**	GNDO	**M-13**	S_{14}	**R-4**	R_{13}
B-11	F_{14}	**F-14**	S_1	**M-14**	S_{15}	**R-5**	R_{12}
B-12	F_9	**F-15**	S_0	**M-15**	S_{12}	**R-6**	R_7
B-13	F_6	**G-1**	$\overline{OE}$	**N-1**	R_{24}	**R-7**	R_6
B-14	F_3	**G-2**	V_{CC}	**N-2**	R_{23}	**R-8**	R_1
B-15	F_4	**G-3**	V_{CC}	**N-3**	GND	**R-9**	R_2
C-1	I_1	**G-13**	GND	**N-4**	R_{15}	**R-10**	S_{30}
C-2	I_0	**G-14**	S_2	**N-5**	R_{14}	**R-11**	S_{29}
C-3	GNDO	**G-15**	S_3	**N-6**	R_9	**R-12**	S_{24}
C-4	GNDO	**H-1**	V_{CC}	**N-7**	R_8	**R-13**	S_{23}
C-5	UNDERFLOW	**H-2**	V_{CC}	**N-8**	R_3	**R-14**	S_{20}
C-6	F_{28}	**H-3**	V_{CC}	**N-9**	R_0	**R-15**	S_{19}

Note: Pin number D4 = 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.

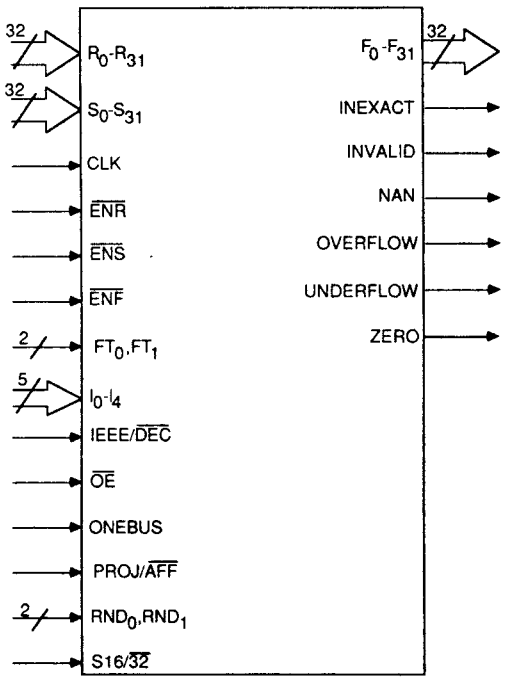
PIN DESIGNATIONS (Cont'd.)

(Sorted by Pin Name)

PIN NO.	PIN NAME	PIN NO.	PIN NAME.	PIN NO.	PIN NAME	PIN NO.	PIN NAME.
J-1	CLK	E-2	FT_0	R-6	R_7	K-14	S_9
D-1	$\overline{ENF}$	F-2	FT_1	N-7	R_8	L-14	S_{10}
D-3	$\overline{ENR}$	G-13	GND	N-6	R_9	L-15	S_{11}
E-3	$\overline{ENS}$	H-13	GND	P-6	R_{10}	M-15	S_{12}
E-14	F_0	H-14	GND	P-5	R_{11}	L-13	S_{13}
C-15	F_1	J-13	GND	R-5	R_{12}	M-13	S_{14}
C-14	F_2	M-3	GND	R-4	R_{13}	M-14	S_{15}
B-14	F_3	N-3	GND	N-5	R_{14}	P-15	S_{16}
B-15	F_4	C-3	GNDO	N-4	R_{15}	F-3	$S16/\overline{32}$
A-15	F_5	C-4	GNDO	P-4	R_{16}	N-15	S_{17}
B-13	F_6	C-13	GNDO	R-2	R_{17}	N-14	S_{18}
A-13	F_7	D-13	GNDO	R-3	R_{18}	R-15	S_{19}
A-14	F_8	D-14	GNDO	P-3	R_{19}	R-14	S_{20}
B-12	F_9	D-15	GNDO	R-1	R_{20}	P-14	S_{21}
C-12	F_{10}	E-13	GNDO	P-1	R_{21}	P-13	S_{22}
C-11	F_{11}	F-13	GNDO	P-2	R_{22}	R-13	S_{23}
A-12	F_{12}	C-2	I_0	N-2	R_{23}	R-12	S_{24}
A-11	F_{13}	C-1	I_1	N-1	R_{24}	P-12	S_{25}
B-11	F_{14}	B-1	I_2	M-1	R_{25}	P-11	S_{26}
B-10	F_{15}	P-9	I_3	L-3	R_{26}	N-11	S_{27}
C-10	F_{16}	E-1	I_4	L-2	R_{27}	N-10	S_{28}
A-9	F_{17}	D-2	IEEE/$\overline{DEC}$	M-2	R_{28}	R-11	S_{29}
A-10	F_{18}	A-1	INEXACT	K-3	R_{29}	R-10	S_{30}
B-8	F_{19}	A-2	INVALID	L-1	R_{30}	P-10	S_{31}
B-9	F_{20}	B-2	NAN	K-1	R_{31}	C-5	UNDERFLOW
A-7	F_{21}	G-1	$\overline{OE}$	J-2	RND_0	G-2	V_{CC}
A-8	F_{22}	F-1	ONEBUS	K-2	RND_1	G-3	V_{CC}
A-5	F_{23}	B-5	OVERFLOW	F-15	S_0	H-1	V_{CC}
B-7	F_{24}	E-15	PROJ/$\overline{AFF}$	F-14	S_1	H-2	V_{CC}
C-7	F_{25}	N-9	R_0	G-14	S_2	H-3	V_{CC}
A-6	F_{26}	R-8	R_1	G-15	S_3	J-3	V_{CC}
B-6	F_{27}	R-9	R_2	J-14	S_4	N-12	V_{CC}
C-6	F_{28}	N-8	R_3	H-15	S_5	N-13	V_{CC}
A-3	F_{29}	P-8	R_4	K-15	S_6	C-8	V_{CCO}
A-4	F_{30}	P-7	R_5	J-15	S_7	C-9	V_{CCO}
B-4	F_{31}	R-7	R_6	K-13	S_8	B-3	ZERO

Note: Pin number D4 = 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.

LOGIC SYMBOL

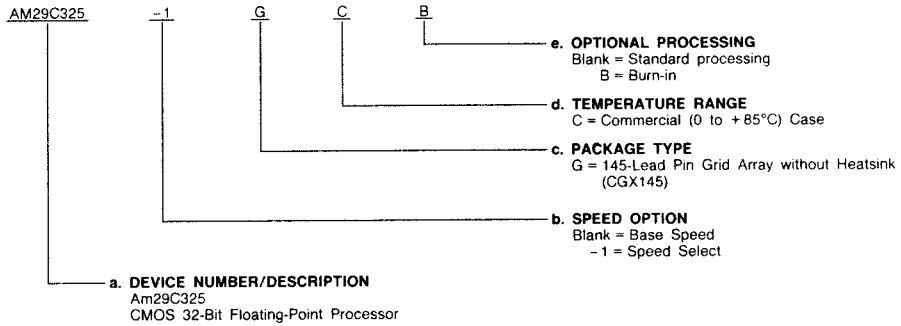


LS002920

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	
AM29C325	GC, GCB
AM29C325-1	

Valid Combinations

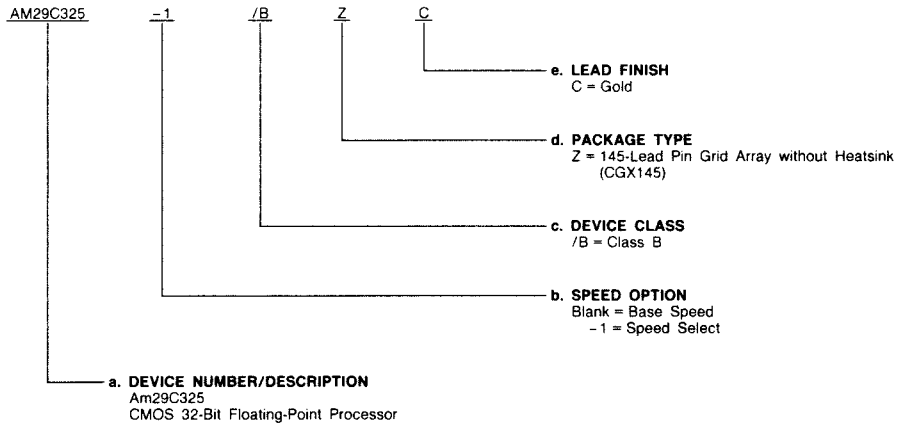
Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations, to check on newly released combinations, and to obtain additional data on AMD's standard military grade products.

2

MILITARY ORDERING INFORMATION

APL Products

AMD products for Aerospace and Defense 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) for APL products 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	
AM29C325	/BZC
AM29C325-1	

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 or to check for newly released valid combinations.

Group A Tests

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

PIN DESCRIPTION

CLK Clock (Input)
For the internal registers.

$\overline{ENF}$ Register F Clock Enable (Input; Active LOW)
When $\overline{ENF}$ is LOW, register F is clocked on the LOW-to-HIGH transition of CLK. When $\overline{ENF}$ is HIGH, register F retains the previous contents.

$\overline{ENR}$ Register R Clock Enable (Input; Active LOW)
When $\overline{ENR}$ is LOW, register R is clocked on the LOW-to-HIGH transition of CLK. When $\overline{ENR}$ is HIGH, register R retains the previous contents.

$\overline{ENS}$ Register S Clock Enable (Input; Active LOW)
When $\overline{ENS}$ is LOW, register S is clocked on the LOW-to-HIGH transition of CLK. When $\overline{ENS}$ is HIGH, register S retains the previous contents.

$F_0 - F_{31}$ F Operand Bus (Output)
F_0 is the least significant bit.

FT_0 Input Register Feedthrough Control (Input; Active HIGH)
When FT_0 is HIGH, registers R and S are transparent.

FT_1 Output Register Feedthrough Control (Input; Active HIGH)
When FT_1 is HIGH, register F and the status flag register are transparent.

$I_0 - I_2$ Operation Select Lines (Input)
Used to select the operation to be performed by the ALU. See Table 1 for a list of operations and the corresponding codes.

I_3 ALU S Port Input Select (Input)
A LOW on I_3 selects register S as the input to the ALU S port. A HIGH on I_3 selects register F as the input to the ALU S port.

I_4 Register R Input Select (Input)
A LOW on I_4 selects R_0–R_{31} as the input to register R. A HIGH selects the ALU F port as the input to register R.

IEEE/$\overline{DEC}$ IEEE/$\overline{DEC}$ Mode Select (Input)
When IEEE/$\overline{DEC}$ is HIGH, IEEE mode is selected. When IEEE/$\overline{DEC}$ is LOW, DEC mode is selected.

INEXACT Inexact Result Flag (Output; Active HIGH)
A HIGH indicates that the final result of the last operation was not infinitely precise, due to rounding.

INVALID Invalid Operation Flag (Output; Active HIGH)
A HIGH indicates that the last operation performed was invalid; e.g., ∞ times 0.

NAN Not-a-Number Flag (Output; Active HIGH)
A HIGH indicates that the final result produced by the last operation is not to be interpreted as a number. The output in such cases is either an IEEE Not-a-Number (NAN) or a DEC-reserved operand.

$\overline{OE}$ Output Enable (Input; Active LOW)
When $\overline{OE}$ is LOW, the contents of register F are placed on F_0–F_{31}. When $\overline{OE}$ is HIGH, F_0–F_{31} assume a high-impedance state.

ONEBUS Input Bus Configuration Control (Input)
A LOW on ONEBUS configures the input bus circuitry for two-input bus operation. A HIGH on ONEBUS configures the input bus circuitry for single-input bus operation.

OVERFLOW Overflow Flag (Output; Active HIGH)
A HIGH indicates that the last operation produced a final result that overflowed the floating-point format.

PROJ/$\overline{AFF}$ Projective/$\overline{Affine}$ Mode Select (Input)
Choice of projective or affine mode determines the way in which infinities are handled in IEEE mode. A LOW on PROJ/$\overline{AFF}$ selects affine mode; a HIGH selects projective mode.

$R_0 - R_{31}$ R Operand Bus (Input)
R_0 is the least significant bit.

RND_0, RND_1 Rounding Mode Selects (Input)
RND_0 and RND_1 select one of four rounding modes. See Table 5 for a list of rounding modes and the corresponding control codes.

$S_0 - S_{31}$ S Operand Bus (Input)
S_0 is the least significant bit.

S16/$\overline{32}$ 16- or $\overline{32}$-Bit I/O Mode Select (Input)
A LOW on S16/$\overline{32}$ selects the 32-bit I/O mode; a HIGH selects the 16-bit I/O mode. In 32-bit mode, input and output buses are 32 bits wide. In 16-bit mode, input and output buses are 16 bits wide, with the least and most significant portions of the 32-bit input and output words being placed on the buses during the HIGH and LOW portions of CLK, respectively.

UNDERFLOW Underflow Flag (Output; Active HIGH)
A HIGH indicates that the last operation produced a rounded result that underflowed the floating-point format.

ZERO Zero Flag (Output; Active HIGH)
A HIGH indicates that the last operation produced a final result of zero.

Definition of Terms

Affine Mode

One of two modes affecting the handling of operations on infinities — see the **Operations with Infinities** section under **Operations in IEEE Mode.**

Biased Exponent

The true exponent of a floating-point number, plus a constant. For IEEE floating-point numbers, the constant is 127; for DEC floating-point numbers, the constant is 128. See also **True Exponent.**

Bus

Data input or output channel for the floating-point processor.

DEC-Reserved Operand

A DEC floating-point number that is interpreted as a symbol and has no numeric value. A DEC-reserved operand has a sign of 1 and a biased exponent of 0.

Destination Format

The format of the final result produced by the floating-point ALU. The destination format can be IEEE floating point, DEC floating point, or integer.

Final Result

The result produced by the floating-point ALU.

Fraction

The 23 least-significant bits of the mantissa.

Infinitely Precise Result

The result that would be obtained from an operation if both exponent range and precision were unbounded.

Input Operands

The value or values on which an operation is performed. For example, the addition 2 + 3 = 5 has input operands 2 and 3.

Mantissa

The portion of a floating-point number containing the number's significant bits. For the floating-point number 1.101×2^{-3}, the mantissa is 1.101.

NAN (Not-a-Number)

An IEEE floating-point number that is interpreted as a symbol and has no numeric value. A NAN has a biased exponent of 255_{10} and a non-zero fraction.

Port

Data input or output channel for the floating-point ALU.

Projective Mode

One of two modes affecting the handling of operations on infinities — see the **Operations with Infinities** section under **Operation in IEEE Mode.**

Rounded Result

The result produced by rounding the infinitely precise result to fit the destination format.

True Exponent (or Exponent)

Number representing the power of two by which a floating-point number's mantissa is to be multiplied. For the floating-point number 1.101×2^{-3}, the true exponent is −3.

FUNCTIONAL DESCRIPTION

Architecture

The Am29C325 comprises a high-speed, floating-point ALU, a status flag generator, and a 32-bit data path.

Floating-Point ALU

The floating-point ALU performs 32-bit floating-point operations. It also performs floating-point-to-integer conversions, integer-to-floating-point floating-point conversions, and conversions between the IEEE and DEC formats. The ALU has two 32-bit input ports, R and S, and a 32-bit output port, F.

Conceptually, the process performed by the ALU can be divided into three stages (see Figure 1). The operation stage performs the arithmetic operation selected by the user; the output of this section is referred to as the infinitely precise result of the operation. The rounding stage rounds the infinitely precise result to fit in the destination format; the output of this stage is called the rounded result. The last stage checks for exceptional conditions. If no exceptional condition is found, the rounded result is passed through this stage. If some exceptional condition is found (e.g., overflow, underflow, or an invalid operation), this section may replace the rounded result with another output, such as $+\infty$, $-\infty$, a NAN, or a DEC-reserved operand. The output of this last stage appears on port F and is called the final result.

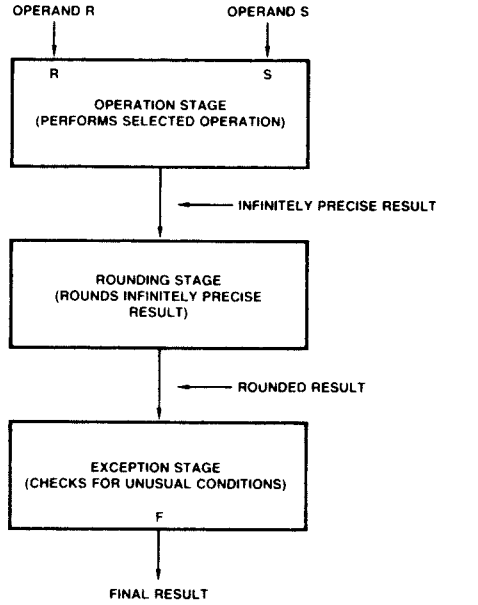


Figure 1. Conceptual Model of the Process Performed by the Floating-Point ALU

The ALU performs one of eight operations; the operation to be performed is selected by placing the appropriate control code on lines I_0–I_2. Table 1 gives the control codes corresponding to each of the eight operations.

The floating-point addition operation (R PLUS S) adds the floating-point numbers on ports R and S and places the floating-point result on port F. In IEEE mode (IEEE/$\overline{\text{DEC}}$ = HIGH) the addition is performed in IEEE floating-point format; in DEC mode (IEEE/$\overline{\text{DEC}}$ = LOW) the addition is performed in DEC format.

The floating-point subtraction operation (R MINUS S) subtracts the floating-point number on port S from the floating-point number on port R and places the floating-point result on port F. In IEEE mode (IEEE/$\overline{\text{DEC}}$ = HIGH) the subtraction is performed in IEEE floating-point point format; in DEC mode (IEEE/$\overline{\text{DEC}}$ = LOW) the subtraction is performed in DEC format.

The floating-point multiplication operation (R TIMES S) multiplies the floating-point numbers on ports R and S and places the floating-point result on port F. In IEEE mode (IEEE/$\overline{\text{DEC}}$ = HIGH) the multiplication is performed in IEEE floating-point format; in DEC mode (IEEE/$\overline{\text{DEC}}$ = LOW) the multiplication is performed in DEC format.

The floating-point constant subtraction (2 MINUS S) operation subtracts the floating-point value on port S from 2 and places the result on port F. The operand on port R is not used in this operation; its value will not affect the operation in any way. In IEEE mode (IEEE/$\overline{\text{DEC}}$ = HIGH) the operation is performed in IEEE floating-point format; in DEC mode (IEEE/$\overline{\text{DEC}}$ = LOW) the operation is performed in DEC format. This operation is

used to support Newton-Raphson floating-point division; a description of its use appears in **Appendix C.**

The integer-to-floating-point conversion (INT-TO-FP) operation takes a 32-bit, two's complement integer on port R and places the equivalent floating-point value on port F. The operand on port S is not used in this operation; its value will not affect the operation in any way. In IEEE mode (IEEE/$\overline{DEC}$ = HIGH) the result is delivered in IEEE format; in DEC mode (IEEE/$\overline{DEC}$ = LOW) the result is delivered in DEC format.

TABLE 1. ALU OPERATION SELECT

I_2	I_1	I_0	Operation	Output Equation
0	0	0	Floating-point addition (R PLUS S)	F = R + S
0	0	1	Floating-point subtraction (R MINUS S)	F = R - S
0	1	0	Floating-point multiplication (R TIMES S)	F = R * S
0	1	1	Floating-point constant subtraction (2 MINUS S)	F = 2 - S
1	0	0	Integer-to-floating-point conversion (INT-TO-FP)	F (floating-point) = R (integer)
1	0	1	Floating-point-to-integer conversion (FP-TO-INT)	F (integer) = R (floating-point)
1	1	0	IEEE-TO-DEC format conversion (IEEE-TO-DEC)	F (DEC format) = R (IEEE format)
1	1	1	DEC-TO-IEEE format conversion (DEC-TO-IEEE)	F (IEEE format) = R (DEC format)

The floating-point-to-integer conversion (FP-TO-INT) operation takes a floating-point number on port R and places the equivalent 32-bit, two's complement integer value on port F. The operand on port S is not used in this operation; its value will not affect the operation in any way. In IEEE mode (IEEE/$\overline{DEC}$ = HIGH) the operand on port R is interpreted using the IEEE floating-point format; in DEC mode (IEEE/$\overline{DEC}$ = LOW) it is interpreted using the DEC floating-point format.

The IEEE-to-DEC conversion operation (IEEE-TO-DEC) takes an IEEE-format floating-point number on port R and places the equivalent DEC-format floating-point number on port F. The operand on port S is not used in this operation; its value will not affect the operation in any way. The operation can be performed in either IEEE mode (IEEE/$\overline{DEC}$ = HIGH) or DEC mode (IEEE/$\overline{DEC}$ = LOW).

The DEC-to-IEEE conversion operation (DEC-TO-IEEE) takes a DEC-format floating-point number on port R and places the equivalent IEEE-floating-point number on port F. The operand on port S is not used in this operation; its value will not affect the operation in any way. The operation can be performed in either IEEE mode (IEEE/$\overline{DEC}$ = HIGH) or DEC mode (IEEE/$\overline{DEC}$ = LOW).

Status Flag Generator

The status flag generator controls the state of six flags that report the status of floating-point ALU operations. The flags indicate when an operation is invalid (e.g., ∞ times 0) or when an operation has produced an overflow, an underflow, a non-numerical result (e.g., a NAN- or DEC-reserved operand), an inexact result, or a result of zero. The flags represent the status of the most recently performed operation. Flag status is stored in the flag status register on the LOW-to-HIGH transition of CLK. When the output register feedthrough control FT_1 is HIGH, the flag status register is made transparent.

Data Path

The 32-bit data path consists of the R and S input buses; the F output bus; data registers R, S, and F; the register R input multiplexer; and the ALU port S input multiplexer.

Input operands enter the floating-point processor through the 32-bit R and S input buses, R_0–R_{31} and S_0–S_{31}. Results of operations appear on the 32-bit F bus, F_0–F_{31}. The F bus assumes a high-impedance state when output enable $\overline{OE}$ is HIGH.

The R and S registers store input operands; the F register stores the final result of the floating-point ALU operation. Each register has an independent clock enable ($\overline{ENR}$, $\overline{ENS}$, and $\overline{ENF}$). When a register's clock enable is LOW, the register stores the data on its input at the LOW-to-HIGH transition of CLK; when the clock enable is HIGH, the register retains its current data. All data registers are fully edge-triggered — both the input data and the register enable need only meet modest setup and hold time requirements. Registers R and S can be made transparent by setting FT_0, the input register feedthrough control, HIGH. Register F can be made transparent by setting FT_1, the output register feedthrough control, HIGH.

The register R input multiplexer selects either the R input bus or the floating-point ALU's F port as the input to register R. Selection is controlled by I_4 — a LOW selects the R input bus; a HIGH selects the ALU F port. The ALU port S input multiplexer selects either register S or register F as the input to the floating-point ALU's S port. Selection is controlled by I_3 — a LOW selects register S; a HIGH selects register F.

Data selected by I_3 and I_4 is described in Table 2. When registers R and S are transparent (FT_0 = HIGH), multiplexer select I_4 must be kept LOW, so that the register R input multiplexer selects R_0–R_{31}. When register F is transparent (FT_1 = HIGH), multiplexer select I_3 must be kept LOW, so that the ALU port S input multiplexer selects register S.

TABLE 2. MUX SELECT

I_3	Data selected for floating-point ALU S port
0	Register S
1	Register F
I_4	**Data selected for register R input**
0	R bus
1	Floating-point ALU port F

TABLE 3. I/O MODE SELECTION

S16/$\overline{32}$	ONEBUS	I/O Mode
0	0	32-bit, two-input-bus mode
0	1	32-bit, single-input-bus mode*
1	0	16-bit, two-input-bus mode*
1	1	Illegal I/O mode selection value

*FT_0 must be held LOW in this mode (see text).

I/O Modes

The Am29C325 datapath can be configured in one of three I/O modes: a 32-bit, two-input bus mode; a 32-bit, single-input bus mode; and a 16-bit, two-input bus mode. These modes affect only the manner in which data is delivered to and taken from the Am29C325; operation of the floating-point ALU is not altered. The I/O mode is selected with the ONEBUS and S16/$\overline{32}$ controls. Table 3 lists the control codes needed to invoke each I/O mode.

32-Bit, Two-Input Bus Mode

In this I/O mode, the R and S buses are configured as independent 32-bit input buses, and the F bus is configured as a 32-bit output bus. Figure 2 is a functional block diagram of the Am29C325 in this I/O mode.

R and S operands are taken from their respective input buses and clocked into the R and S registers on the LOW-to-HIGH transition of CLK. Register F is also clocked on the LOW-to-HIGH transition of CLK. Figure 5a depicts typical I/O timing in this mode.

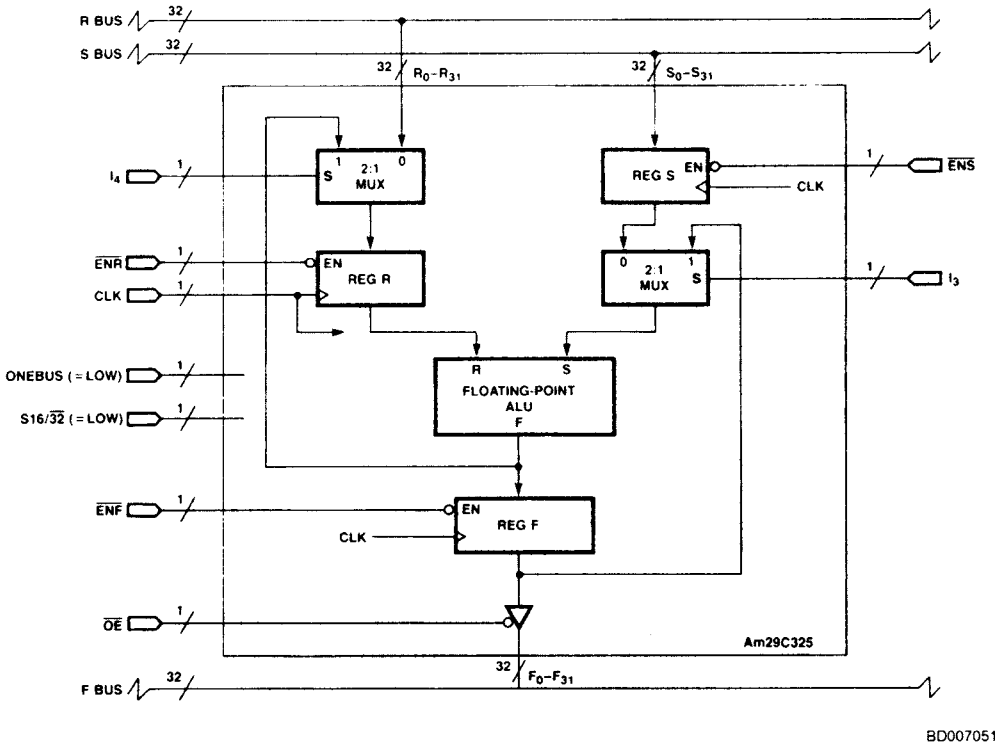


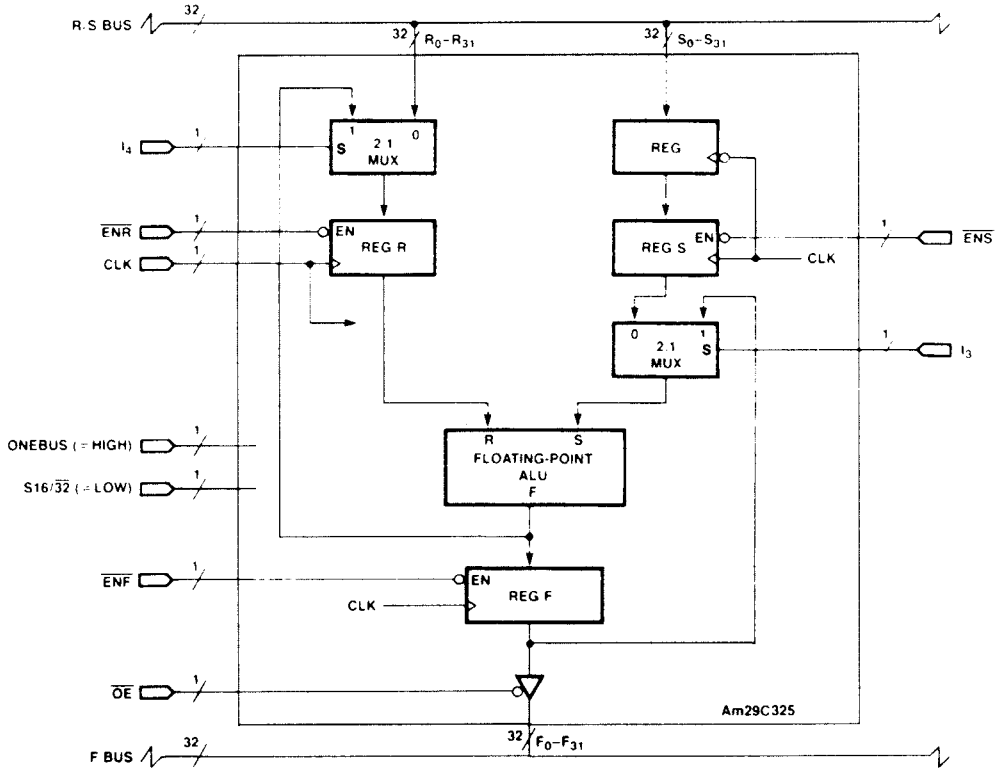
Figure 2. Functional Block Diagram for the 32-Bit, Two-Input Bus Mode

32-Bit, Single-Input Bus Mode

In this I/O mode, the R and S buses are connected to a single 32-bit multiplexed input data bus; the F bus is configured as an independent 32-bit output bus. Figure 3 is a functional block diagram of the Am29C325 in this I/O mode. Note that both the R and S bus lines must be wired to the input bus.

R and S operands are multiplexed onto the input bus by the host system. The S operand is clocked from the input bus into a temporary holding register on the HIGH-to-LOW transition of CLK and is transferred to register S on the LOW-to-HIGH transition of CLK. The R operand is clocked from the input bus into register R on the LOW-to-HIGH transition of CLK. Register F is clocked on the LOW-to-HIGH transition of CLK. Figure 5b depicts typical I/O timing in this mode.

When placed in this I/O mode, the data path will not function properly if the R and S registers are made transparent. Therefore, input register feedthrough control FT_0 must be held LOW in this mode.



BD007062

Figure 3. Functional Block Diagram for the 32-Bit, Single-Input Bus Mode

16-Bit, Two-Input Bus Mode

In this I/O mode, the R and S buses are configured as independent 16-bit input buses, and the F bus is configured as a 16-bit output bus. Figure 4 is a functional block diagram of the Am29C325 in this I/O mode. Note that the 16 least significant bits (LSBs) and 16 most significant bits (MSBs) of the R, S, and F buses must be wired to their respective system buses in parallel.

Thirty-two-bit operands are passed along the 16-bit data buses by time-multiplexing the 16 LSBs and 16 MSBs of each 32-bit word. For the R input bus, the host system multiplexes the 16 LSBs and 16 MSBs of the R operand onto the 16-bit R bus. The 16 LSBs of the R operand are stored in a temporary holding register on the HIGH-to-LOW transition of CLK. The 16 MSBs are clocked into register R on the LOW-to-HIGH transition of CLK; at the same time, the 16 LSBs are transferred from the temporary holding register to register R. Transfer of data from the S input bus to the S register takes place in a similar fashion. Register F is clocked on the LOW-to-HIGH transition of CLK. Circuitry internal to the Am29C325 multiplexes data from register F onto the 16-bit output bus by enabling the 16 LSBs of the F output bus when CLK is HIGH and enabling the 16 MSBs of the F output bus when CLK is LOW. Figure 5c depicts typical I/O timing in this mode.

When placed in this I/O mode, the data path will not function properly if the R and S registers are made transparent. Therefore, input register feedthrough control FT_0 must be held LOW in this mode. Caution must also be taken in controlling the register R input multiplexer control line, I_4, in this I/O mode. I_4 should be changed only when CLK is HIGH, in addition to meeting the setup and hold time requirements given in the **Switching Characteristics** section.

Operation in IEEE Mode

When input signal IEEE/$\overline{DEC}$ is HIGH, the IEEE mode of operation is selected. In this mode the Am29C325 uses the floating-point format set forth in the IEEE Proposed Standard for Binary Floating-Point Arithmetic, P754. In addition, the IEEE mode complies with most other aspects of single-precision floating-point operation outlined in the proposed standard — differences are discussed in **Appendix A.**

IEEE Floating-Point Format

The IEEE single-precision floating-point word is 32 bits wide and is arranged in the format shown in Figure 6. 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 indicates the sign of the floating-point number's value. Non-negative values have a sign of 0; negative values, a sign of 1. The value zero may have either sign.

The biased exponent is an 8-bit unsigned integer field representing a multiplicative factor of some power of two. The bias value is 127. If, for example, the multiplicative factor for a floating-point number is to be 2^a, the value of the biased exponent would be a + 127; "a" is called the true exponent.

The fraction is a 23-bit unsigned fraction field containing the 23 LSBs of the floating-point number's 24-bit mantissa. The weight of fraction's MSB is 2^{-1}; the weight of the LSB is 2^{-23}.

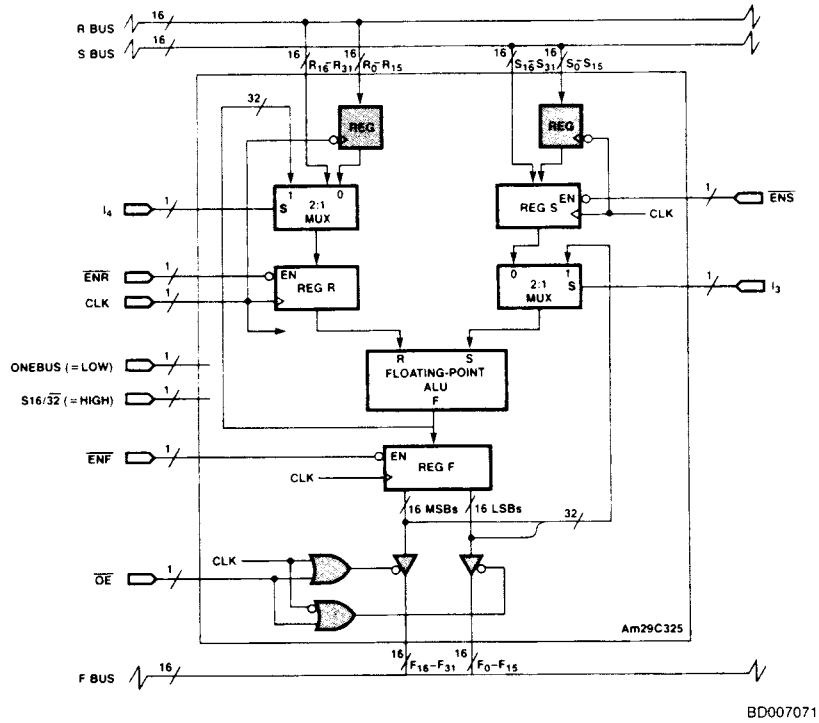


Figure 4. Functional Block Diagram for the 16-Bit, Two-Input Bus Mode

A floating-point number is evaluated or interpreted per the following conventions:

let s = sign bit
e = biased exponent
f = fraction

if e = 0 and f = 0...value = $(-1)^{s*}(0)$ (+ 0, −0)
if e = 0 and f ≠ 0...value = denormalized number
if 0 < e < 255...value = $(-1)^{s*}(2^{e-127})*(1.f)$
(normalized number)
if e = 255 and f = 0...value = $(-1)^{s*}(\infty)$ (+ ∞, −∞)
if e = 255 and f ≠ 0...value = not-a-number (NAN)

Zero: The value zero can have either a positive or negative sign. Rules for determining the sign of a zero produced by an operation are given in the **Sign Bit** section.

Denormalized Number: A denormalized number represents a quantity with magnitude less than 2^{-126} but greater than zero.

Normalized Number: A normalized number represents a quantity with magnitude greater than or equal to 2^{-126} but less than 2^{128}.

Example 1:

The number + 3.5 can be represented in floating-point format as follows:

$$+3.5 = 11.1_2 \times 2^0$$
$$= 1.11_2 \times 2^1$$

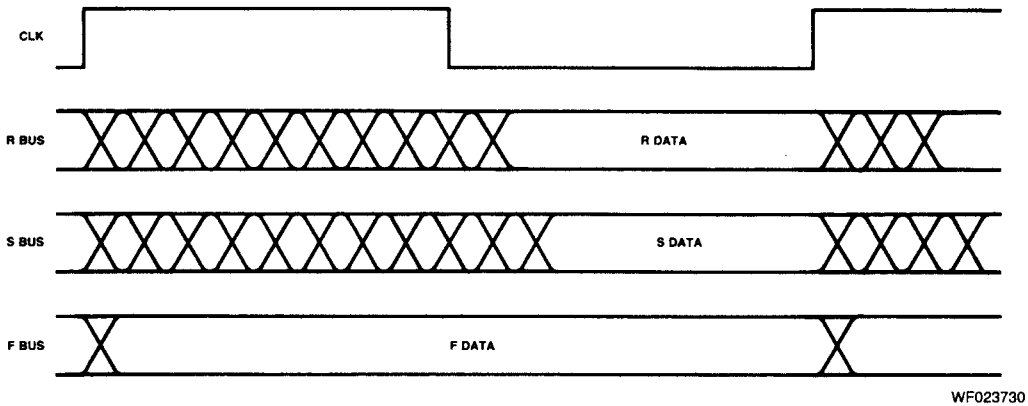
sign = 0

biased exponent = $1_{10} + 127_{10} = 128_{10}$
$= 10000000_2$

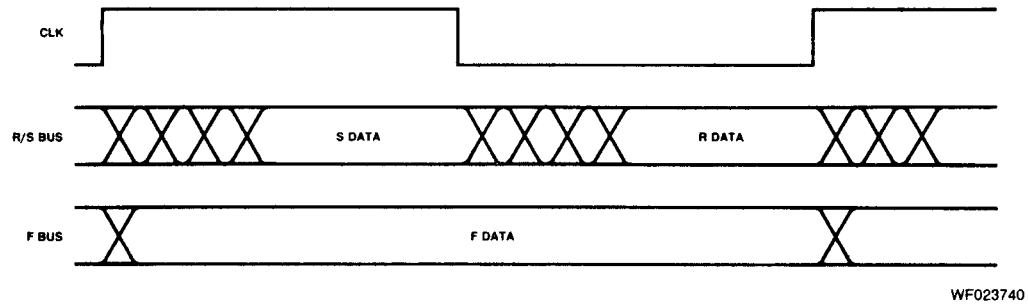
fraction = $11000000000000000000000_2$
(the leading 1 is implied in the format)

Concatenating these fields produces the floating-point word 40600000_{16}.

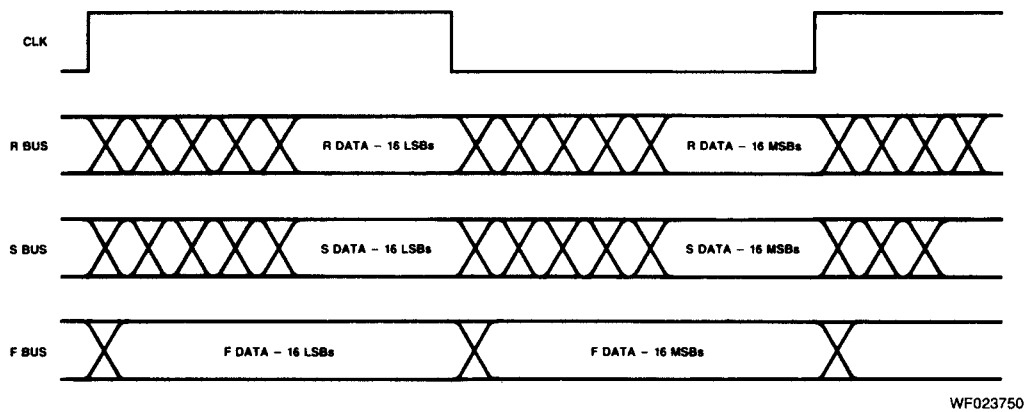
2



a) 32-Bit, Two-Input-Bus Mode

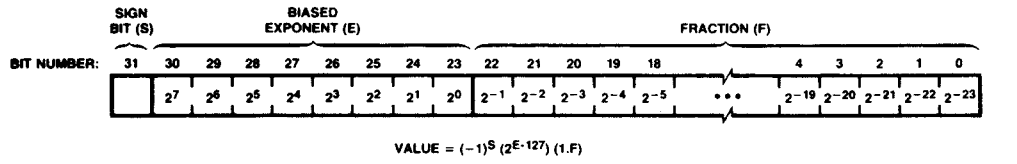


b) 32-Bit, Single-Input-Bus Mode



c) 16-Bit, Two-Input-Bus Mode

Figure 5. Typical Bus Timing for the I/O Modes with FT_0 = LOW, FT_1 = LOW



TB000640

Figure 6. IEEE Mode Single-Precision Floating-Point Format

Example 2:

The number −11.375 can be represented in floating-point format as follows:

$$-11.375 = -1011.011_2 \times 2^0$$
$$= -1.011011_2 \times 2^3$$

sign = 1

biased exponent = $3_{10} + 127_{10} = 130_{10}$
$= 10000010_2$

fraction = $01101100000000000000000_2$
(the leading 1 is implied in the format)

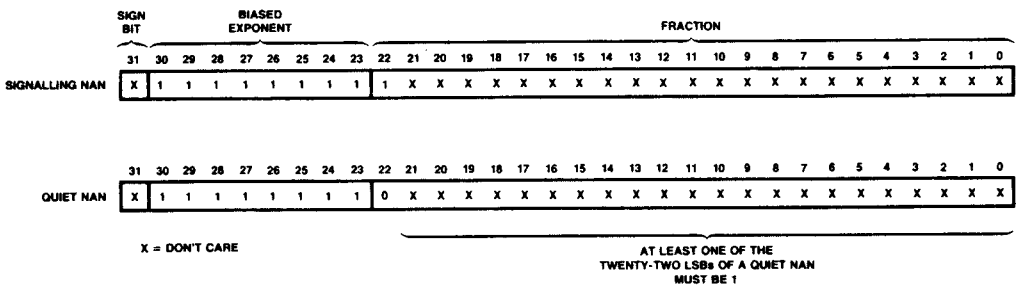
Concatenating these fields produces the floating-point word $C1360000_{16}$.

Infinity: Infinity can have either a positive or negative sign. The way in which infinities are interpreted is determined by the state of the projective/affine mode select, PROJ/$\overline{\text{AFF}}$.

Not-a-Number: A not-a-number, or NAN, does not represent a numeric value but is interpreted as a signal or symbol. NANs are used to indicate invalid operations and as a means of passing process status information through a series of calculations. NANs arise in two ways: 1) they can be generated by the Am29C325 to indicate that an invalid operation has taken place (e.g., ∞ x 0), or 2) be provided by the user as an input operand. There are two types of NANs, signalling and quiet (see Figure 7 for formats).

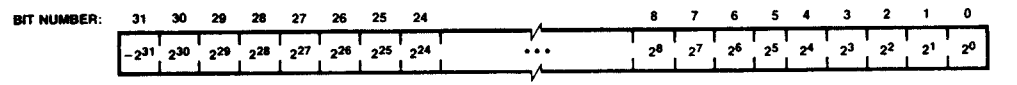
IEEE Mode Integer Format

Integer numbers are represented as 32-bit, two's complement words (Figure 8 depicts the integer format). The integer word can represent a range of integer values from -2^{31} to $2^{31} - 1$.



TB000650

Figure 7. Signalling and Quiet NAN Formats



TB000660

Figure 8. 32-Bit Integer Format

Operations

All eight floating-point ALU operations discussed in the **Functional Description** section can be performed in IEEE mode. Various exceptional aspects of the R PLUS S, R MINUS S, R TIMES S, 2 MINUS S, INT-TO-FP, and FP-TO-INT operations for this mode are described below. The IEEE-TO-DEC and DEC-TO-IEEE operations are discussed separately in the **IEEE-TO-DEC AND DEC-TO-IEEE Operations** section.

Operations with NANs: NANs arise in two ways: 1) they can be generated by the Am29C325 to indicate that an invalid operation has taken place (e.g., ∞ x 0), or 2) be provided by the user as an input operand. There are two types of NANs, signalling and quiet (see Figure 7 for formats).

Signalling NANs set the invalid operation flag when they appear as an input operand to an operation. They are useful for indicating uninitialized variables, or for implementing user-

designed extensions to the operations provided. The ALU never produces a signalling NAN as the final result of an operation.

Quiet NANs are generated for invalid operations. When they appear as an input operand, they are passed through most operations without setting the invalid flag, the floating-point-to-integer conversion operation being the exception.

The sign of any input operand NAN is ignored. All quiet NANs produced as the final result of an operation have a sign of 0.

When a NAN appears as an input operand, the final result of the operation is a quiet NAN that is created by taking the input NAN and forcing bit 22 LOW and bit 21 HIGH. If an operation has two NANs as input operands, the resulting quiet NAN is created using the NAN on the R port.

When a quiet NAN is produced as the final result of an invalid operation whose input operand or operands are not NANs, the resulting NAN will always have the value $7FA00000_{16}$.

The NAN flag will be HIGH whenever an operation produces a NAN as a final result.

Example 1:

Suppose the floating-point addition operation is performed with the following input operands:

R port: $3F800000_{16}$ $(1.0*2^0)$
S port: $7FC12345_{16}$ (signalling NAN)

Result: The signalling NAN on the S port is converted to a quiet NAN by forcing bit 22 LOW and bit 21 HIGH. The operation's final result will be $7FA12345_{16}$. Since one of the two input operands is a signalling NAN, the invalid flag will be HIGH; the NAN flag will also be HIGH.

Example 2:

Suppose the floating-point multiplication operation is performed with the following input operands:

R port: $FFF11111_{16}$ (signalling NAN)
S port: $7FC22222_{16}$ (quiet NAN)

Result: Since both input operands are NANs, the NAN on the R port is chosen for output. In addition to forcing bit 22 LOW, the sign bit (bit 31) is set LOW (bit 21 is already HIGH, and need not be changed). The operation's final result will be $7FB11111_{16}$. Since one of the two input operands is a signalling NAN, the invalid flag is HIGH; the NAN flag will also be HIGH.

Example 3:

Suppose the floating-point subtraction operation is performed with the following input operands:

R port: $FF800001_{16}$ (quiet NAN)
S port: $7F800000_{16}$ $(+\infty)$

Result: To create the final result, the quiet NANs sign bit (bit 31) is forced LOW and bit 21 is forced HIGH (bit 22 is already LOW, and need not be changed). The final result will be $7FA00001_{16}$. The NAN flag will be HIGH.

Operations with Denormalized Numbers: The proposed IEEE standard incorporates denormalized numbers to allow a means of gradual underflow for operations that produce non-zero results too small to be expressed as a normalized floating-point number. The Am29C325 does not support gradual underflow. If a floating-point operation produces a non-zero rounded result that is not large enough to be expressed as a normalized floating-point number, the final result will be a zero of the same sign; the inexact, underflow, and zero flags will be HIGH. If an input operand is a denormalized number, the floating-point ALU will assume that operand to be a zero of the same sign.

Operations Producing Overflows: If an operation has a finite input operand or operands and if the operation produces a rounded result that is too large to fit in the destination format, the operation is said to have overflowed.

A floating-point overflow occurs if an R PLUS S, R MINUS S, R TIMES S, or 2 MINUS S operation with finite input operand(s) produces a result which, after rounding, has a magnitude greater than or equal to 2^{128}. Positive or negative infinity will appear as the final result if the rounded result is positive or negative, respectively, and the overflow and inexact flags will be HIGH.

Integer overflow occurs when the floating-point-to-integer conversion operation attempts to convert a number which, after rounding, is greater than $2^{31} - 1$ or less than -2^{31}. The final result will be quiet NAN $7FA00000_{16}$, and the invalid operation and NAN flags will be HIGH. Note that the overflow and inexact flags remain LOW for integer overflow.

Operations Producing Underflows: If an operation produces a floating-point rounded result having a magnitude too small to be expressed as a normalized floating-point number but greater than zero, that operation is said to have underflowed. Underflow occurs when an R PLUS S, R MINUS S, or R TIMES S operation produces a result which, after rounding, has a magnitude in the range:

$$0 < \text{magnitude} < 2^{-126}.$$

In such cases, the final result will be $+0$ (00000000_{16}) if the rounded result is non-negative and -0 (80000000_{16}) if the rounded result is negative. The underflow, inexact, and zero flags will be HIGH.

Underflow does not occur if the destination format is integer. If the infinitely precise result of a floating-point-to-integer conversion has a magnitude greater than 0 and less than 1 but the rounded result is 0, the underflow flag remains LOW.

Operations with Infinities: In most cases, positive and negative infinity are valid inputs for the R PLUS S, R MINUS S, R TIMES S, and 2 MINUS S operations. Those cases for which infinities are not valid inputs for these operations are listed in Table 4.

Infinities in IEEE mode can be handled either as projective or affine. The projective mode is selected when PROJ/$\overline{\text{AFF}}$ is HIGH; the affine mode is selected when PROJ/$\overline{\text{AFF}}$ is LOW. The only differences between the modes that are relevant to Am29C325 operation occur during the addition and subtraction of infinities:

Operation	Affine Mode	Projective Mode
$(+\infty) + (+\infty)$	Output $+\infty$	Output $7FA00000_{16}$ (quiet NAN), set invalid and NAN flags
$(-\infty) + (-\infty)$	Output $-\infty$	Output $7FA00000_{16}$ (quiet NAN), set invalid and NAN flags
$(+\infty) - (-\infty)$	Output $+\infty$	Output $7FA00000_{16}$ (quiet NAN), set invalid and NAN flags
$(-\infty) - (+\infty)$	Output $-\infty$	Output $7FA00000_{16}$ (quiet NAN), set invalid and NAN flags

If an R PLUS S, R MINUS S, or 2 MINUS S operation has infinity as an input operand or operands, the final result, if valid, is presumed to be exact. For example, adding $+\infty$ and 2.0 will produce a final result of $+\infty$; since the result is considered exact, the inexact flag remains LOW.

Invalid Operations: If an input operand is invalid for the operation to be performed, that operation is considered invalid. When an invalid operation is performed, the floating-point ALU produces a quiet NAN as the final result and the invalid operation flag goes HIGH. Table 4 lists the cases for which the invalid flag is HIGH in IEEE mode and the final results produced for these operations.

TABLE 4. IEEE MODE INVALID OPERATIONS

Operation	Input Operand	Final Result
R PLUS S	$(+\infty) + (-\infty)$ or $(-\infty) + (+\infty)$	$7FA00000_{16}$ (quiet NAN)
R PLUS S	$(+\infty) + (+\infty)$ or $(-\infty) + (-\infty)$ (Note 1)	$7FA00000_{16}$ (quiet NAN)
R MINUS S	$(+\infty) - (+\infty)$ or $(-\infty) - (-\infty)$	$7FA00000_{16}$ (quiet NAN)
R MINUS S	$(+\infty) - (-\infty)$ or $(-\infty) - (+\infty)$ (Note 1)	$7FA00000_{16}$ (quiet NAN)
R TIMES S	$(+0) * (+\infty)$ or $(+0) * (-\infty)$ or $(-0) * (+\infty)$ or $(-0) * (-\infty)$	$7FA00000_{16}$ (quiet NAN)
R PLUS S R MINUS S R TIMES S	R or S is a signalling NAN	(Note 2)
2 MINUS S	S is a signalling NAN	(Note 2)
FP-TO-INT	R is a signalling or quiet NAN	(Note 2)
FP-TO-INT	$R > 2^{31} - 1$ or $R < -(2^{31})$	$7FA00000_{16}$ (quiet NAN)

Notes: 1. These cases are invalid in projective mode only.
2. Results for these operations are described in the **Operations with NANs** section.

The Sign Bit

For most floating-point operations, the sign bit of the final result is unambiguous; i.e., there is only one sign bit value that yields a numerically correct result. Operations that produce an infinitely precise result of zero, however, present a problem, as the IEEE floating-point format allows for representation of both +0 and −0. The following rules can be used to determine the signs of zero produced in such cases.

R PLUS S: The operations $+x + (-x)$ and $-x + (+x)$ produce a final result of zero; the sign of the zero is dependent on the rounding mode:

Rounding Mode	Sign of Final Result
Round to nearest	0
Round toward $-\infty$	1
Round toward $+\infty$	0
Round toward 0	0

Operations $+0 + (-0)$ and $-0 + (+0)$ produce a result of 0, with the sign of the result determined by the table above.

The operation $+0 + (+0)$ produces a final result of $+0$; the operation $-0 + (-0)$ produces a final result of -0.

R MINUS S: The operations $+x - (+x)$ and $-x - (-x)$ produce a final result of zero; the sign of the zero is dependent on the rounding mode:

Rounding Mode	Sign of Result
Round to nearest	0
Round toward $-\infty$	1
Round toward $+\infty$	0
Round toward 0	0

Operations $+0 - (+0)$ and $-0 - (-0)$ produce a result of 0, with the sign of the result determined by the table above.

The operation $+0 - (-0)$ produces a final result of $+0$; the operation $-0 - (+0)$ produces a final result of -0.

R TIMES S: The sign of any multiplication result other than a NAN is the exclusive OR of the signs of the input operands. Therefore, if x is non-negative,
$+0$ times $+x$ produces a final result of $+0$,
$+0$ times $-x$ produces a final result of -0,
-0 times $+x$ produces a final result of -0,
-0 times $-x$ produces a final result of $+0$.

2 MINUS S: If S equals 2, the final result is -0 for the round toward $-\infty$ mode and $+0$ for all other rounding modes.

Rounding

Rounding is performed whenever an operation produces an infinitely precise result that cannot be represented exactly in the destination format. For example, suppose a floating-point operation produces the infinitely precise result:

$1.10101010101010101010101\backslash 01 \times 2^3$.

In this example, the fraction portion of the mantissa has 25 bits; the IEEE floating-point format can accommodate only 23. The backslash (\) in the mantissa represents the boundary between the first 23 bits of the fraction and any remaining bits. Rounding is the process by which this result is approximated by a representation that fits the destination format.

There are four rounding modes in IEEE mode: 1) round to nearest, 2) round toward $+\infty$, 3) round toward $-\infty$, and 4) round toward 0. The rounding mode is chosen using the rounding mode select lines, RND_0 and RND_1. Table 5 lists the select states needed to obtain the desired rounding mode.

TABLE 5. ROUNDING MODE SELECT

RND_1	RND_0	Rounding Mode
0	0	Round to nearest
0	1	Round toward $-\infty$
1	0	Round toward $+\infty$
1	1	Round toward 0

2

Round to Nearest: In this rounding mode the infinitely precise result of an operation is rounded to the closest representation that fits in the destination format. If the infinitely precise result is exactly halfway between two representations, it is rounded to the representation having an LSB of zero. Rounding is performed both for floating-point and integer destination formats.

Figure 9 illustrates four examples of the round-to-nearest process for operations having a floating-point destination format. The infinitely precise result of an operation is represented by an "X" on the number line; the black dots on the number line indicate those values that can be represented exactly in the floating-point format.

Example 1:

In Figure 9(a), the infinitely precise result of an operation is:

$2^{20} + 2^{-4} + 2^{-5} = 1.00000000000000000000000\backslash 11 \times 2^{20}$

The result is rounded to the closest representable floating-point value,

$2^{20} + 2^{-3} = 1.00000000000000000000001 \times 2^{20}$

Example 2:

In Figure 9(b), the infinitely precise result of an operation is:

$2^{20} - 2^{-4} + 2^{-8} =$
$1.11111111111111111111111\backslash 0001 \times 2^{19}$

This result is rounded to the closest representable floating-point value,

$2^{20} - 2^{-4} = 1.11111111111111111111111 \times 2^{19}$

Example 3:

In Figure 9(c), the infinitely precise result of an operation is:

$-(2^{20} + 2^{-3} + 2^{-4})$
$= -1.00000000000000000000001\backslash 1 \times 2^{20}$

This result is exactly halfway between two representable floating-point values. Accordingly, it is rounded to the closest representation with an LSB of zero, or

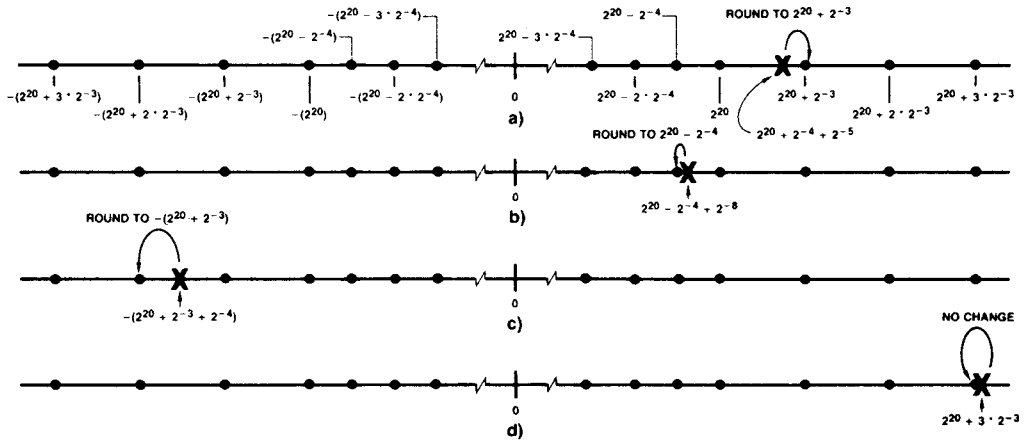
$-(2^{20} + 2*2^{-3}) = -1.00000000000000000000010 \times 2^{20}$

Example 4:

In Figure 9(d), the infinitely precise result of an operation is:

$2^{20} + 3*2^{-3} = 1.00000000000000000000011 \times 2^{20}$

This result can be represented exactly in the floating-point format, and is left unaltered by the rounding process.



AF004550

Figure 9. Floating-Point Rounding Examples for Round-to-Nearest Mode

Figure 10 illustrates four examples of the round-to-nearest process for operations having an integer destination format. The infinitely precise result of an operation is represented by an "X" on the number line; the black dots on the number line indicate those values that can be represented exactly in the integer format.

Example 1:

In Figure 10(a), the infinitely precise result of an operation is:

$2^{10} - 2^{-2} = 00...001111111111.11$

The result is rounded to the closest representable integer value,

$2^{10} = 00...010000000000$

Example 2:

In Figure 10(b), the infinitely precise result of an operation is:

$2^{10} + 2^{0} + 2^{-3} = 00...010000000001.001$

This result is rounded to the closest representable integer value,

$2^{10} + 2^{0} = 00...010000000001$

Example 3:

In Figure 10(c), the infinitely precise result of an operation is:

$-(2^{10} + 2^{0} + 2^{-1}) = -11...101111111110.1$

This result is exactly halfway between two representable integer values. Accordingly, it is rounded to the closest representation with an LSB of zero, or

$-(2^{10} + 2*2^{0}) = 11...101111111110$

Example 4:

In Figure 10(d), the infinitely precise result of an operation is:

$2^{10} + 3*2^{0} = 00...010000000011$

This result can be represented exactly in the integer format and is left unaltered by the rounding process.

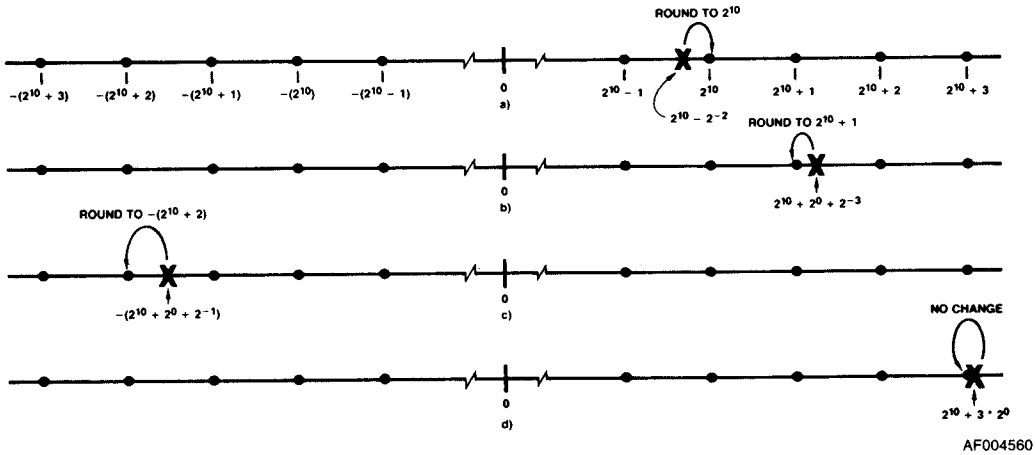


Figure 10. Integer Rounding Examples for Round-to-Nearest Mode

Round Toward $-\infty$: In this rounding mode the result of an operation is rounded to the closest representation that is less than or equal to the infinitely precise result and which fits the destination format. Rounding is performed both for floating-point and integer destination formats.

Figure 11 illustrates four examples of the round toward $-\infty$ process for operations having a floating-point destination format. The infinitely precise result of an operation is represented by an "X" on the number line; the black dots on the number line indicate those values that can be represented exactly in the floating-point format.

Example 1:

In Figure 11(a), the infinitely precise result of an operation is:

$2^{20} + 2^{-4} + 2^{-5} = 1.00000000000000000000000\backslash 11 \times 2^{20}$

This result cannot be represented exactly in floating-point format and is rounded to the next-smaller floating-point representation:

$2^{20} = 1.00000000000000000000000 \times 2^{20}$

Example 2:

In Figure 11(b), the infinitely precise result of an operation is:

$2^{20} - 2^{-4} + 2^{-8} =$
$1.11111111111111111111111\backslash 0001 \times 2^{19}$

This result cannot be represented exactly in floating-point format and is rounded to the next-smaller floating point representation:

$2^{20} - 2^{-4} = 1.11111111111111111111111 \times 2^{19}$

Example 3:

In Figure 11(c), the infinitely precise result of an operation is:

$-(2^{20} + 2^{-3} + 2^{-4}) =$
$-1.00000000000000000000001\backslash 1 \times 2^{20}$

This result cannot be represented exactly in floating-point format and is rounded to the next-smaller floating-point representation.

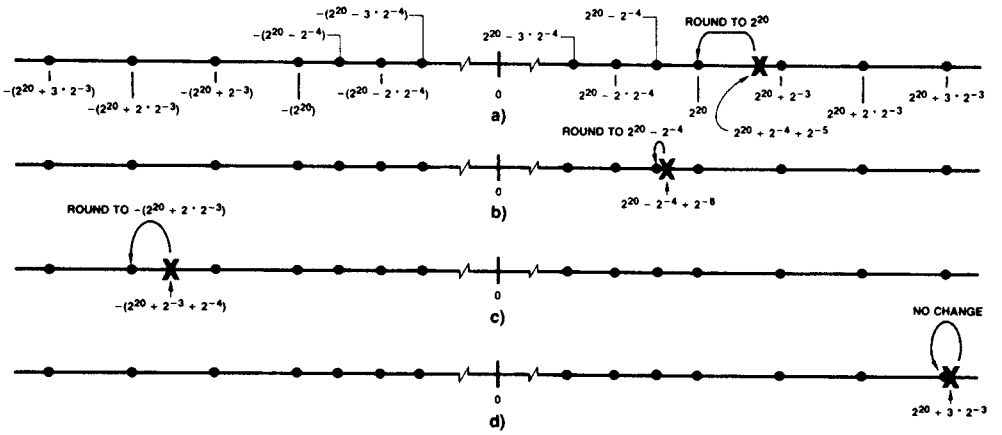
$-(2^{20} + 2^{*}2^{-3}) = -1.00000000000000000000010 \times 2^{20}$

Example 4:

In Figure 11(d), the infinitely precise result of an operation is:

$2^{20} + 3^{*}2^{-3} = 1.00000000000000000000011 \times 2^{20}$

This result can be represented exactly in the floating-point format and is left unaltered by the rounding process.



AF004510

Figure 11. Floating-Point Rounding Examples for Round Toward $-\infty$ Mode

Figure 12 illustrates four examples of the round toward $-\infty$ process for operations having an integer destination format. The infinitely precise result of an operation is represented by an "X" on the number line; the black dots on the number line indicate those values that can be exactly represented in the integer format.

Example 1:

In Figure 12(a), the infinitely precise result of an operation is:

$2^{10} - 2^{-2}$ = 00...001111111111.11

The result is rounded to the next-smaller representable integer value,

$2^{10} - 2^{0}$ = 00...001111111111

Example 2:

In Figure 12(b), the infinitely precise result of an operation is:

$2^{10} + 2^{0} + 2^{-3}$ = 00...010000000001.001

This result is rounded to the next-smaller representable integer value,

$2^{10} + 2^{0}$ = 00...010000000001

Example 3:

In Figure 12(c), the infinitely precise result of an operation is:

$-(2^{10} + 2^{0} + 2^{-1})$ = 11...101111111110.1

This result is rounded to the next-smaller representable integer value:

$-(2^{10} + 2*2^{0})$ = 11...101111111110

Example 4:

In Figure 12(d), the infinitely precise result of an operation is:

$2^{10} + 3*2^{0}$ = 00...010000000011

This result can be represented exactly in the integer format and is unaltered by the rounding process.

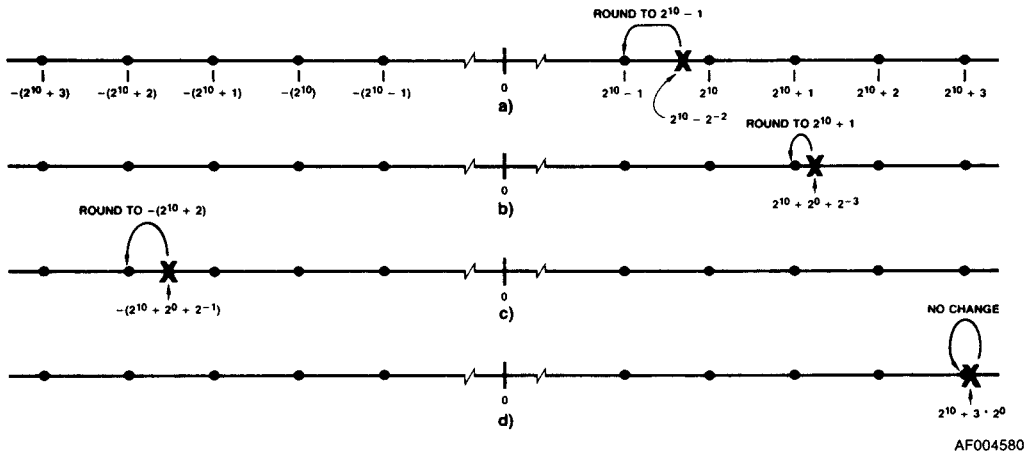


Figure 12. Integer Rounding Examples for Round Toward $-\infty$ Mode

Round Toward +∞: In this rounding mode the result of an operation is rounded to the closest representation that is greater than or equal to the infinitely precise result and which fits the destination format. Rounding is performed both for floating-point and integer destination formats.

Figure 13 illustrates four examples of the round toward +∞ process for operations having a floating-point destination format. The infinitely precise result of an operation is represented by an "X" on the number line; the black dots on the number line indicate those values that can be represented exactly in the floating-point format.

Example 1:

In Figure 13(a), the infinitely precise result of an operation is:

$2^{20} + 2^{-4} + 2^{-5} = 1.00000000000000000000000\backslash 11 \times 2^{20}$

This result cannot be represented exactly in floating-point format and is rounded to the next-larger floating-point representation:

$2^{20} + 2^{-3} = 1.00000000000000000000001 \times 2^{20}$

Example 2:

In Figure 13(b), the infinitely precise result of an operation is:

$2^{20} - 2^{-4} + 2^{-8} =$
$1.11111111111111111111111\backslash 0001 \times 2^{19}$

This result cannot be represented exactly in floating-point format and is rounded to the next-larger floating point representation:

$2^{20} = 1.00000000000000000000000 \times 2^{20}$

Example 3:

In Figure 13(c), the infinitely precise result of an operation is:

$-(2^{20} + 2^{-3} + 2^{-4}) =$
$-1.00000000000000000000001\backslash 1 \times 2^{20}$

This result cannot be represented exactly in floating-point format and is rounded to the next-larger floating-point representation.

$-(2^{20} + 2^{-3}) = -1.00000000000000000000001 \times 2^{20}$

Example 4:

In Figure 13(d), the infinitely precise result of an operation is:

$2^{20} + 3*2^{-3} = 1.00000000000000000000011 \times 2^{20}$.

This result can be represented exactly in the floating-point format — no rounding takes place.

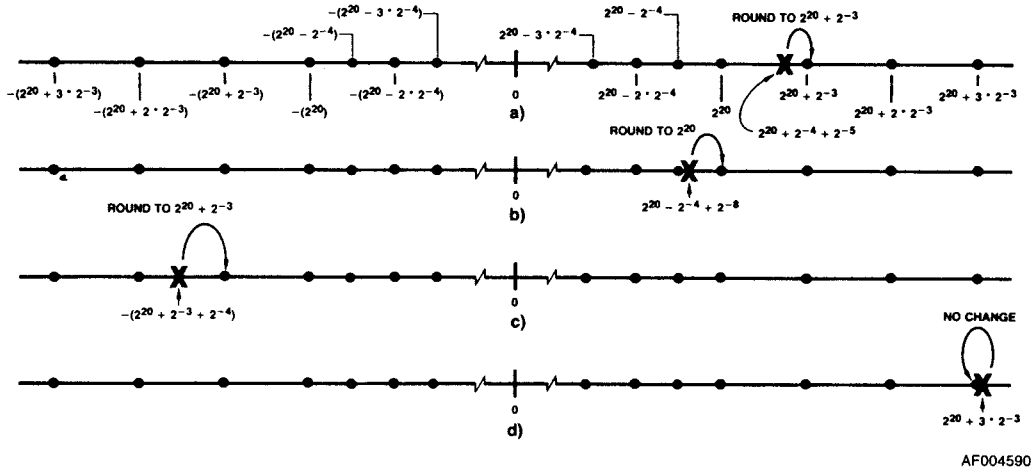


Figure 13. Floating-Point Rounding Examples for Round Toward +∞ Mode

Figure 14 illustrates four examples of the round toward $+\infty$ process for having an integer destination format. The infinitely precise result of an operation is represented by an "X" on the number line; the black dots on the number line indicate those values that can be exactly represented in the integer format.

Example 1:

In Figure 14(a), the infinitely precise result of an operation is:

$2^{10} - 2^{-2} = 00...001111111111.11$

The result is rounded to the next-larger representable integer value,

$2^{10} = 00...010000000000$

Example 2:

In Figure 14(b), the infinitely precise result of an operation is:

$2^{10} + 2^{0} + 2^{-3} = 00...010000000001.001$

This result is rounded to the next-larger representable integer value,

$2^{10} + 2*2^{0} = 00...010000000010$

Example 3:

In Figure 14(c), the infinitely precise result of an operation is:

$-(2^{10} + 2^{0} + 2^{-1}) = 11.101111111110.1$

This result is rounded to the next-larger representable integer value:

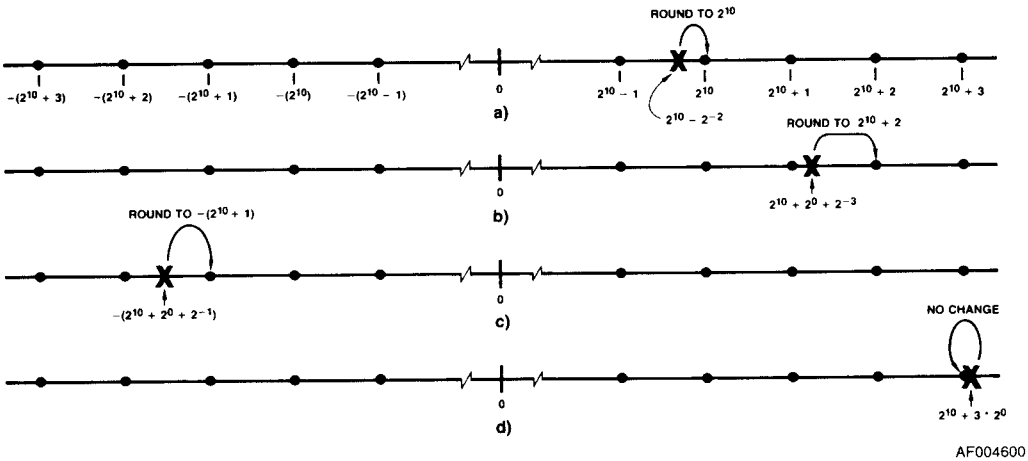
$-(2^{10} + 2^{0}) = 11...1011111111110$

Example 4:

In Figure 14(d), the infinitely precise result of an operation is:

$2^{10} + 3*2^{0} = 00...010000000011$

This result can be represented exactly in the integer format — no rounding takes place.



AF004600

Figure 14. Integer Rounding Examples for Round Toward $+\infty$ Mode

Round Toward 0: In this rounding mode the result of an operation is rounded to the closest representation whose magnitude is less than or equal to the infinitely precise result and which fits the destination format. Rounding is performed both for floating-point and integer destination formats.

Figure 15 illustrates four examples of the round toward 0 process for operations having a floating-point destination format. The infinitely precise result of an operation is represented by an "X" on the number line; the black dots on the number line indicate those values that can be represented exactly in the floating-point format.

Example 1:

In Figure 15(a), the infinitely precise result of an operation is:

$2^{20} + 2^{-4} + 2^{-5} =$
$1.00000000000000000000000\backslash 11 \times 2^{20}$

This result cannot be represented exactly in floating-point format and is rounded to:

$2^{20} = 1.00000000000000000000000 \times 2^{20}$

Example 2:

In Figure 15(b), the infinitely precise result of an operation is:

$2^{20} - 2^{-4} + 2^{-8} =$
$1.11111111111111111111111\backslash 001 \times 2^{19}$

This result cannot be represented exactly in floating-point format and is rounded to:

$2^{20} - 2^{-4} = 1.11111111111111111111111 \times 2^{19}$

Example 3:

In Figure 15(c), the infinitely precise result of an operation is:

$-(2^{20} + 2^{-3} + 2^{-4}) =$
$-1.00000000000000000000001\backslash 1 \times 2^{20}$

This result cannot be represented exactly in floating-point format and is rounded to:

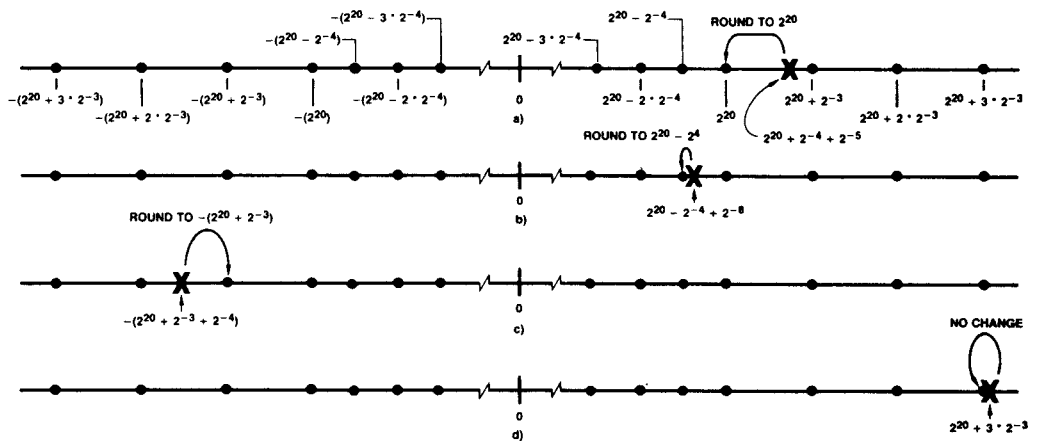
$-(2^{20} + 2^{-3}) = -1.00000000000000000000001 \times 2^{20}$

Example 4:

In Figure 15(d), the infinitely precise result of an operation is:

$2^{20} + 3*2^{-3} = 1.00000000000000000000011 \times 2^{20}$

This result can be represented exactly in the floating-point format and is unaffected by the rounding process.



AF004610

Figure 15. Floating-Point Rounding Examples for Round Toward 0 Mode

Figure 16 illustrates four examples of the round toward 0 process for operations having an integer destination format. The infinitely precise result of an operation is represented by an "X" on the number line; the black dots on the number line indicate those values that can be exactly represented in the integer format.

Example 1:

In Figure 16(a), the infinitely precise result of an operation is:

$2^{10} - 2^{-2}$ = 00...001111111111.11

The result is rounded to:

$2^{10} - 2^{0}$ = 00...001111111111

Example 2:

In Figure 16(b), the infinitely precise result of an operation is:

$2^{10} + 2^{0} + 2^{-3}$ = 00...010000000001.001

The result is rounded to:

$2^{10} + 2^{0}$ = 00...010000000001

Example 3:

In Figure 16(c), the infinitely precise result of an operation is:

$-(2^{10} + 2^{0} + 2^{-1})$ = 11...101111111110.1

The result is rounded to:

$-(2^{10} + 2^{0})$ = 11...101111111111

Example 4:

In Figure 16(d), the infinitely precise result of an operation is:

$2^{10} + 3*2^{0}$ = 00...010000000011

This result can be represented exactly in the integer format and is unaffected by the rounding process.

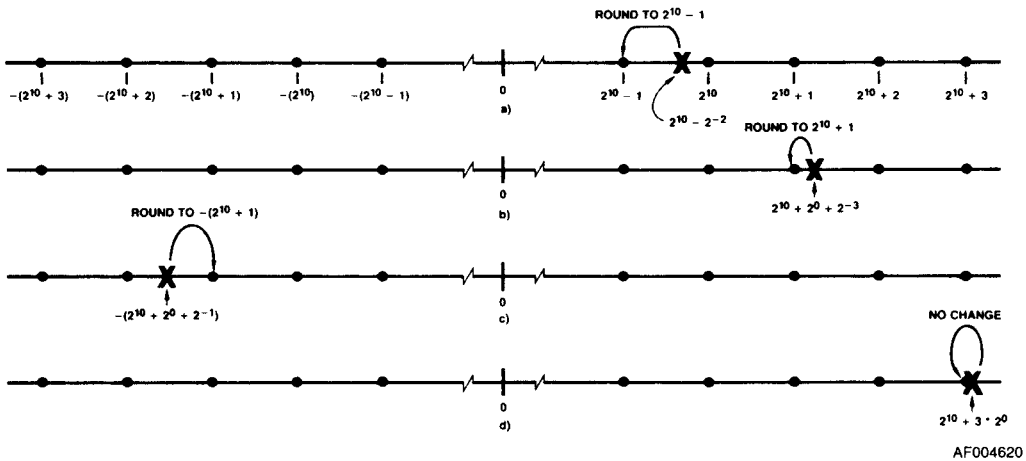


Figure 16. Integer Rounding Examples for Round Toward 0 Mode

Flag Operation

The Am29C325 generates six status flags to monitor floating-point processor operation. The following is a summary of flag conventions in IEEE mode:

Invalid Operation Flag: The invalid operation flag is HIGH when an input operand is invalid for the operation to be performed. Table 4 lists the cases for which the invalid operation flag is HIGH in IEEE mode and the corresponding final result. In cases where the invalid operation flag is HIGH, the overflow, underflow, zero, and inexact flags are LOW; the NAN flag will be HIGH.

Overflow Flag: The overflow flag is HIGH if an R PLUS S, R MINUS S, R TIMES S, or 2 MINUS S operation with finite input operand(s) produces a result which, after rounding, has a magnitude greater than or equal to 2^{128}. The final result will be $+\infty$ or $-\infty$.

Underflow Flag: The underflow flag is HIGH if an R PLUS S, R MINUS S, or R TIMES S operation produces a result which, after rounding, has a magnitude in the range:
$0 <$ magnitude $< 2^{-126}$.

The final result will be +0 (00000000_{16}) if the rounded result is non-negative and −0 (80000000_{16}) if the rounded result is negative.

Inexact Flag: The inexact flag is HIGH if the final result of an R PLUS S, R MINUS S, R TIMES S, 2 MINUS S, INT-TO-FP, or FP-TO-INT operation is not equal to the infinitely precise result. Note that if the underflow or overflow flag is HIGH, the inexact flag will also be HIGH.

Zero Flag: The zero flag is HIGH if the final result of an operation is zero. For operations producing an IEEE floating-point number, the flag accompanies outputs +0 (00000000_{16}) and −0 (80000000_{16}). For operations producing an integer, the flag accompanies the output 0 (00000000_{16}).

NAN Flag: The NAN flag is HIGH if an R PLUS S, R MINUS S, R TIMES S, 2 MINUS S, or FP-TO-INT operation produces a NAN as a final result.

Operation in DEC Mode

When input signal IEEE/$\overline{\text{DEC}}$ is LOW, the DEC mode of operation is selected. In this mode the Am29C325 uses the single-precision floating-point format (floating F) set forth in

Digital Equipment Corporation's VAX Architecture Manual. In addition, the DEC mode complies with most other aspects of single-precision floating-point operation outlined in the manual — differences are discussed in **Appendix B**.

DEC Floating-Point Format

The DEC single-precision floating-point word is 32 bits wide and is arranged in the format shown in Figure 17. 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 indicates the sign of the floating-point number's value. Non-negative values have a sign of 0, negative values a sign of 1.

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

The fraction is a 23-bit unsigned fractional field containing the 23 LSBs of the floating-point number's 24-bit mantissa. The weight of this field's MSB is 2^{-2}; the weight of the LSB is 2^{-24}.

A floating-point number is evaluated or interpreted per the following conventions:
let s = sign bit
e = biased exponent
f = fraction

if e = 0 and s = 0...value = 0
if e = 0 and s = 1...value = DEC-reserved operand
if $0 < e \leqslant 255$...value = $(-1)^s * (2^{e-128}) * (.1f)$
(normalized number)

Zero: The value zero always has a sign of zero.

DEC-Reserved Operand: A DEC-reserved operand does not represent a numeric value but 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.

Normalized Number: A normalized number represents a quantity with magnitude greater than or equal to 2^{-128} but less than 2^{127}.

Example 1:

The number +3.5 can be represented in floating-point format as follows:

$+3.5 = 11.1_2 \times 2^0$
$= .111_2 \times 2^2$

sign = 0

biased exponent = $2_{10} + 128_{10} = 130_{10}$
$= 10000010_2$

fraction = $11000000000000000000000_2$
(the leading 1 is implied in the format)

Concatenating these fields produces the floating-point word 41600000_{16}.

Example 2:

The number −11.375 can be represented in floating-point format as follows:

$-11.375 = -1011.011_2 \times 2^0$
$= -.1011011_2 \times 2^4$

sign = 1

biased exponent = $4_{10} + 128_{10} = 132_{10}$
$= 10000100_2$

fraction = $01101100000000000000000_2$
(the leading 1 is implied in the format)

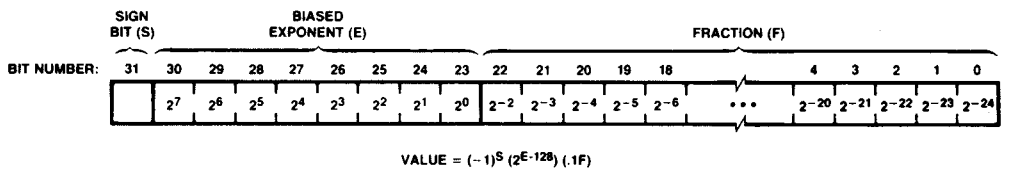
Concatenating these fields produces the floating-point word $C2360000_{16}$.

DEC Mode Integer Format

DEC mode integer format is identical to that of the IEEE mode. Integer numbers are represented as 32-bit, two's complement words (Figure 8 depicts the integer format). The integer word can represent a range of integer values from -2^{31} to $2^{31} - 1$.

Operations

All eight floating-point ALU operations discussed in the **General Description** section can be performed in DEC mode.



TB000671

Figure 17. DEC-Mode Floating-Point Format

Various exceptional aspects of the R PLUS S, R MINUS S, R TIMES S, 2 MINUS S, INT-TO-FP, and FP-TO-INT operations for this mode are described below. The IEEE-TO-DEC and DEC-TO-IEEE operations are discussed separately in the **IEEE-TO-DEC and DEC-TO-IEEE Operations** section.

Operations with DEC-Reserved Operands: DEC-reserved operands arise in two ways: 1) they can be generated by the Am29325 to indicate that an invalid operation or floating-point overflow has taken place, or 2) be provided by the user as an input operand.

When a DEC-reserved operand appears as an input operand, the final result of the operation is the same DEC-reserved operand. If an operation has two DEC-reserved operands as inputs, the DEC-reserved operand on the R port becomes the final result.

The NAN flag will be HIGH whenever an operation produces a DEC-reserved operand as a final result.

Example 1:

Suppose the floating-point addition operation is performed with the following input operands:

R port: 40800000_{16} $(0.1*2^1)$
S port: 80012345_{16} (DEC-reserved operand)

Result: This operation produces the DEC-reserved operand on the S port, 80012345_{16}, as the final result. The NAN flag will be HIGH.

Example 2:

Suppose the floating-point multiplication operation is performed with the following input operands:

R port: 80765432_{16} (DEC-reserved operand)
S port: 80000001_{16} (DEC-reserved operand)

Result: Since both input operands are DEC-reserved operands, the operand on the R port, 80765432_{16}, is the final result of the operation. The NAN flag will be HIGH.

Operations Producing Overflows: If an operation produces a rounded result that is too large to fit in the the destination format, that operation is said to have overflowed.

A floating-point overflow occurs if an R PLUS S, R MINUS S, R TIMES S, or 2 MINUS S operation with finite input operand(s) produces a result which, after rounding, has a magnitude greater than or equal to 2^{127}. The final result in such cases will be DEC-reserved operand 80000000_{16}; the overflow, inexact, and NAN flags will be HIGH.

Integer overflow occurs when the "floating-point-to-integer" conversion operation attempts to convert to integer a floating-point number which, after rounding, is greater than $2^{31} - 1$ or less than -2^{31}. The final result in such cases will be DEC-reserved operand 80000000_{16}; the invalid operation flag will be HIGH. Note that the overflow and inexact flags remain LOW for integer overflow.

Operations Producing Underflows: If an operation produces a floating-point result which, after rounding, has a magnitude too small to be expressed as a normalized floating-point number but greater than 0, that operation is said to have underflowed. Underflow occurs when an R PLUS S, R MINUS S, or R TIMES S operation produces a result which, after rounding, has the magnitude:

$$0 < \text{magnitude} < 2^{-128}.$$

The final result in such cases will be 0 (00000000_{16}). The underflow, inexact, and zero flags will be HIGH.

Underflow does not occur if the destination format is integer. If the infinitely precise result of a floating-point-to-integer conversion has a magnitude greater than 0 and less than 1 but the rounded result is 0, the underflow flag remains LOW.

Invalid Operations: If an input operand is invalid for the operation to be performed, that operation is considered invalid. There is only one invalid operation in DEC mode: performing a floating-point-to-integer conversion on a value too large to be converted to an integer. In this case, the final result will be DEC-reserved operand 80000000_{16}, and the invalid operation and NAN flags will be HIGH.

Sign Bit

For all operations producing a DEC floating-point result, the sign bit of the final result is unambiguous; i.e., there is only one sign bit value that yields a numerically correct result.

Rounding

There are four rounding modes for DEC operation: 1) round to nearest, 2) round toward $+\infty$, 3) round toward $-\infty$, and 4) round toward 0. The round toward $+\infty$, round toward $-\infty$, and round toward 0 modes are performed in a manner identical to that for IEEE operation; refer to the **Rounding** section under **Operation in IEEE Mode**. The round to nearest mode is similar to that for IEEE operation but differs in one respect: for the case in which the infinitely precise result of an operation is exactly halfway between two representable values, DEC round to nearest mode rounds to the value with the larger magnitude, rather than to the value whose LSB is 0.

Flag Operation

The Am29C325 generates six status flags to monitor floating-point processor operation. The following is a summary of flag operation in DEC mode:

Invalid Operation Flag: The invalid operation flag is HIGH if the FP-TO-INT operation is performed on a floating-point number too large to be converted to an integer. The final result for such an operation will be the DEC-reserved operand 80000000_{16}.

Overflow Flag: The overflow flag is HIGH if an R PLUS S, R MINUS S, R TIMES S, or 2 MINUS S operation produces a result which, after rounding, has a magnitude greater than or equal to 2^{127}. The final result will be the DEC-reserved operand 80000000_{16}.

Underflow Flag: The underflow flag is HIGH if an R PLUS S, R MINUS S, or R TIMES S operation produces a result which, after rounding, has a magnitude in the range:

$$0 < \text{magnitude} < 2^{-128}.$$

The final result will be 0 (00000000_{16}) in such cases.

Inexact Flag: The inexact flag is HIGH if the final result of an R PLUS S, R MINUS S, R TIMES S, 2 MINUS S, INT-TO-FP, or FP-TO-INT operation is not equal to the infinitely precise result. Note that if the underflow or overflow flag is HIGH, the inexact flag will also be HIGH.

Zero Flag: The zero flag is HIGH if the final result of an operation is 0. For operations producing an integer or a DEC floating-point number, the flag accompanies the output 0 (00000000_{16}). (It should be noted that any operation producing a floating-point 0 in DEC mode will output 00000000_{16}.)

NAN Flag: The NAN flag is HIGH if an R PLUS S, R MINUS S, R TIMES S, 2 MINUS S, or FP-TO-INT operation produces a DEC-reserved operand as the final result.

IEEE-TO-DEC and DEC-TO-IEEE Operations

The IEEE-TO-DEC and DEC-TO-IEEE operations are used to convert floating-point numbers between the IEEE and DEC formats. Both operations work in a manner independent of the IEEE/$\overline{\text{DEC}}$ mode control.

IEEE-TO-DEC Conversion

The operation converts an IEEE floating-point number to DEC floating-point format. Most conversions are exact; in no case does the round mode have any effect on the final result. There are, however, a few exceptional cases:

a) If the IEEE floating-point input has a magnitude greater than or equal to 2^{127}, it is too large to be represented by a DEC floating-point number. The final result will be the DEC-reserved operand 80000000_{16}; the overflow, inexact, and NAN flags will be HIGH.

b) If the IEEE floating-point input is a NAN, the final result will be the DEC-reserved operand 80000000_{16}; the invalid and NAN flags will be HIGH.

c) If the IEEE floating-point input is a denormalized number, the final result will be a DEC 0 (00000000_{16}); the zero flag will be HIGH.

d) If the IEEE floating-point input is + 0 or −0, the final result will be a DEC 0 (00000000_{16}); the zero flag will be HIGH.

DEC-TO-IEEE Conversion

This operation converts a DEC floating-point number to IEEE floating-point format. Most conversions are exact; in no case does the round mode have any effect on the final result. There are, however, a few exceptional cases:

a) If the DEC floating-point input is not 0 but has a magnitude less than 2^{-126}, it is too small to be expressed as a normalized IEEE floating-point number. The final result will be an IEEE floating-point 0 having the same sign as the input (00000000_{16} for positive inputs and 80000000_{16} for negative inputs); the underflow, inexact, and zero flags will be HIGH.

b) If the DEC floating-point input is a DEC-reserved operand, the result will be quiet NAN $7FA00000_{16}$; the invalid operation and NAN flags will be HIGH.

c) If the DEC floating-point input is 0, the final result will be IEEE floating-point + 0 (00000000_{16}); the zero flag will be HIGH.

APPLICATIONS

Suggestions for Power and Ground Pin Connections

The Am29C325 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 GNDO pins carry output driver switching currents and can be electrically noisy. The V_{CC} and GND 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 GND and GNDO 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 18. By physically separating the V_{CC} and V_{CCO} pins, coupled noise will be reduced.

3. All GND and GNDO pins should be connected directly to the ground plane.

4. The V_{CCO} pins should be decoupled to ground with a 0.1-μF ceramic capacitor and a 10-μF electrolytic capacitor, placed as closely to the Am29C325 as is practical. V_{CC} pins should be decoupled to ground in a similar manner.

A suggested layout is shown in Figure 18.

(Bottom View)

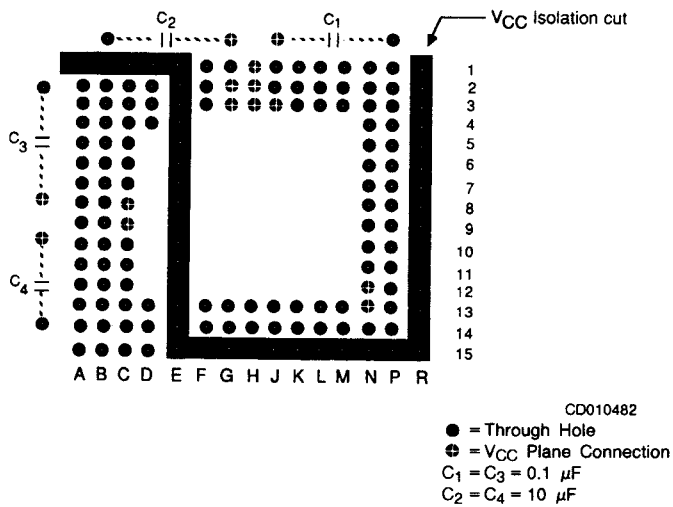


Figure 18. Suggested Printed-Circuit Board Layout (Power and Ground Connections)

ABSOLUTE MAXIMUM RATINGS

Storage Temperature ..... −65 to +150°C
Ambient Temperature Under Bias ..... −55 to +125°C
Supply Voltage to Ground Potential
Continuous ..... −0.3 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 to V_{CC} + 0.3 V
DC Output Current, into LOW Outputs ..... 30 mA
DC Input Current ..... −10 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
Temperature, Case (T_A) ..... 0 to +70°C
Supply Voltage (V_{CC}) ..... +4.75 to +5.25 V

Military* (M) Devices
Temperature (T_A) ..... −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.

Thermal Resistance (Typical)

Symbol	CGX145	Unit
θ_{JA}	23	°C/W

*Military product 100% tested at T_A = +25°C, +125°C, and −55°C.

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

Parameter Symbol	Parameter Description	Test Conditions (Note 1)			Min.	Max.	Unit
V_{OH}	Output HIGH Voltage	V_{CC} = Min. V_{IN} = V_{IL} or V_{IH}	I_{OH} = 0.4 mA		2.4		V
V_{OL}	Ouput LOW Voltage	V_{CC} = Min. V_{IN} = V_{IL} or V_{IH}	I_{OL} = 4.0 mA			0.5	V
V_{IH}	Guaranteed Input Logical HIGH Voltage (Note 2)				2.0		V
V_{IL}	Guaranteed Input Logical LOW Voltage (Note 2)					0.8	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 Power Supply Current (Note 3)	V_{CC} = Max. I_O = 0 μA	COM'L T_A = 0 to +70°C	CMOS V_{IN} = V_{CC} or GND		20	mA
				TTL V_{IN} = 0.5 V or 2.4 V		20	
			MIL T_A = −55 to +125°C	CMOS V_{IN} = V_{CC} or GND		25	
				TTL V_{IN} = 0.5 V or 2.4 V		25	
C_{PD}	Power Dissipation Capacitance (Notes 4,5)	V_{CC} = MAX No Load				9,000	pF

Notes: 1. V_{CC} conditions shown as Min. or Max. refer to the commercial and military V_{CC} limits.
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 I_{CC} when the device is driven by CMOS circuits and TTL I_{CC} when the device is driven by TTL circuits.
4. C_{PD} determines the dynamic current consumption:

$$I_{CC}\ (\text{Total}) = I_{CC}\ (\text{Static}) + (C_{PD} + nCL)V_{CC}\frac{f}{2},$$

where f is the clock frequency, CL output load capacitance, and n number of loads.

5. Tested on a sample basis.

CAPACITANCE

C_{IN}	Input Capacitance	fc = 1 MHz (Note 5)		12	pF
C_{OUT}	Output Capacitance			12	pF

SWITCHING CHARACTERISTICS over COMMERCIAL operating range

No.	Parameter Symbol	Parameter Description		Test Conditions	29C325		29C325-1		29C325-2		Unit
					Min.	Max.	Min.	Max.	Min.	Max.	
1	t_{ASC}	Clocked Add, Subtract Time (R PLUS S, R MINUS S, 2 MINUS S)				124		109		87	ns
2	t_{MC}	Clocked Multiply Time (R TIMES S)				107		97		78	ns
3	t_{CC}	Clocked Conversion Time (INT-TO-FP, FP-TO-INT, IEEE-TO-DEC, DEC-TO-IEEE)				105		94		75	ns
4	t_{ASUC}	Unclocked Add, Subtract Time (R, S to F, Flags) for R PLUS S, R MINUS S,and 2 MINUS S Instructions				146		135		108	ns
5	t_{MUC}	Unclocked Multiply Time (R, S to F, Flags) for R TIMES S Instruction		FT_0 = HIGH FT_1 = HIGH		154		142		114	ns
6	t_{CUC}	Unclocked Conversion Time (R, S to F, Flags) for INT-TO-FP, FP-TO-INT, IEEE- TO-DEC and DEC-TO-IEEE Instructions				133		122		98	ns
7	t_{PWH}	Clock Pulse Width HIGH			15		15		15		ns
8	t_{PWL}	Clock Pulse Width LOW			15		15		15		ns
9	t_{PDOF1}	Clock to $F_0 - F_{31}$ and Flag Outputs		FT_0 = LOW FT_1 = HIGH		145		130		104	ns
10	t_{PDOF2}			FT_1 = LOW		26		24		22	ns
11	t_{PZL}	OE Enable Time	Z to LOW			22		22		20	ns
12	t_{PZH}		Z to HIGH			22		22		20	ns
13	t_{PLZ}	OE Disable Time	LOW to Z			13		13		12	ns
14	t_{PHZ}		HIGH to Z			13		13		12	ns
15	t_{PZLLSB}	Clock ↑ to $F_0 - F_{15}$ Enable, 16-Bit I/O Mode	Z to LOW	S16/32 = HIGH ONEBUS = LOW		26		26		24	ns
16	t_{PZHLSB}		Z to HIGH			26		26		24	ns
17	t_{PLZLSB}	Clock ↓ to $F_0 - F_{15}$ Disable, 16-Bit I/O Mode	LOW to Z			22		22		20	ns
18	t_{PHZLSB}		HIGH TO Z			22		22		20	ns
19	t_{PZLMSB}	Clock ↓ to $F_{16} - F_{31}$ Enable, 16-Bit I/O Mode	Z to LOW	S16/32 = HIGH ONEBUS = LOW		26		26		24	ns
20	t_{PZHMSB}		Z to HIGH			26		26		24	ns
21	t_{PLZMSB}	Clock ↑ to $F_{16} - F_{31}$ Disable,16-Bit I/O Mode	LOW to Z			22		22		20	ns
22	t_{PHZMSB}		HIGH to Z			22		22		20	ns
23	t_{SCE}	Register Clock Enable Setup Time		FT_0 = LOW FT_1 = LOW	10		9		8		ns
24	t_{HCE}	Register Clock Enable Hold Time		FT_0 = LOW FT_1 = LOW	3		3		3		ns
25	t_{SD1}	$R_0 - R_{31}$, $S_0 - S_{31}$ Setup Time (see note below)		FT_0 = LOW	20		16		14		ns
26	t_{HD1}	$R_0 - R_{31}$, $S_0 - S_{31}$ Hold Time (see note below)			5		5		4		ns
27	t_{SD2}	$R_0 - R_{31}$, $S_0 - S_{31}$ Setup Time (see note below)		FT_0 = HIGH FT_1 = LOW	133		118		94		ns
28	t_{HD2}	$R_0 - R_{31}$, $S_0 - S_{31}$ Hold Time (see note below)			0		0		0		ns
29	t_{SI02}	$I_0 - I_2$ Instruction Select Setup Time		FT for Destination Register = LOW	132		114		91		ns
30	t_{HI02}	$I_0 - I_2$ Instruction Select Hold Time			1		1		1		ns
31	t_{PDI02}	$I_0 - I_2$ Instruction Select to $F_0 - F_{31}$, Flags		FT_1 = HIGH		150		134		107	ns
32	t_{SI3}	I_3 Port S Input Select Setup Time		FT_1 = LOW	89		78		68		ns
33	t_{HI3}	I_3 Port S Input Select Hold Time			0		0		0		ns
34	t_{SI4}	I_4 Register R Input Select Setup Time (see note below)		FT_0 = LOW	18		13		11		ns
35	t_{HI4}	I_4 Register R Input Select Hold Time (see note below)			3		3		3		ns
36	t_{SRM}	Round Mode Select Setup Time		FT for Destination Register = LOW	67		57		46		ns
37	t_{HRM}	Round Mode Select Hold Time			3		3		3		ns
38	t_{PRF}	Round Mode Select to $F_0 - F_{31}$, Flags		FT_1 = HIGH		78		69		55	ns

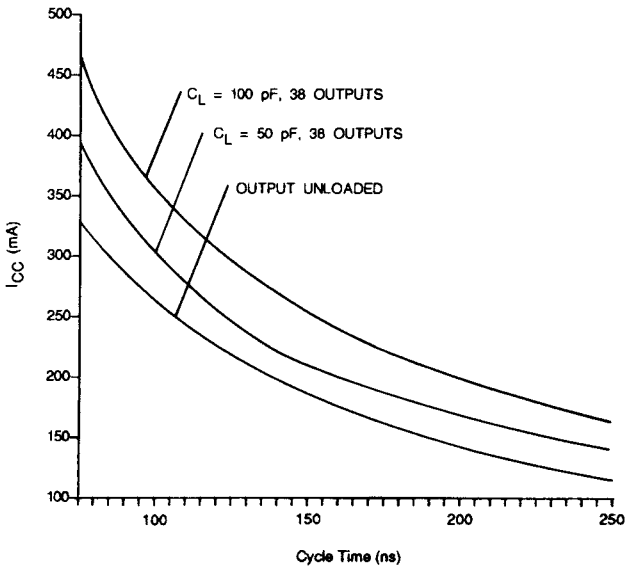
Note: See timing diagram for desired mode of operation to determine clock edge to which these setup and hold times apply.

SWITCHING CHARACTERISTICS over **MILITARY** operating range (for APL Products, Group A, Subgroups 9, 10, 11 are tested unless otherwise noted)

No.	Parameter Symbol	Parameter Description		Test Conditions	29C325 Min.	29C325 Max.	29C325-1 Min.	29C325-1 Max.	Unit
1	t_{ASC}	Clocked Add, Subtract Time (R PLUS S, R MINUS S, 2 MINUS S)				149		131	ns
2	t_{MC}	Clocked Multiply Time (R TIMES S)				133		117	ns
3	t_{CC}	Clocked Conversion Time (INT-TO-FP, FP-TO-INT, IEEE-TO-DEC, DEC-TO-IEEE)				130		114	ns
4	t_{ASUC}	Unclocked Add, Subtract Time (R, S to F, Flags) for R PLUS S, R MINUS S, and 2 MINUS S Instructions		FT_0 = HIGH FT_1 = HIGH		178		163	ns
5	t_{MUC}	Unclocked Multiply Time (R, S to F, Flags) for R TIMES S Instruction				190		179	ns
6	t_{CUC}	Unclocked Conversion Time (R, S to F, Flags) for INT-TO-FP, FP-TO-INT, IEEE-TO-DEC and DEC-TO-IEEE Instructions				161		150	ns
7	t_{PWH}	Clock Pulse Width HIGH			15		15		ns
8	t_{PWL}	Clock Pulse Width LOW			15		15		ns
9	t_{PDOF1}	Clock to $F_0 - F_{31}$ and Flag Outputs		FT_0 = LOW FT_1 = HIGH		166		156	ns
10	t_{PDOF2}			FT_1 = LOW		35		33	ns
11	t_{PZL}	$\overline{OE}$ Enable Time	Z to LOW			29		29	ns
12	t_{PZH}		Z to HIGH			29		29	ns
13	t_{PLZ}	$\overline{OE}$ Disable Time	LOW to Z			16		16	ns
14	t_{PHZ}		HIGH to Z			16		16	ns
15	t_{PZLLSB}	Clock ↑ to $F_0 - F_{15}$ Enable, 16-Bit I/O Mode	Z to LOW	S16/$\overline{32}$ = HIGH ONEBUS = LOW		31		31	ns
16	t_{PZHLSB}		Z to HIGH			31		31	ns
17	t_{PLZLSB}	Clock ↓ to $F_0 - F_{15}$ Disable, 16-Bit I/O Mode	LOW to Z			23		23	ns
18	t_{PHZLSB}		HIGH TO Z			23		23	ns
19	t_{PZLMSB}	Clock ↓ to $F_{16} - F_{31}$ Enable, 16-Bit I/O Mode	Z to LOW	S16/$\overline{32}$ = HIGH ONEBUS = LOW		31		31	ns
20	t_{PZHMSB}		Z to HIGH			31		31	ns
21	t_{PLZMSB}	Clock ↑ to $F_{16} - F_{31}$ Disable,16-Bit I/O Mode	LOW to Z			23		23	ns
22	t_{PHZMSB}		HIGH to Z			23		23	ns
23	t_{SCE}	Register Clock Enable Setup Time		FT_0 = LOW FT_1 = LOW	11		10		ns
24	t_{HCE}	Register Clock Enable Hold Time		FT_0 = LOW FT_1 = LOW	4		4		ns
25	t_{SD1}	$R_0 - R_{31}$, $S_0 - S_{31}$ Setup Time (see note below)		FT_0 = LOW	21		16		ns
26	t_{HD1}	$R_0 - R_{31}$, $S_0 - S_{31}$ Hold Time (see note below)			6		6		ns
27	t_{SD2}	$R_0 - R_{31}$, $S_0 - S_{31}$ Setup Time (see note below)		FT_0 = HIGH FT_1 = LOW	163		148		ns
28	t_{HD2}	$R_0 - R_{31}$, $S_0 - S_{31}$ Hold Time (see note below)			0		0		ns
29	t_{SI02}	$I_0 - I_2$ Instruction Select Setup Time		FT for Destination Register = LOW	167		143		ns
30	t_{HI02}	$I_0 - I_2$ Instruction Select Hold Time			2		2		ns
31	t_{PDI02}	$I_0 - I_2$ Instruction Select to $F_0 - F_{31}$, Flags		FT_1 = HIGH		186		166	ns
32	t_{SI3}	I_3 Port S Input Select Setup Time		FT_1 = LOW	109		94		ns
33	t_{HI3}	I_3 Port S Input Select Hold Time			0		0		ns
34	t_{SI4}	I_4 Register R Input Select Setup Time (see note below)		FT_0 = LOW	18		13		ns
35	t_{HI4}	I_4 Register R Input Select Hold Time (see note below)			5		5		ns
36	t_{SRM}	Round Mode Select Setup Time		FT for Destination Register = LOW	85		71		ns
37	t_{HRM}	Round Mode Select Hold Time			5		5		ns
38	t_{PRF}	Round Mode Select to $F_0 - F_{31}$, Flags		FT_1 = HIGH		102		95	ns

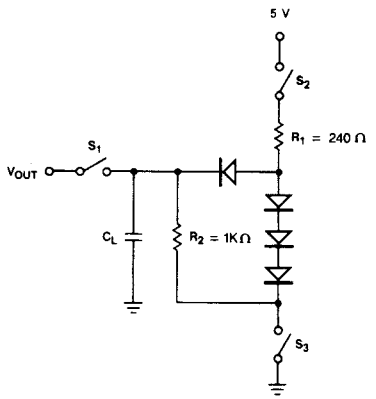
Note: See timing diagram for desired mode of operation to determind clock edge to which these setup and hold times apply.

Am29C325 I_{CC} vs Cycle Time



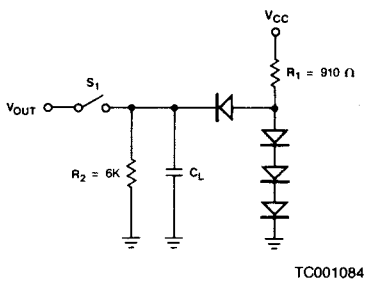
OP002770

SWITCHING TEST CIRCUITS



TC004810

A. Three-State Outputs



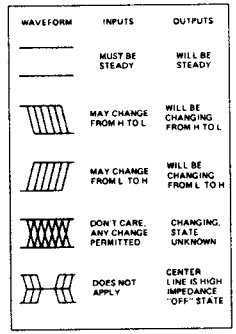
TC001084

B. Normal Outputs

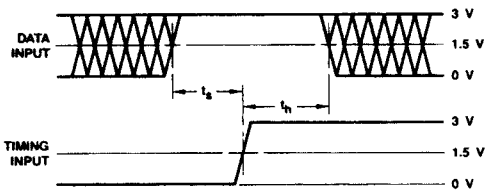
Notes: 1. C_L = 50 pF includes scope probe, wiring, and stray capacitances without device in test fixture.
2. S_1, S_2, S_3 are closed during function tests and all AC tests except output enable tests.
3. S_1 and S_3 are closed while S_2 is open for t_{PZH} test.
S_1 and S_2 are closed while S_3 is open for t_{PZL} test.
4. C_L = 5.0 pF for output disable tests.

SWITCHING TEST WAVEFORMS

KEY TO SWITCHING WAVEFORMS



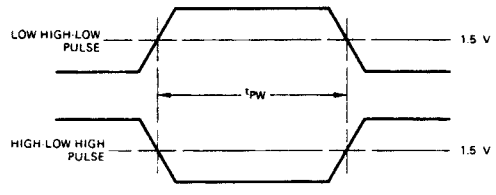
KS000010



WFR02970

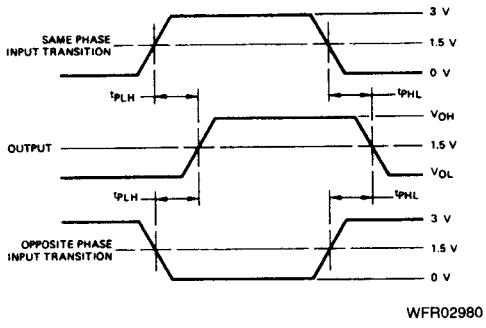
Notes: 1. Diagram shown for HIGH data only.
Output transition may be opposite sense.
2. Cross hatched area is don't care condition.

Set-Up, Hold, and Release Times



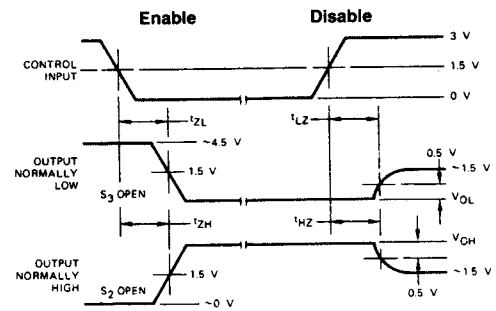
WFR02790

Pulse Width



WFR02980

Propagation Delay



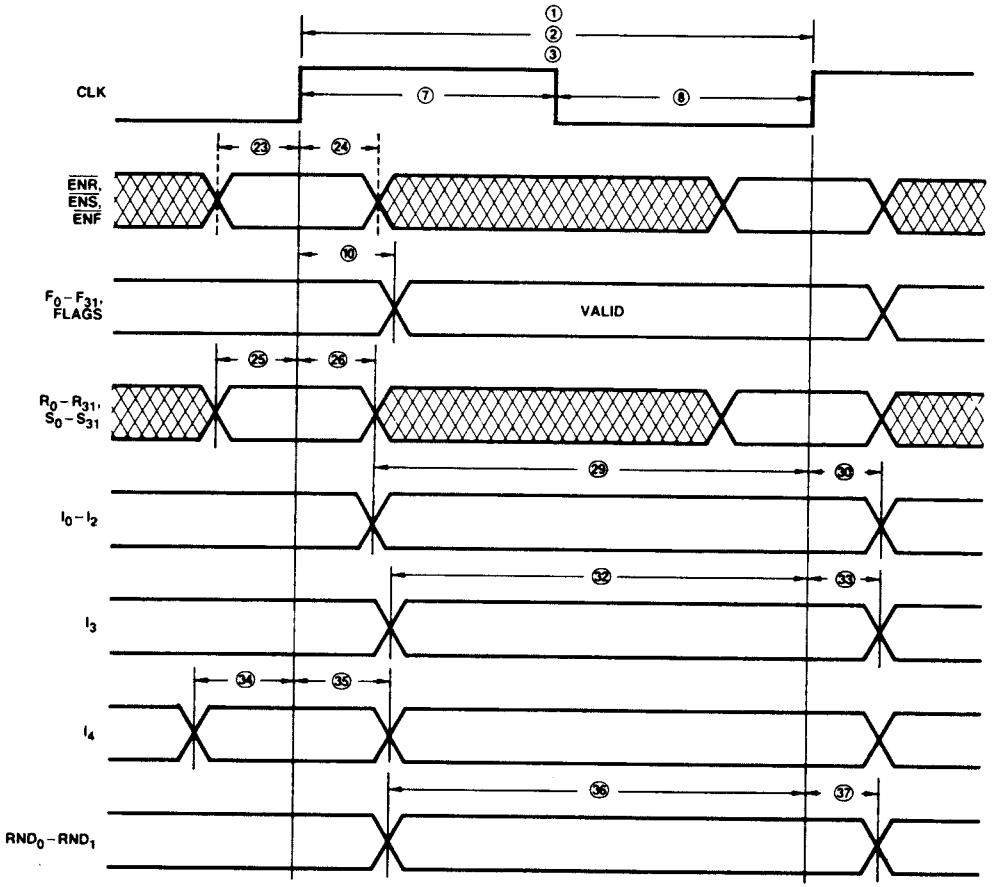
WFR02660

Notes: 1. Diagram shown for Input Control Enable-LOW and Input Control Disable-HIGH.
2. S_1, S_2 and S_3 of Load Circuit are closed except where shown.

Enable and Disable Times

2

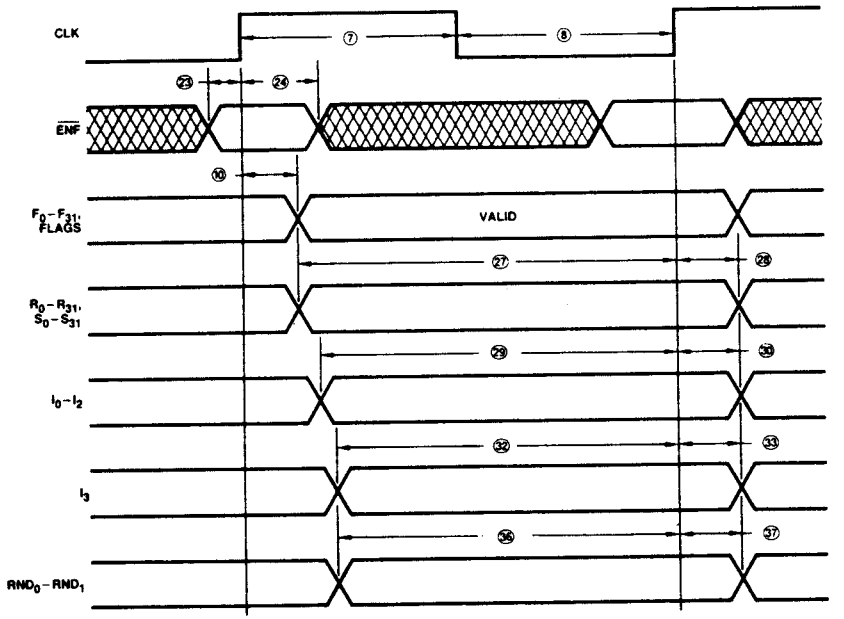
SWITCHING WAVEFORMS



WF023760

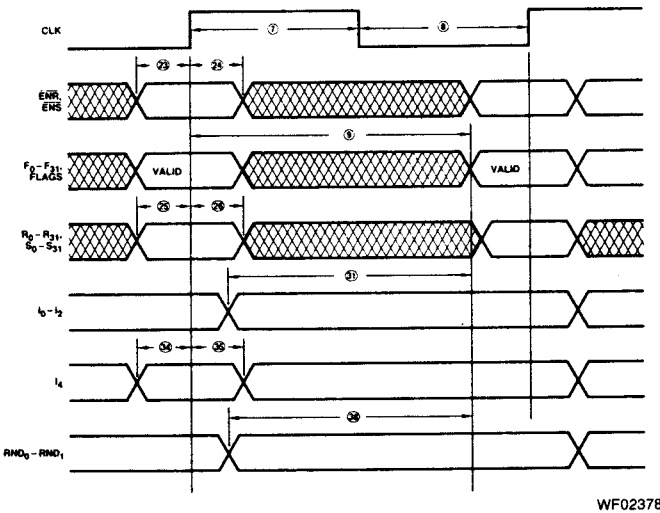
Clocked Operation: FT_0 = LOW
FT_1 = LOW

SWITCHING WAVEFORMS (Cont'd.)



WF023770

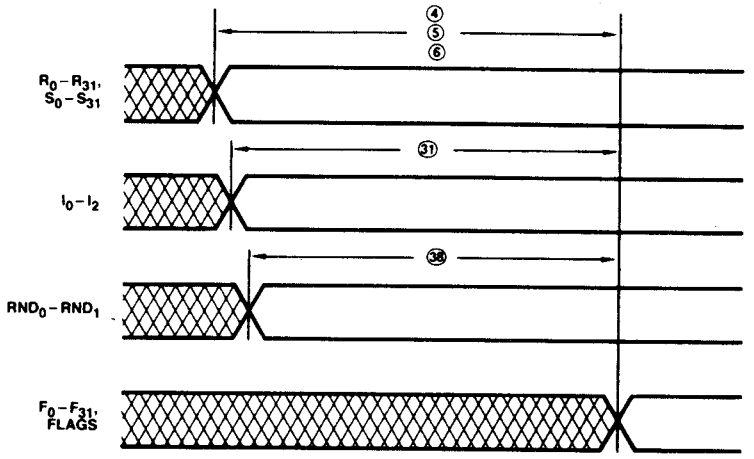
Clocked Operation: FT_0 = HIGH
FT_1 = LOW



WF023780

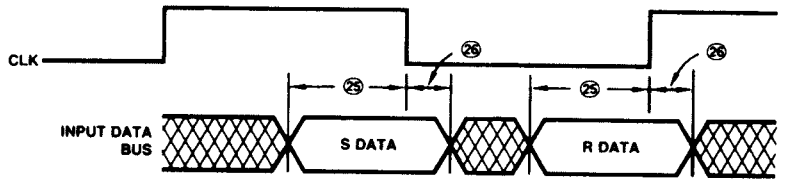
Clocked Operation: FT_0 = LOW
FT_1 = HIGH

SWITCHING WAVEFORMS (Cont'd.)



WF023790

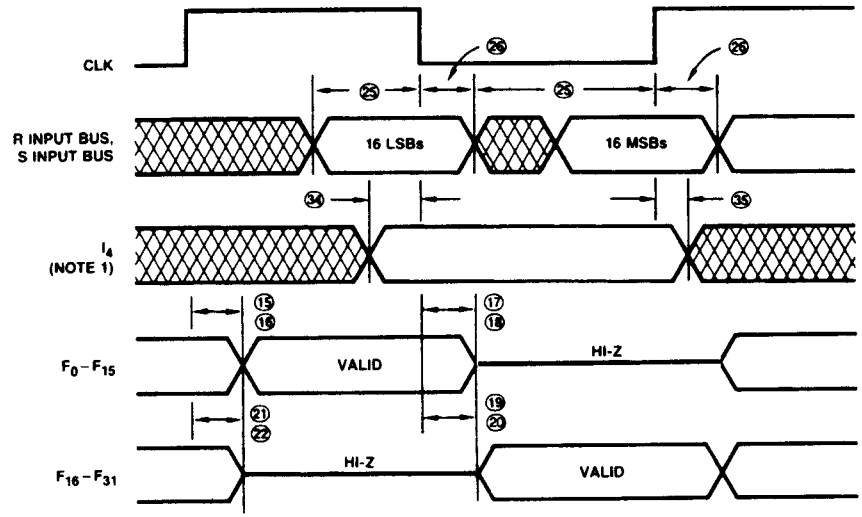
Flow-Through Operation (FT_0 = HIGH, FT_1 = HIGH)



WF023800

32-Bit, Single-Input Bus Mode

SWITCHING WAVEFORMS (Cont'd.)



WF023811

Note 1. 1_4 has special setup and hold time requirements in this mode. All other control signals have timing requirements as shown in the diagram "Clocked operation, FT_0 = LOW, FT_1 = LOW."

16-Bit, Two-Input Bus Mode

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 $V_{IL} \leqslant 0$ V and $V_{IH} \geqslant 3.0$ 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

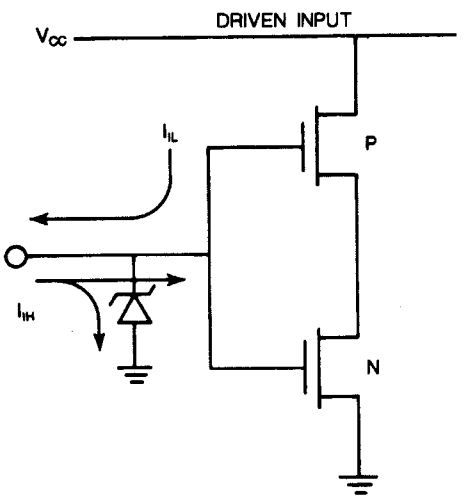
   The noise associated with automatic testing (due to the long, inductive cables) and the high gain of the tested device when in the vicinity of the actual device threshold, frequently give rise to oscillations when testing high speed circuits. These oscillations are not indicative of a reject device, 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

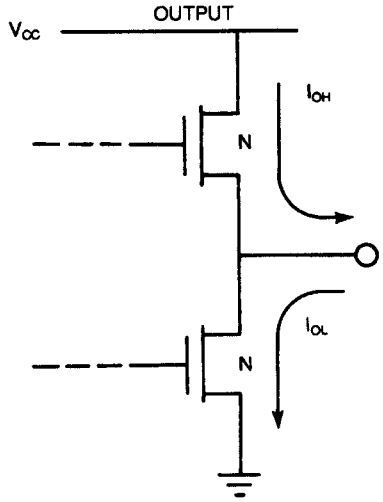
   Occasionally, parameters are specified that cannot be measured directly on automatic testers because of tester limitations. Data input hold times often fall into this category. In these cases, the parameter in question is guaranteed by correlating these tests with other AC tests that have been performed. 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.

INPUT/OUTPUT CIRCUIT DIAGRAMS



IC000865



IC000870

2

APPENDIX A

DIFFERENCES BETWEEN THE IEEE PROPOSED STANDARD FOR BINARY FLOATING-POINT ARITHMETIC AND THE Am29C325'S IEEE MODE

When operated in IEEE mode, the Am29C325 high-speed floating-point processor complies with the single-precision portion of the IEEE Proposed Standard for Binary Floating-Point Arithmetic (P754, draft 10.0) in most respects. There are, however, several differences:

Denormalized Numbers

The Am29C325 does not handle denormalized numbers. A denormalized input will be converted to zero of the same sign before the specified operation takes place. The operation proceeds in exactly the same manner as if the input were +0 or −0, producing the same numerical result and flags.

If the result of an operation, after rounding, has a magnitude smaller than 2^{-126}, the result is replaced by a zero of the same sign.

Representation of Overflows

In some rounding modes the proposed IEEE standard requires that overflows be represented as the format's most-positive or most-negative finite number. In particular:

When rounding toward 0, all overflows should produce a result of the largest representable finite number with the sign of the intermediate result.

When rounding toward $-\infty$, all positive overflows should produce a result of the largest representable positive finite number.

When rounding toward $+\infty$, all negative overflows should produce a result of the largest representable negative finite number.

The Am29C325, however, always represents positive overflows as $+\infty$ and negative overflows as $-\infty$, regardless of rounding mode.

Projective Mode

The proposed IEEE standard provides only for an affine mode to control the handling of infinities. The Am29C325 provides both affine and projective modes; the desired mode can be selected by the user.

Traps

The proposed IEEE standard stipulates that the user be able to request a trap on any exception. The Am29C325 does not support trapped operation and behaves as if traps are disabled.

Resetting of Flags

The proposed IEEE standard states that once an exception flag has been set, it is reset only at the user's request. The Am29C325's flags, however, reflect the status of the most recent operation.

Generation of the Underflow Flag

The proposed IEEE standard suggests several possible criteria for determining if underflow occurs. These criteria generate underflow flags that differ in subtle ways. The underflow criteria chosen for the Am29C325 stipulate that underflow occurs if:

a) the rounded result of an operation has a magnitude in the range:

$$0 < \text{magnitude} < 2^{-126},$$

and

b) the final result is not equal to the infinitely precise result.

Since the Am29C325 never produces a denormalized number as the final result of a calculation, condition (b) is true whenever (a) is true. Note then that the operation of the Am29C325's underflow flag is somewhat different than that of an "IEEE standard" system using the same underflow criteria. For example, if an operation should produce an infinitely precise result that is exactly 2^{-127}, an "IEEE standard" system would produce that value as the final result, expressed as a denormalized number. Since that system's final result is exact, the underflow flag would remain LOW. The Am29C325, on the other hand, would output zero; since its final result is not exact, the underflow flag would be HIGH.

APPENDIX B

DIFFERENCES BETWEEN DEC VAX AND Am29C325 DEC MODE

Operation in DEC mode complies with most aspects of single-precision floating-point operation outlined in the Digital Equipment Corporation's VAX Architecture Manual. However, there are some differences that should be noted:

Format

The Am29C325's DEC format is:

sign	– bit 31
exponent	– bits 30 – 23
mantissa	– 22 – 0

The VAX format is:

sign	– bit 15
exponent	– 14 – 7
mantissa	– bits 6 – 0, bits 31 – 16

In both cases, fields are listed from MSB to LSB, with bit 31 the MSB of the 32-bit word. The Am29C325's DEC format can be converted to VAX format by swapping the 16 LSBs and 16 MSBs of the 32-bit word.

Flags vs. Exceptions

In DEC VAX operation, certain unusual conditions arising during system operation may incur an exception or an indication to the operating system that special handling is needed.

The VAX recognizes a number of arithmetic exceptions. The following exceptions are relevant to the operations supported by the Am29C325:

Integer Overflow Trap: indicates that the last operation produced an integer overflow. The LSBs of the correct result are stored in the destination operand.

Floating-Point Overflow Trap/Fault: indicates that the last operation produced, after normalization and rounding, a floating-point number with magnitude greater than or equal to 2^{127}. A trap replaces the destination operand with the DEC-reserved operand 80000000_{16}; a fault leaves the destination operand unchanged.

Floating-Point Underflow Trap/Fault: indicates that the last operation produced, after normalization and rounding, a floating-point number with magnitude less than 2^{-128}. A trap replaces the destination operand with zero; a fault leaves the destination operand unchanged.

Reserved Operand Fault: indicates that the last operation had a reserved operand as an input. The destination operand is unchanged.

The Am29C325 does not directly support DEC traps and faults. Rather, it indicates unusual conditions by setting one or more of the six status flags HIGH. Table D2 describes flag operation in DEC mode.

Integer Overflow

In cases of integer overflow, the VAX signals the integer overflow trap and stores the LSBs of the correct result. The Am29C325 sets the invalid operation flag and outputs the DEC-reserved operand 80000000_{16}.

Floating-Point Underflow/Overflow Operation

The VAX Architecture Manual specifies the action to be taken on the destination operand when floating-point underflow or overflow is encountered. The Am29C325 has no immediate control over this destination operand, as it resides somewhere off-chip, either in a register or memory location. This isn't so much a difference between the VAX specification and Am29C325 operation as it is a difference in scope.

The Am29C325 responds to floating-point underflow by producing a final result of 0 (00000000_{16}); the underflow, inexact, and zero flags will be HIGH. It responds to floating-point overflow by producing the DEC-reserved operand 80000000_{16} as the final result; the overflow, inexact, and NAN flags will be HIGH.

Handling of DEC-Reserved Operands

If an operation has a DEC-reserved operand as an input, the Am29C325 will produce that operand as the final result. If an operation has two input arguments and both are DEC-reserved operands, the operand on port R becomes the final result. For the VAX, operations with a DEC-reserved operand input or inputs do not modify the destination operand. As mentioned above, control of the destination operand is beyond the scope of the Am29C325's operation.

Inexact Flag

The Am29C325 provides an inexact flag to indicate that the final result produced by an operation is not equal to the infinitely precise result. The VAX does not provide this flag.

APPENDIX C

PERFORMING FLOATING-POINT DIVISION ON THE Am29C325

While the Am29C325 does not have a floating-point division instruction, it can be used to evaluate reciprocals. The division:

$$C = A/B$$

can then be performed by evaluating:

$$C = A*(1/B)$$

Only a modest amount of external hardware is needed to implement the reciprocal function.

The technique for calculating reciprocals is based on the Newton-Raphson method for obtaining the roots of an equation. The roots of equation:

$$F(x) = 0$$

can be found by iteratively evaluating the equation:

$$x_{i+1} = x_i - F(x_i)/F'(x_i)$$

The process begins by making a guess as to the value of x_i and using this guess or "seed" value to perform the first iteration. Iterations are continued until the root is evaluated to the desired accuracy. The number of iterations needed to achieve a given accuracy depends both on the accuracy of the seed value and the nature of F(x).

Now consider the equation:

$$F(x) = (1/x) - B$$

The root of F(x) is 1/B. The reciprocal of B, then, can be found by using the Newton-Raphson method to find the root of F(x). The iterative equation for finding the root is:

$$\begin{aligned} x_{i+1} &= x_i - F(x_i)/F'(X_i) \\ &= x_i - (1/x_i - B)/-(x_i)^{-2} \\ &= x_i\,(2 - B*x_i) \end{aligned}$$

It can be shown that, in order for this iterative equation to converge, the seed value x_0 must fall in the range:

$$\begin{aligned} &0 < x_0 < 2/B && \text{if } B > 0 \\ \text{or}\quad &2/B < x_0 < 0 && \text{if } B < 0 \end{aligned}$$

For example, if the reciprocal of 3 is to be evaluated, the seed value must be between 0 and 2/3.

The error of x_i reduces quadratically; that is, if the error of x_i is e, the error is reduced to order e^2 by the next iteration. The number of bits of accuracy in the result, then, roughly doubles after every iteration. While this is only an approximation of the actual error produced, it is a handy rule of thumb for determining the number of iterations needed to produce a result of a certain accuracy, given the accuracy of the seed.

Example 1:

Find the reciprocal of 7.25.

Solution:

The seed value must fall in the range:

$$0 < x_0 < 2/7.25$$
$$\text{or } 0 < x_0 < .275862$$

Suppose x_0 is chosen to be .1:

Iteration 1: $x_1 = x_0\,(2 - B*x_0)$
$= .1(2 - (7.25)\,(.1))$
$= .1275$

Iteration 2: $x_2 = x_1\,(2 - B*x_1)$
$= .1275(2 - (7.25)\,(.1275))$
$= .1371421875$

Iteration 3: $x_3 = x_2\,(2 - B*x_2)$
$= .1371421875*$
$(2 - (7.25)\,(.1371421875))$
$= .1379265230$

The actual value of 1/7.25, to ten decimal places, is .1379310345.

The error after each iteration is:

Iteration	x_i	Error to Ten Places
0	0.1	-0.0379310345
1	0.1275	-0.0104310345
2	0.1371421875	-0.0007888470
3	0.1379265230	-0.0000045115

Example 2:

Find the reciprocal of -0.3.

Solution:

The seed value must fall in the range:

$$2/(-0.3) < x_0 < 0$$
$$\text{or } -6.66 < x_0 < 0$$

Suppose x_0 is chosen to be -2.0:

Iteration 1: $x_1 = x_0\,(2 - B*x_0)$
$= -2.0(2 - (-0.3)\,(-2.0))$
$= -2.8$

Iteration 2: $x_2 = x_1\,(2 - B*x_1)$
$= -2.8(2 - (-0.3)\,(-2.8))$
$= -3.248$

Iteration 3: $x_3 = x_2\,(2 - B*x_2)$
$= -3.248(2-(-0.3)\,(-3.248))$
$= -3.3311488$

Iteration 4: $x_4 = x_3\,(2 - B*x_3)$
$= -3.3311488*$
$(2-(-0.3)\,(-3.3311488))$
$= -3.333331902$

The actual value of 1/(-0.3), to ten decimal places, is -3.333333333.

The error after each iteration is:

i	x_i	Error to Ten Places
0	-2.0	1.333333333
1	-2.8	0.533333333
2	-3.248	0.085333333
3	-3.3311488	0.002184533
4	-3.333331902	0.000001431

In order to implement the Newton-Raphson method on the Am29C325, some means is needed to generate the seed used in the first iteration. One approach is to place a hardware seed look-up table between the R bus and the Am29C325; see Table C1. A more detailed diagram of the look-up table appears in Figure C2.

TABLE C1. CONTENTS OF THE SEED EXPONENT PROM

DEC		IEEE	
Address (16)	**Data (16)**	**Address (16)**	**Data (16)**
000	(Note 1)	100	(Note 1)
001	(Note 1)	101	FC
002	FF	102	FB
003	FE	103	FA
004	FD	104	F9
005	FC	105	F8
006	FB	106	F7
007	FA	107	F6
008	F9	108	F5
009	F8	109	F4
00A	F7	10A	F3
00B	F6	10B	F2
00C	F5	10C	F1
00D	F4	10D	F0
00E	F3	10E	EF
00F	F2	10F	EE
010	F1	110	ED
011	F0	111	EC
012	EF	112	EB
.	.	.	.
.	.	.	.
.	.	.	.
0EE	13	1EE	0F
0EF	12	1EF	0E
0F0	11	1F0	0D
0F1	10	1F1	0C
0F2	0F	1F2	0B
0F3	0E	1F3	0A
0F4	0D	1F4	09
0F5	0C	1F5	08
0F6	0B	1F6	07
0F7	0A	1F7	06
0F8	09	1F8	05
0F9	08	1F9	04
0FA	07	1FA	03
0FB	06	1FB	02
0FC	05	1FC	01
0FD	04	1FD	(Note 2)
0FE	03	1FE	(Note 2)
0FF	02	1FF	(Note 2)

Notes: 1. The reciprocals of these numbers are too large to be represented in the selected format.
2. The reciprocals of these numbers are too small to be represented in normalized IEEE format.

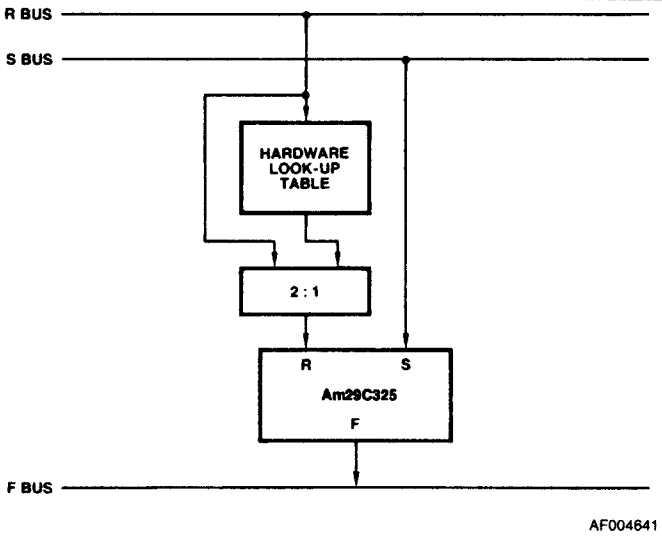


Figure C1. Adding a Hardware Look-Up Table to the Am29C325

The look-up table has two sections: a biased exponent look-up PROM, and a fraction look-up PROM. The seed-biased exponent look-up table is stored in a 512-by-8-bit PROM. This table consists of two sections: the DEC format section (which occupies addresses $000 - 0FF_{16}$), and the IEEE section (which occupies addresses $100 - 1FF_{16}$. The appropriate table will be selected automatically if address line A_8 is wired to the Am29C325's IEEE/$\overline{DEC}$ pin. The equations implemented by these table sections are:

DEC table: seed biased exponent
$= 257_{10}$ – input biased exponent

IEEE table: seed biased exponent
$= 253_{10}$ – input biased exponent

Table C1 lists the contents of this PROM.

The seed fraction look-up table is stored in one or more PROMs, the number of PROMs depending on the desired accuracy of the seed value. The hardware depicted in Figure C2 uses two 4K-by-8-bit PROMs to implement a fraction look-up table whose inputs are the 12 MSBs of the input argument's fraction. These PROMs output the 16 MSBs of the seed's fraction field — the remaining 7 bits of fraction are set to 0. The equation implemented in this table is:

$$\text{seed fraction} = \frac{2}{1 + \text{input fraction}} - 1$$

where the value of the input fraction falls in the range

$$0 \leqslant \text{input fraction} < 1$$

Note that the seed fraction must also be constrained to fall in the range

$$0 \leqslant \text{seed fraction} < 1$$

Therefore, if the input fraction is 0, the corresponding seed fraction stored in the table must be $.111...111_2$, not 1.0_2. The same seed fraction look-up table may be used for both IEEE and DEC formats. Table C2 contains a partial listing for the seed fraction look-up table shown in Figure C2.

TABLE C2. CONTENTS OF THE SEED FRACTION PROMS

Address (16)	Value of Input Fraction (10)	Value of Seed Fraction (10)	PROM Outputs (16)	
			$R_{22}-R_{15}$	$R_{14}-R_7$
000	0.0	0.9999999999 (see text)	FF	FF
001	0.0002441406	0.9995118370	FF	E0
002	0.0004882812	0.9990239150	FF	C0
003	0.0007324219	0.9985362280	FF	A0
004	0.0009765625	0.9980487790	FF	80
005	0.0012207031	0.9975615710	FF	60
006	0.0014648438	0.9970745970	FF	40
007	0.0017089844	0.9965878630	FF	20
008	0.0019531250	0.9961013650	FF	00
009	0.0021972656	0.9956151030	FE	E1
00A	0.0024414063	0.9951290800	FE	C0
00B	0.0026855469	0.9946432920	FE	A1
00C	0.0029296875	0.9941577400	FE	81
.	.	.	.	.
.	.	.	.	.
.	.	.	.	.
FF6	0.9975585938	0.0012221950	00	50
FF7	0.9978027344	0.0010998410	00	48
FF8	0.9980486750	0.0009775170	00	40
FF9	0.9982910156	0.0008552230	00	38
FFA	0.9985351563	0.0007329590	00	30
FFB	0.9987792969	0.0006107240	00	28
FFC	0.9990234375	0.0004885200	00	20
FFD	0.9992675781	0.0003663450	00	18
FFE	0.9995117188	0.0002442000	00	10
FFF	0.9997558594	0.0001220850	00	08

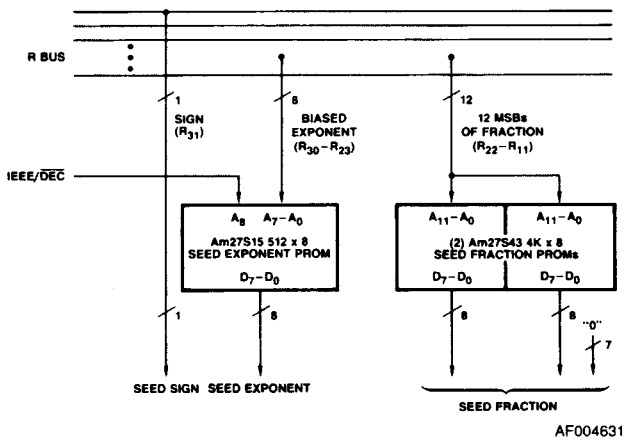


Figure C2. The Hardware Look-Up Table

With the hardware look-up table in place, the reciprocal of value B can be calculated with the following series of operations:

1) Place B on both the R and S buses. The 2 : 1 multiplexer at the output of the hardware look-up table should select the output of the look-up table (see Figure C3-A).

2) Load the seed value x_0 into register R and load B into register S. Select the R TIMES S operation (see Figure C3-B).

3) Load product $B*x_0$ into register F. Select the 2 MINUS S operation, and select register F as the input to the ALU S port (see Figure C3-C).

4) Load $2-B*x_0$ into register F. Select the R TIMES S operation and select register F as the input to the ALU S port (see Figure C3-D).

5) Load the value x_1 ($x_1 = x_0(2-B*x_0)$) into registers R and F. Select the R TIMES S operation (see Figure C3-E).

6) Repeat steps 3 through 5 until the result has the accuracy desired.

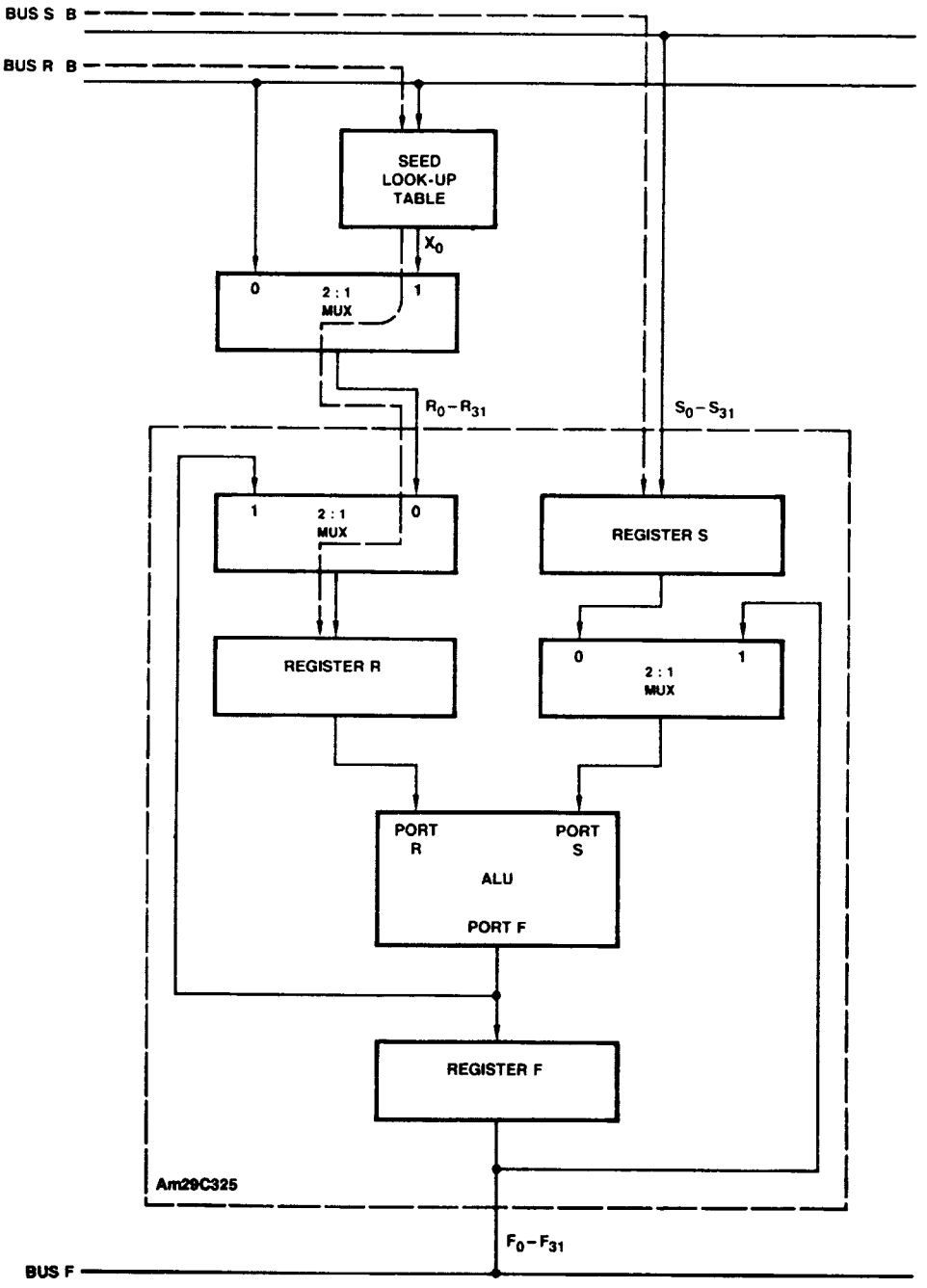


Figure C3-A. Data Flow for Step 1 of the Reciprocal Procedure

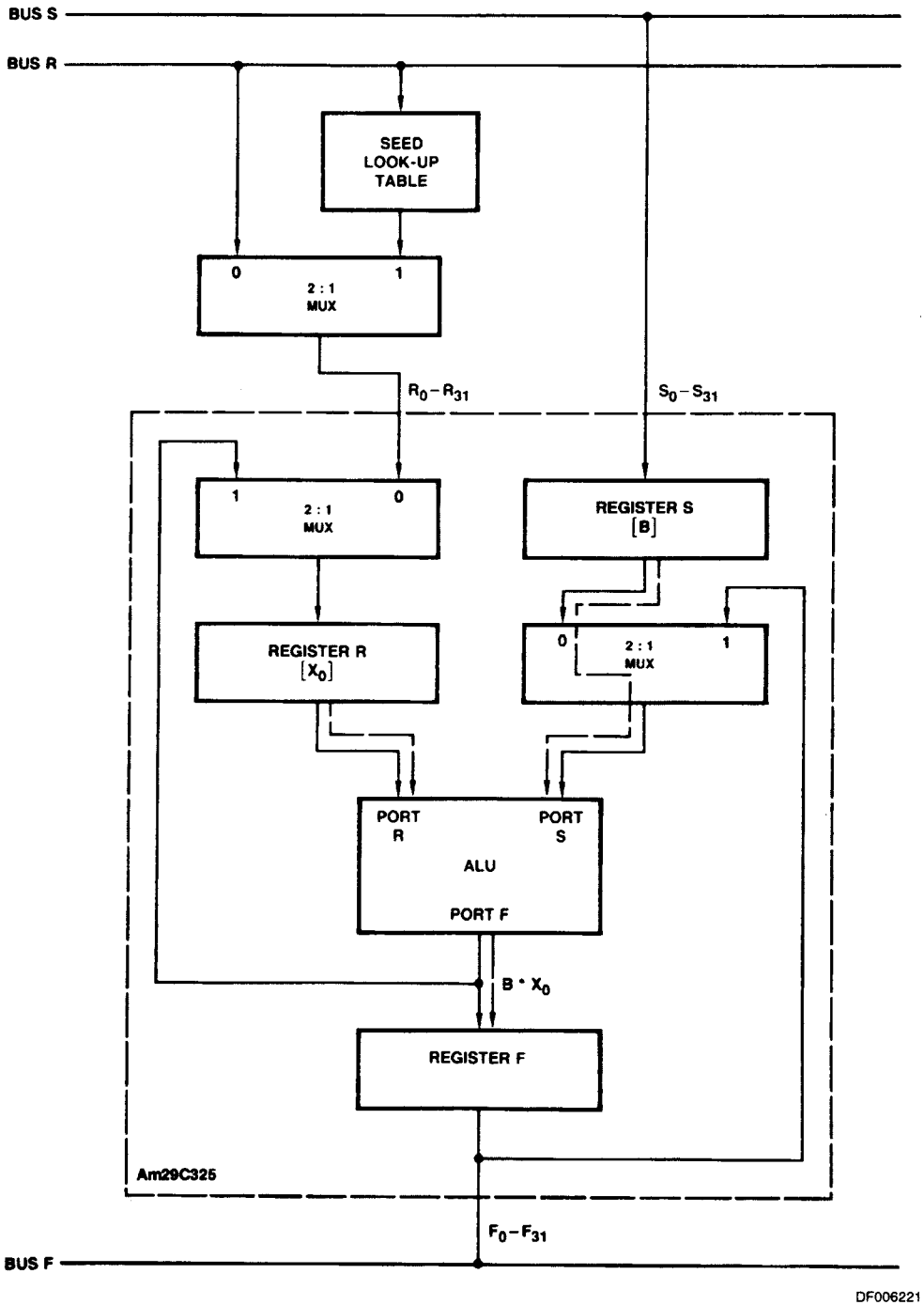


Figure C3-B. Data Flow for Step 2 of the Reciprocal Procedure

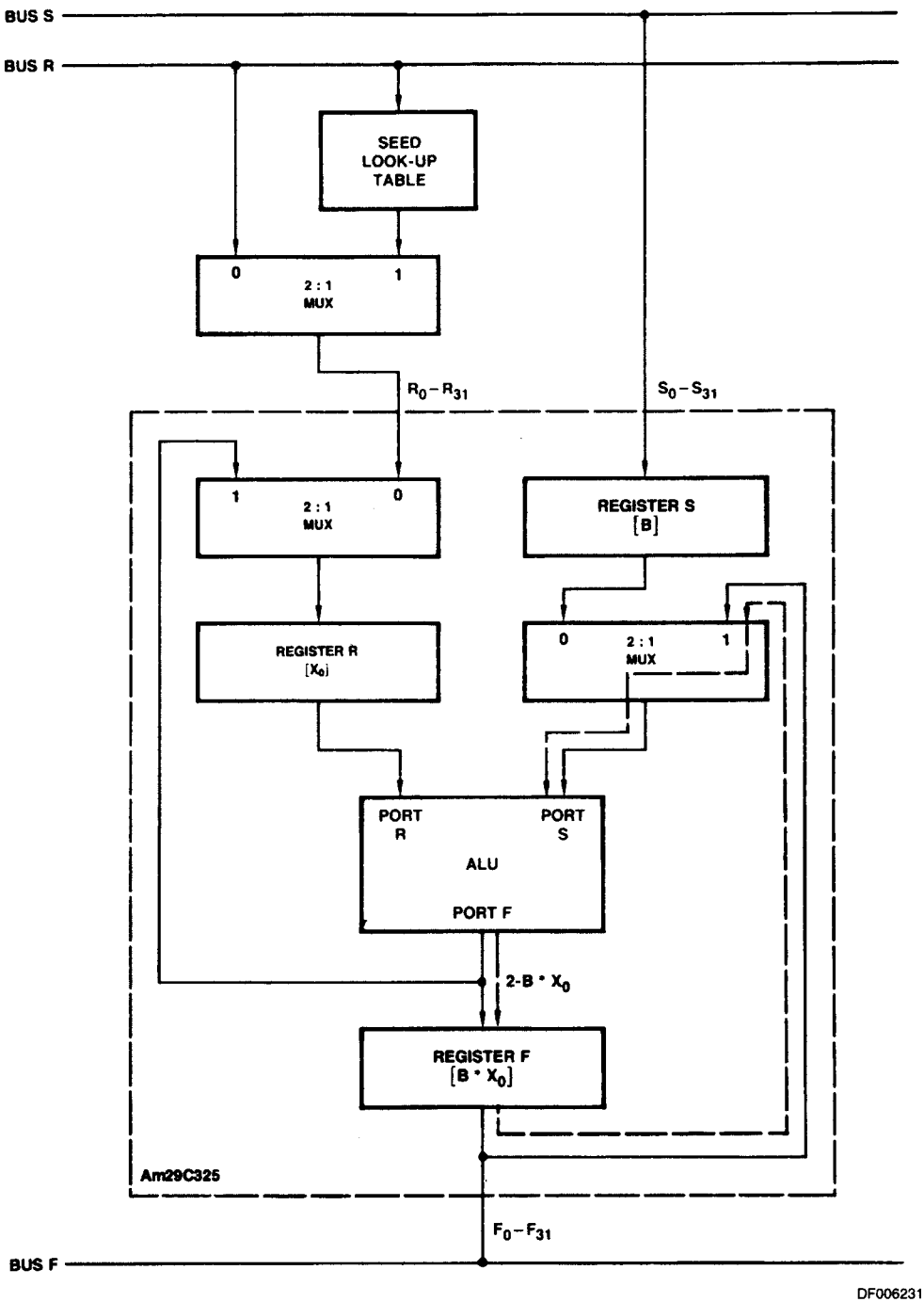


Figure C3-C. Data Flow for Step 3 of the Reciprocal Procedure

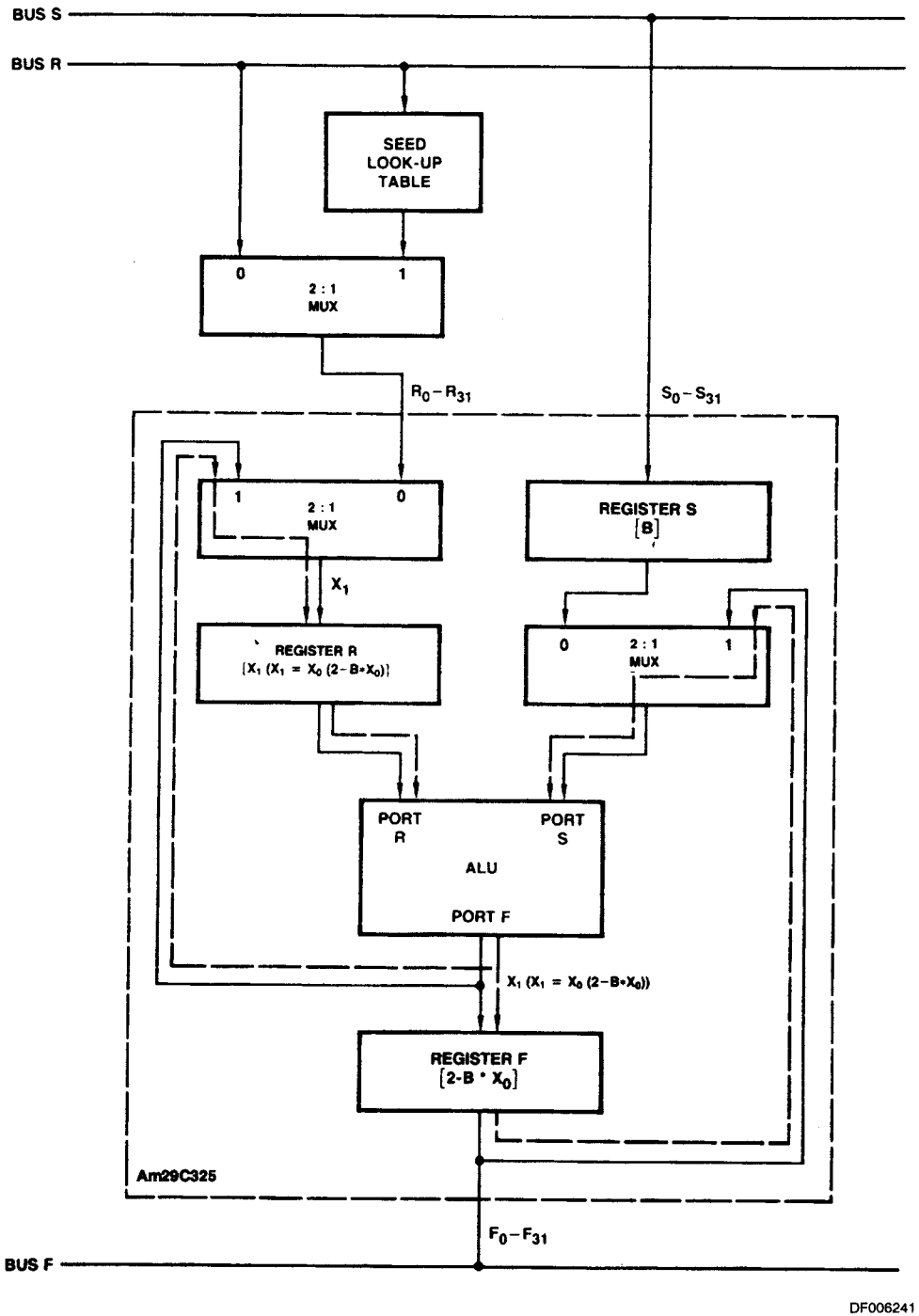


Figure C3-D. Data Flow for Step 4 of the Reciprocal Procedure

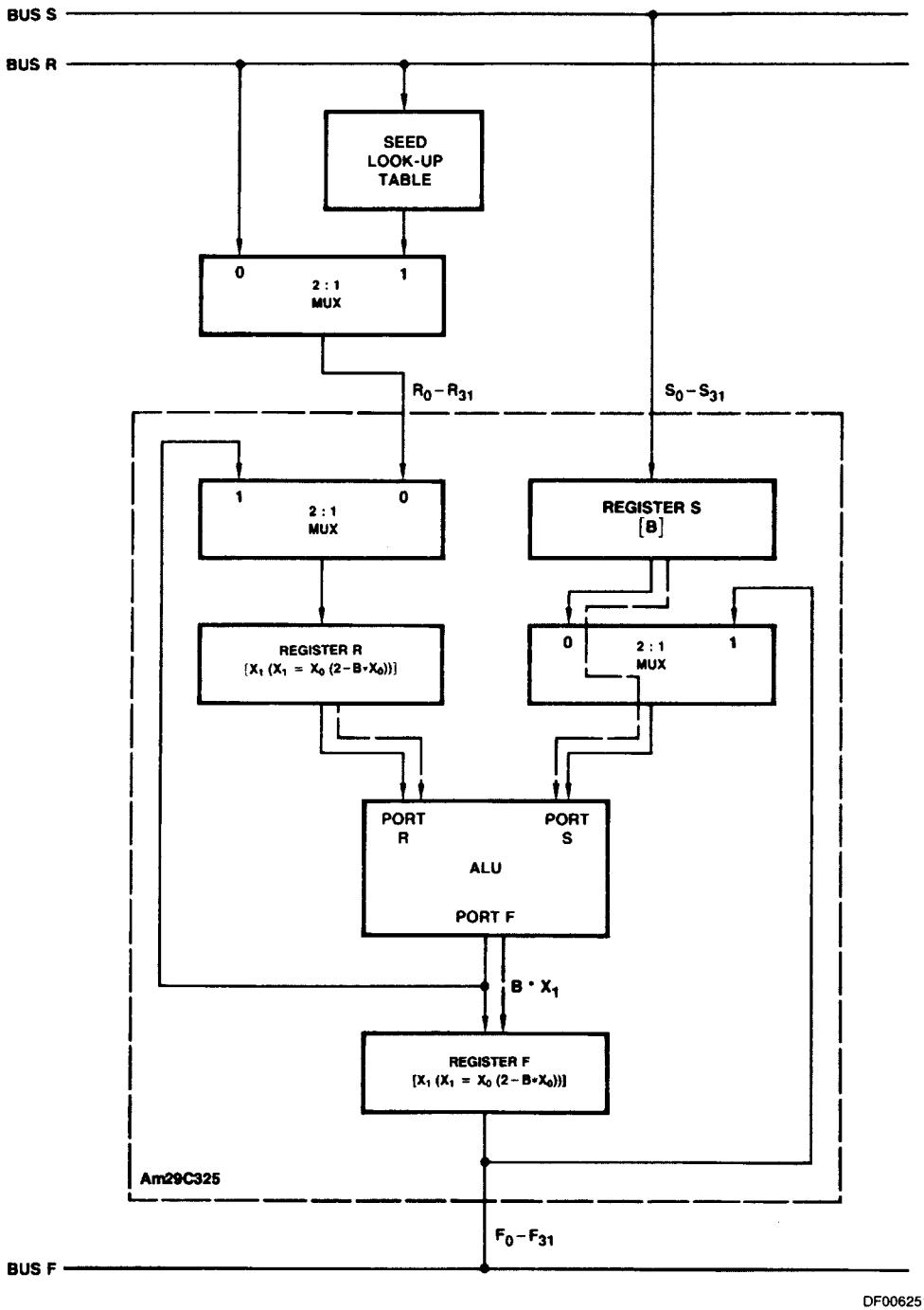


Figure C3-E. Data Flow for Step 5 of the Reciprocal Procedure

A tabular description of the operations above is given in Table C3. The following examples, performed in IEEE format, illustrate the process.

Example 1:

Find the reciprocal of 25.3.

Solution: The IEEE floating-point representation for 25.3 is $41CA6666_{16}$. The reciprocal process is begun by feeding this value to both the seed look-up table and port S. The look-up table produces the value $.03952789_{10}$ ($3D21E800_{16}$). The reciprocal is evaluated using the procedure described above; register values for each step are given in Table C4. The expected result, to the precision of the floating-point word, is $.03952569_{10}$ ($3D21E5B1_{16}$). In this case the expected result is produced after the first iteration. All subsequent iterations produce the same result and are therefore unnecessary.

TABLE C3. SEQUENCE OF EVENTS FOR EVALUATING RECIPROCALS

Clock Cycle	$I_0 - I_2$	I_3	I_4	$\overline{ENR}$	$\overline{ENS}$	$\overline{ENF}$	Register R	Register S	Register F	
1	Y	X	0	0	0	X	–	–	–	
2	R TIMES S	0	X	1	1	0	X_0	B	–	
3	2 MINUS S	1	X	1	1	0	X_0	B	$B*X_0$	First iteration
4	R TIMES S	1	1	0	1	0	X_0	B	$2 - B*X_0$	
5	R TIMES S	0	X	1	1	0	$X_1(= X_0(2 - B*X_0))$	B	$X_1(= X_0(2 - B*X_0))$	
6	2 MINUS S	1	X	1	1	0	X_1	B	$B*X_1$	Second iteration
7	R TIMES S	1	1	0	1	0	X_1	B	$2 - B*X_1$	
8	R TIMES S	0	X	1	1	0	$X_2(= X_1(2 - B*X_1))$	B	$X_2(= X_1(2 - B*X_1))$	

X = DON'T CARE

TABLE C4. INPUT BUS AND REGISTER VALUES FOR EXAMPLE 1

Clock Cycle	R Input	S Input	Register R	Register S	Register F	
1	3D21E800 (.03952789)	$41CA6666_{16}$ (25.3)	–	–	–	
2	–	–	$3D21E800_{16}$ (.03952789)	$41CA6666_{16}$ (25.3)	–	
3	–	–	$3D21E800_{16}$ (.03952789)	$41CA6666_{16}$ (25.3)	$3F8001D3_{16}$ (1.0000556)	
4	–	–	$3D21E800_{16}$ (.03952789)	$41CA6666_{16}$ (25.3)	$3F7FFC5A_{16}$ (.99984419)	
5	–	–	$3D21E5B1_{16}$ (.03952569)	$41CA6666_{16}$ (25.3)	$3D21E5B1_{16}$ (.03952569)	← Result of first iteration
6	–	–	$3D21E5B1_{16}$ (.03952569)	$41CA6666_{16}$ (25.3)	$3F7FFFFF_{16}$ (.99999994)	
7	–	–	$3D21E5B1_{16}$ (.03952569)	$41CA6666_{16}$ (25.3)	$3F800000_{16}$ (1.0)	
8	–	–	$3D21E5B1_{16}$ (.03952569)	$41CA6666_{16}$ (25.3)	$3D21E5B1_{16}$ (.03952569)	← Result of second iteration

Example 2:

Find the reciprocal of −0.4725.

Solution: The IEEE floating-point representation for −0.4725 is $BEF1EB85_{16}$. The reciprocal process is begun by feeding this value to both the seed look-up table and port S. The look-up table produces the value -2.1162109_{10} ($C0077000_{16}$). The reciprocal is evaluated using the procedure described above; register values for each step are given in Table C5. The expected result, to the precision of the floating-point word, is -2.116402_{10} ($C0077322_{16}$). In this case the expected result is produced after the first iteration. All subsequent iterations produce the same result and are therefore unnecessary.

TABLE C5. INPUT BUS AND REGISTER VALUES FOR EXAMPLE 2

Clock Cycle	R Input	S Input	Register R	Register S	Register F	
1	$C0077000_{16}$ (−2.1162109)	$BEF1EB85_{16}$ (−0.4725)	–	–	–	
2	–	–	$C0077000_{16}$ (−2.1162109)	$BEF1EB85_{16}$ (−0.4725)	–	
3	–	–	$C0077000_{16}$ (−2.1162109)	$BEF1EB85_{16}$ (−0.4725)	$3F7FFA14_{16}$ (0.99990963)	
4	–	–	$C0077000_{16}$ (−2.1162109)	$BEF1EB85_{16}$ (−0.4725)	$3F8002F6_{16}$ (1.0000904)	
5	–	–	$C0077322_{16}$ (−2.116402)	$BEF1EB85_{16}$ (−0.4725)	$C0077322_{16}$ (−2.116402)	← Result of first iteration
6	–	–	$C0077322_{16}$ (−2.116402)	$BEF1EB85_{16}$ (−0.4725)	$3F800000_{16}$ (1.0)	
7	–	–	$C0077322_{16}$ (−2.116402)	$BEF1EB85_{16}$ (−0.4725)	$3F800000_{16}$ (1.0)	
8	–	–	$C0077322_{16}$ (−2.116402)	$BEF1EB85_{16}$ (−0.4725)	$C0077322_{16}$ (−2.116402)	← Result of second iteration

APPENDIX D

SUMMARY OF FLAG OPERATION

Tables D1, D2, and D3 summarize flag operation for the IEEE mode, the DEC mode and for the IEEE-TO-DEC and DEC-TO-IEEE operations.

TABLE D1. FLAG SUMMARY FOR IEEE MODE

Operation	Condition(s)	INV	OVF	UNF	INE	ZER	NAN
Any operation listed in the IEEE Invalid Operations Table		H	L	L	L	L	H
R PLUS S R MINUS S R TIMES S 2 MINUS S	Input operands are finite |rounded result| $\geq 2^{128}$	L	H	L	H	L	L
R PLUS S R MINUS S R TIMES S	$0 <$ |rounded result| $< 2^{-126}$	L	L	H	H	H	L
R PLUS S R MINUS S R TIMES S 2 MINUS S INT-TO-FP FP-TO-INT	Final result does not equal infinitely precise result	L	*	*	H	*	L
R PLUS S R MINUS S R TIMES S 2 MINUS S INT-TO-FP FP-TO-INT	Final result is zero	L	L	*	*	H	L
R PLUS S R MINUS S R TIMES S 2 MINUS S FP-TO-INT	Final result is a NAN	*	L	L	L	L	H

Notes: INV = Invalid operation flag
OVF = Overflow flag
UNF = Underflow flag
INE = Inexact flag
ZER = Zero flag
NAN = NAN flag
L = LOW
H = HIGH
* = State of flag depends on the input operands and the operation performed

TABLE D2. FLAG SUMMARY FOR DEC MODE

Operation	Condition(s)	INV	OVF	UNF	INE	ZER	NAN
FP-TO-INT	Rounded result $> 2^{31}-1$ or rounded result $< -2^{31}$	H	L	L	L	L	H
FP-TO-INT	Input is a DEC-reserved operand	L	L	L	L	L	H
R PLUS S R MINUS S R TIMES S 2 MINUS S	$\lvert$Rounded result$\rvert \geq 2^{127}$	L	H	L	H	L	H
R PLUS S R MINUS S R TIMES S	$0 < \lvert$rounded result$\rvert < 2^{-128}$	L	L	H	H	H	L
R PLUS S R MINUS S R TIMES S 2 MIMUS S INT-TO-FP FP-TO-INT	Final result does not equal infinitely precise result	L	*	*	H	*	*
R PLUS S R MINUS S R TIMES S 2 MINUS S INT-TO-FP FP-TO-INT	Final result is zero	L	L	*	*	H	L
R PLUS S R MINUS S R TIMES S 2 MINUS S FP-TO-INT	Final result is a DEC-reserved operand	*	*	L	L	L	H

Notes: INV = Invalid operation flag
OVF = Overflow flag
UNF = Underflow flag
INE = Inexact flag
ZER = Zero flag
NAN = NAN flag
L = LOW
H = HIGH
* = State of flag depends on the input operands and the operation performed

TABLE D3. FLAG SUMMARY FOR IEEE-TO-DEC AND DEC-TO-IEEE CONVERSIONS

Operation	Condition(s)	INV	OVF	UNF	INE	ZER	NAN
IEEE-TO-DEC	Input is a NAN	H	L	L	L	L	H
IEEE-TO-DEC	$\lvert$Input$\rvert \geq 2^{127}$	L	H	L	H	L	H
DEC-TO-IEEE	Input is a DEC-reserved operand	H	L	L	L	L	H
DEC-TO-IEEE	$0 < \lvert$rounded result$\rvert < 2^{-126}$	L	L	H	H	H	L
DEC-TO-IEEE IEEE-TO-DEC	Final result is zero	L	L	*	*	H	L

Notes: INV = Invalid operation flag
OVF = Overflow flag
UNF = Underflow flag
INE = Inexact flag
ZER = Zero flag
NAN = NAN flag
L = LOW
H = HIGH
* = State of flag depends on the input operands and the operation performed