

Am29332

32-Bit Arithmetic Logic Unit



Am29332

Advanced Micro Devices

DISTINCTIVE CHARACTERISTICS

- **Single Chip, 32-Bit ALU**
Supports 80–90 ns microcycle time for the 32-bit data path. It is a combinatorial ALU with equal cycle time for all instructions.
- **Flow-through Architecture**
A combinatorial ALU with two input data ports and one output data port allows implementation of either parallel or pipelined architectures.
- **64-Bit In, 32-Bit Out Funnel Shifter**
This unique functional block allows n-bit shift-up, shift-down, 32-bit barrel shift or 32-bit field extract.
- **Supports All Data Types**
It supports one-, two-, three- and four-byte data for all operations and variable-length fields for logical operations.
- **Multiply and Divide Support**
Built-in hardware to support two-bit-at-a-time modified Booth's algorithm and one-bit-at-a-time division algorithm.
- **Extensive Error Checking**
Parity check and generate provides data transmission check and master/slave mode provides complete function checking.

GENERAL DESCRIPTION

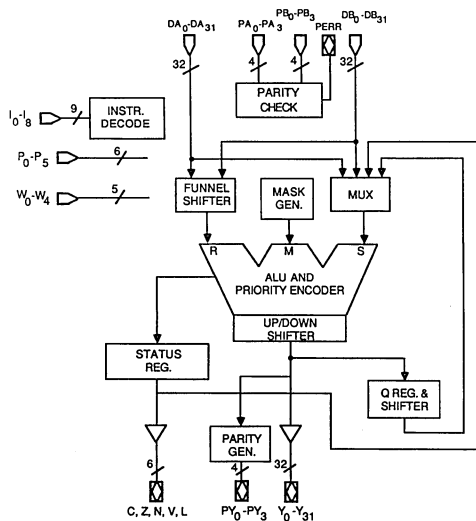
The Am29332 is a 32-bit wide non-cascadable Arithmetic Logic Unit (ALU) with integration of functions that normally don't cascade, such as barrel shifters, priority encoders and mask generators. Two input data ports and one output data port provide flow-through architecture and allow the designer to implement his/her architecture with any degree of pipelining and no built-in penalties for branching. Also, the simplicity of a three-bus ALU allows easy implementation of parallel or reconfigurable architectures. The register file is off-chip to allow unlimited expansion and regular addressability.

The Am29332 supports one-, two-, three- and four-byte data for arithmetic and logic operations. It also supports

multiprecision arithmetic and shift operations. For logical operations, it can support variable-length fields up to 32 bits. When fewer than four bytes are selected, unselected bits are passed to the destination without modification. The device also supports two-bit-at-a-time modified Booth's algorithm for high-speed multiplication and one-bit-at-a-time division. Both signed and unsigned integers for all byte aligned data types mentioned above are supported.

The Am29332 is designed to support 80–90 ns microcycle time. The device is packaged in a 169-lead pin-grid-array package.

SIMPLIFIED BLOCK DIAGRAM



BD007040

| | | |
|-----------------------|------|-----------|
| Publication # | Rev. | Amendment |
| 05730 | E | /0 |
| Issue Date: July 1987 | | |

RELATED AMD PRODUCTS

| Part No. | Description |
|----------|---|
| Am29C01 | CMOS 4-Bit Microprocessor Slice |
| Am29C10A | CMOS 12-Bit Sequencer |
| Am29C101 | CMOS 16-Bit Microprocessor |
| Am29112 | 8-Bit Cascadable Microprogram Sequencer |
| Am29114 | Real-Time Interrupt Controller |
| Am29C116 | CMOS 16-Bit Microcontroller |
| Am29C323 | CMOS 32 x 32 Parallel Multiplier |
| Am29325 | 32-Bit Floating Point Processor |
| Am29C325 | CMOS 32-Bit Floating Point Processor |
| Am29331 | 16-Bit Microprogram Sequencer |
| Am29C331 | CMOS 16-Bit Microprogram Sequencer |
| Am29334 | 64 x 18 Four-Port, Dual-Access Register File |
| Am29C334 | CMOS 64 x 18 Four-Port, Dual-Access Register File |
| Am29337 | 16-Bit Bounds Checker |
| Am29338 | 32-Bit Byte Queue |
| Am29C516 | CMOS 16 x 16 Multiplier |
| Am29C517 | CMOS 16 x 16 Multiplier with Separate I/O |

CONNECTION DIAGRAM 169-Lead PGA Bottom View

| | A | B | C | D | E | F | G | H | J | K | L | M | N | P | R | T | U |
|----|-------------|------|-------|-----|-----|------|------|------|------|------|------|------|------|-----------------|-------|------|------|
| 1 | DB6 | DA6 | DB7 | DB8 | DB9 | DB10 | DA11 | DB12 | DA14 | DB16 | PB1 | DB18 | DB19 | DB20 | DB22 | DA23 | PA2 |
| 2 | DA5 | DA5 | DA7 | PB0 | DA9 | DB11 | DA12 | DA13 | DB14 | PA1 | DA16 | DA17 | DA19 | DA20 | DA21 | DB23 | PB2 |
| 3 | DB4 | DA3 | DA4 | PA0 | DA8 | DA10 | GND | DB13 | DB15 | DA15 | VCC | DB17 | DA18 | DB21 | DA22 | DA24 | DB24 |
| 4 | DB2 | DA2 | DB3 | * | | | | | | | | | | | DB25 | DA25 | DB26 |
| 5 | DB1 | DA1 | DA0 | | | | | | | | | | | | DB26 | DA27 | DB27 |
| 6 | DB0 | P5 | P4 | | | | | | | | | | | | DB28 | DA28 | DB29 |
| 7 | P1 | P3 | VCC | | | | | | | | | | | | VCC | DA30 | DA29 |
| 8 | P2 | P0 | W4 | | | | | | | | | | | | DB31 | DA31 | DB30 |
| 9 | W2 | W1 | W3 | | | | | | | | | | | | MSERR | PA3 | PB3 |
| 10 | I2 | W0 | I0 | | | | | | | | | | | | Y31 | Y30 | Y28 |
| 11 | I3 | I1 | GND | | | | | | | | | | | | GND | Y27 | Y29 |
| 12 | I6 | I4 | I5 | | | | | | | | | | | | VCC | GND | Y26 |
| 13 | I8 | I7 | CP | | | | | | | | | | | | Y25 | Y23 | Y24 |
| 14 | MINK | RS | SLAVE | | | | | | | | | | | | GND | VCC | Y22 |
| 15 | M \bar{M} | MCin | N | C | PY3 | PY2 | GND | Y3 | GND | Y7 | VCC | Y8 | Y12 | CE-Y | Y19 | Y21 | Y20 |
| 16 | BOROW | V | L | VCC | PY1 | GNDT | GND | Y2 | Y5 | Y6 | VCC | Y11 | Y10 | Y13 | Y15 | Y18 | Y17 |
| 17 | HOLD | Z | GND | PY0 | Y0 | FERR | GND | Y1 | Y4 | GNDT | VCC | Y9 | VCC | GND | Y14 | Y16 | GND |

CD010462

* This pin is not used

Key: VCCE = VCC, ECL
 VCCT = VCC, TTL
 GNDE = GND, ECL
 GNDT = GND, TTL

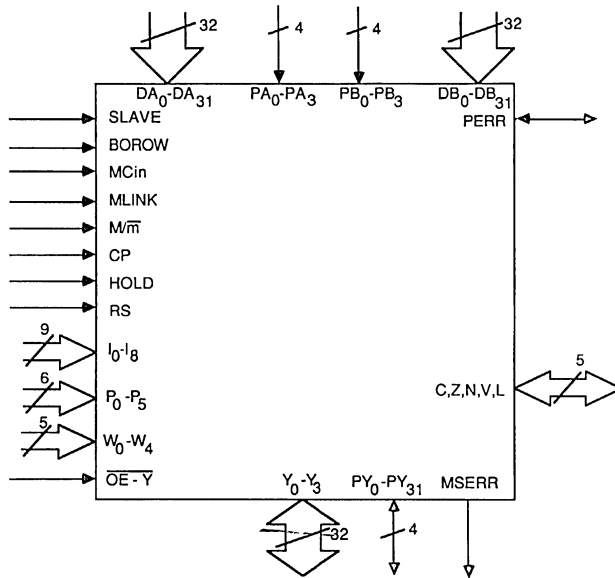
PIN DESIGNATIONS
(Sorted by Pin No.)

| PIN NO. | PIN NAME | PAD NO. | PIN NO. | PIN NAME | PAD NO. | PIN NO. | PIN NAME | PAD NO. | PIN NO. | PIN NAME | PAD NO. |
|---------|-----------------------|---------|---------|-----------------------|---------|---------|----------------------------------|---------|---------|-----------------------|---------|
| A-1 | DB ₆ | 1 | C-9 | W ₃ | 145 | J-15 | GND, TTL | 105 | R-10 | Y ₃₁ | 66 |
| A-2 | DA ₅ | 164 | C-10 | I ₀ | 139 | J-16 | Y ₅ | 101 | R-11 | GND, ECL | 64 |
| A-3 | DB ₄ | 161 | C-11 | GND, ECL | 143 | J-17 | Y ₄ | 102 | R-12 | V _{CC} , TTL | 71 |
| A-4 | DB ₂ | 157 | C-12 | I ₅ | 134 | K-1 | DB ₁₆ | 27 | R-13 | Y ₂₅ | 74 |
| A-5 | DB ₁ | 155 | C-13 | CP | 130 | K-2 | PA ₁ | 25 | R-14 | GND, TTL | 79 |
| A-6 | DB ₀ | 153 | C-14 | SLAVE | 127 | K-3 | DA ₁₅ | 24 | R-15 | Y ₁₉ | 82 |
| A-7 | P ₁ | 148 | C-15 | N | 120 | K-15 | Y ₇ | 99 | R-16 | Y ₁₅ | 88 |
| A-8 | P ₂ | 149 | C-16 | L | 118 | K-16 | Y ₆ | 100 | R-17 | Y ₁₄ | 89 |
| A-9 | W ₂ | 142 | C-17 | GND, TTL | 117 | K-17 | GND, TTL | 98 | T-1 | DA ₂₃ | 42 |
| A-10 | I ₂ | 137 | D-1 | DB ₈ | 7 | L-1 | PB ₁ | 26 | T-2 | DB ₂₃ | 41 |
| A-11 | I ₃ | 136 | D-2 | PB ₀ | 6 | L-2 | DA ₁₆ | 28 | T-3 | DA ₂₄ | 46 |
| A-12 | I ₆ | 133 | D-3 | PA ₀ | 5 | L-3 | V _{CC} , ECL | 22 | T-4 | DA ₂₅ | 48 |
| A-13 | I ₈ | 131 | D-15 | C | 119 | L-15 | V _{CC} , ECL | 103 | T-5 | DA ₂₇ | 52 |
| A-14 | MLINK | 129 | D-16 | V _{CC} , TTL | 116 | L-16 | V _{CC} , ECL | 103 | T-6 | DA ₂₈ | 54 |
| A-15 | M/ \overline{m} | 125 | D-17 | PY ₀ | 115 | L-17 | V _{CC} , ECL | 103 | T-7 | DA ₃₀ | 58 |
| A-16 | BOROW | 124 | E-1 | DB ₉ | 9 | M-1 | DB ₁₈ | 31 | T-8 | DA ₃₁ | 60 |
| A-17 | HOLD | 123 | E-2 | DA ₉ | 10 | M-2 | DA ₁₇ | 30 | T-9 | PA ₃ | 61 |
| B-1 | DA ₆ | 2 | E-3 | DA ₈ | 8 | M-3 | DB ₁₇ | 29 | T-10 | Y ₃₀ | 67 |
| B-2 | DB ₅ | 163 | E-15 | PY ₃ | 112 | M-15 | Y ₈ | 96 | T-11 | Y ₂₇ | 70 |
| B-3 | DA ₃ | 160 | E-16 | PY ₁ | 114 | M-16 | Y ₁₁ | 93 | T-12 | GND, TTL | 72 |
| B-4 | DA ₂ | 158 | E-17 | Y ₀ | 109 | M-17 | Y ₉ | 95 | T-13 | Y ₂₃ | 76 |
| B-5 | DA ₁ | 156 | F-1 | DB ₁₀ | 11 | N-1 | DB ₁₉ | 33 | T-14 | V _{CC} , TTL | 78 |
| B-6 | P ₅ | 152 | F-2 | DB ₁₁ | 13 | N-2 | DA ₁₉ | 34 | T-15 | Y ₂₁ | 80 |
| B-7 | P ₃ | 150 | F-3 | DA ₁₀ | 12 | N-3 | DA ₁₈ | 32 | T-16 | Y ₁₈ | 83 |
| B-8 | P ₀ | 147 | F-15 | PY ₂ | 113 | N-15 | Y ₁₂ | 92 | T-17 | Y ₁₆ | 86 |
| B-9 | W ₁ | 141 | F-16 | GND, TTL | 110 | N-16 | Y ₁₀ | 94 | U-1 | PA ₂ | 43 |
| B-10 | W ₀ | 140 | F-17 | PERR | 111 | N-17 | V _{CC} , TTL | 97 | U-2 | PB ₂ | 44 |
| B-11 | I ₁ | 138 | G-1 | DA ₁₁ | 14 | P-1 | DB ₂₀ | 35 | U-3 | DB ₂₄ | 45 |
| B-12 | I ₄ | 135 | G-2 | DA ₁₂ | 16 | P-2 | DA ₂₀ | 36 | U-4 | DB ₂₆ | 49 |
| B-13 | I ₇ | 132 | G-3 | GND, ECL | 21 | P-3 | DB ₂₁ | 37 | U-5 | DB ₂₇ | 51 |
| B-14 | RS | 128 | G-15 | GND, ECL | 104 | P-15 | \overline{OE} - \overline{Y} | 87 | U-6 | DB ₂₉ | 55 |
| B-15 | MCin | 126 | G-16 | GND, ECL | 104 | P-16 | Y ₁₃ | 90 | U-7 | DA ₂₉ | 56 |
| B-16 | V | 121 | G-17 | GND, ECL | 104 | P-17 | GND, TTL | 91 | U-8 | DB ₃₀ | 57 |
| B-17 | Z | 122 | H-1 | DB ₁₂ | 15 | R-1 | DB ₂₂ | 39 | U-9 | PB ₃ | 62 |
| C-1 | DB ₇ | 3 | H-2 | DA ₁₃ | 18 | R-2 | DA ₂₁ | 38 | U-10 | Y ₂₈ | 69 |
| C-2 | DA ₇ | 4 | H-3 | DB ₁₃ | 17 | R-3 | DA ₂₂ | 40 | U-11 | Y ₂₉ | 68 |
| C-3 | DA ₄ | 162 | H-15 | Y ₃ | 106 | R-4 | DB ₂₅ | 47 | U-12 | Y ₂₆ | 73 |
| C-4 | DB ₃ | 159 | H-16 | Y ₂ | 107 | R-5 | DA ₂₆ | 50 | U-13 | Y ₂₄ | 75 |
| C-5 | DA ₀ | 154 | H-17 | Y ₁ | 108 | R-6 | DB ₂₈ | 53 | U-14 | Y ₂₂ | 77 |
| C-6 | P ₄ | 151 | J-1 | DA ₁₄ | 20 | R-7 | V _{CC} , ECL | 63 | U-15 | Y ₂₀ | 81 |
| C-7 | V _{CC} , ECL | 144 | J-2 | DB ₁₄ | 19 | R-8 | DB ₃₁ | 59 | U-16 | Y ₁₇ | 84 |
| C-8 | W ₄ | 146 | J-3 | DB ₁₅ | 23 | R-9 | MSERR | 65 | U-17 | GND, TTL | 85 |

PIN DESIGNATIONS
(Sorted by Pin Names)

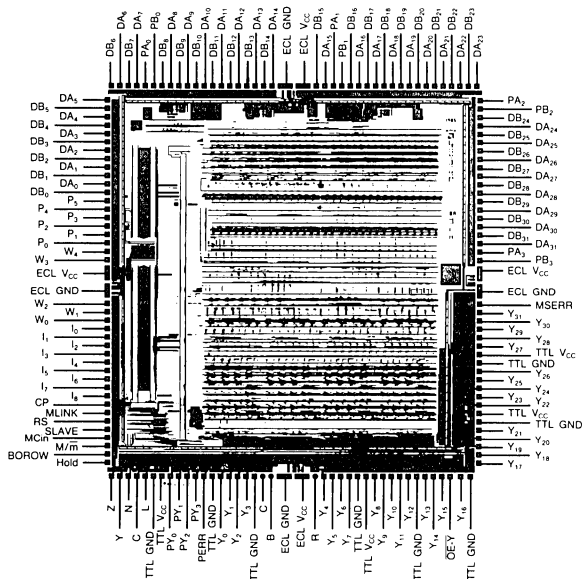
| PIN NAME | PIN NO. | PAD NO. | PIN NAME | PIN NO. | PAD NO. | PIN NAME | PIN NO. | PAD NO. | PIN NAME | PIN NO. | PAD NO. |
|------------------|---------|---------|------------------|---------|---------|-----------------------|---------|---------|-----------------------|---------|---------|
| BOROW | A-16 | 124 | DB ₇ | C-1 | 3 | I ₂ | A-10 | 137 | V _{CC} , TTL | T-14 | 78 |
| C | D-15 | 119 | DB ₈ | D-1 | 7 | I ₃ | A-11 | 136 | V _{CC} , TTL | N-17 | 97 |
| CP | C-13 | 130 | DB ₉ | E-1 | 9 | I ₄ | B-12 | 135 | V _{CC} , TTL | D-16 | 116 |
| DA ₀ | C-5 | 154 | DB ₁₀ | F-1 | 11 | I ₅ | C-12 | 134 | V _{CC} , TTL | H-12 | 71 |
| DA ₁ | B-5 | 156 | DB ₁₁ | F-2 | 13 | I ₆ | A-12 | 133 | W ₀ | B-10 | 140 |
| DA ₂ | B-4 | 158 | DB ₁₂ | H-1 | 15 | I ₇ | B-13 | 132 | W ₁ | B-9 | 141 |
| DA ₃ | B-3 | 160 | DB ₁₃ | H-3 | 17 | I ₈ | A-13 | 131 | W ₂ | A-9 | 142 |
| DA ₄ | C-3 | 162 | DB ₁₄ | J-2 | 19 | L | C-16 | 118 | W ₃ | C-9 | 145 |
| DA ₅ | A-2 | 164 | DB ₁₅ | J-3 | 23 | MC _{in} | B-15 | 126 | W ₄ | C-8 | 146 |
| DA ₆ | B-1 | 2 | DB ₁₆ | K-1 | 27 | MLINK | A-14 | 129 | Y ₀ | E-17 | 109 |
| DA ₇ | C-2 | 4 | DB ₁₇ | M-3 | 29 | M/ \bar{m} | A-15 | 125 | Y ₁ | H-17 | 108 |
| DA ₈ | E-3 | 8 | DB ₁₈ | M-1 | 31 | MSERR | R-9 | 65 | Y ₂ | H-16 | 107 |
| DA ₉ | E-2 | 10 | DB ₁₉ | N-1 | 33 | N | C-15 | 120 | Y ₃ | H-15 | 106 |
| DA ₁₀ | F-3 | 12 | DB ₂₀ | P-1 | 35 | $\overline{OE-Y}$ | P-15 | 87 | Y ₄ | J-17 | 102 |
| DA ₁₁ | G-1 | 14 | DB ₂₁ | P-3 | 37 | P ₀ | B-8 | 147 | Y ₅ | J-16 | 101 |
| DA ₁₂ | G-2 | 16 | DB ₂₂ | R-1 | 39 | P ₁ | A-7 | 148 | Y ₆ | K-16 | 100 |
| DA ₁₃ | H-2 | 18 | DB ₂₃ | T-2 | 41 | P ₂ | A-8 | 149 | Y ₇ | K-15 | 99 |
| DA ₁₄ | J-1 | 20 | DB ₂₄ | U-3 | 45 | P ₃ | B-7 | 150 | Y ₈ | M-15 | 96 |
| DA ₁₅ | K-3 | 24 | DB ₂₅ | R-4 | 47 | P ₄ | C-6 | 151 | Y ₉ | M-17 | 95 |
| DA ₁₆ | L-2 | 28 | DB ₂₆ | U-4 | 49 | P ₅ | B-6 | 152 | Y ₁₀ | N-16 | 94 |
| DA ₁₇ | M-2 | 30 | DB ₂₇ | U-5 | 51 | PA ₀ | D-3 | 5 | Y ₁₁ | M-16 | 93 |
| DA ₁₈ | N-3 | 32 | DB ₂₈ | R-6 | 53 | PA ₁ | K-2 | 25 | Y ₁₂ | N-15 | 92 |
| DA ₁₉ | N-2 | 34 | DB ₂₉ | U-6 | 55 | PA ₂ | U-1 | 43 | Y ₁₃ | P-16 | 90 |
| DA ₂₀ | P-2 | 36 | DB ₃₀ | U-8 | 57 | PA ₃ | T-9 | 61 | Y ₁₄ | R-17 | 89 |
| DA ₂₁ | R-2 | 38 | DB ₃₁ | R-8 | 59 | PB ₀ | D-2 | 6 | Y ₁₅ | R-16 | 88 |
| DA ₂₂ | R-3 | 40 | GND, ECL | G-3 | 21 | PB ₁ | L-1 | 26 | Y ₁₆ | T-17 | 86 |
| DA ₂₃ | T-1 | 42 | GND, ECL | R-11 | 64 | PB ₂ | U-2 | 44 | Y ₁₇ | U-16 | 84 |
| DA ₂₄ | T-3 | 46 | GND, ECL | G-17 | 104 | PB ₃ | U-9 | 62 | Y ₁₈ | T-16 | 83 |
| DA ₂₅ | T-4 | 48 | GND, ECL | G-15 | 104 | PERR | F-17 | 111 | Y ₁₉ | R-15 | 82 |
| DA ₂₆ | R-5 | 50 | GND, ECL | G-16 | 104 | PY ₀ | D-17 | 115 | Y ₂₀ | U-15 | 81 |
| DA ₂₇ | T-5 | 52 | GND, ECL | C-11 | 143 | PY ₁ | E-16 | 114 | Y ₂₁ | T-15 | 80 |
| DA ₂₈ | T-6 | 54 | GND, TTL | T-12 | 72 | PY ₂ | F-15 | 113 | Y ₂₂ | U-14 | 77 |
| DA ₂₉ | U-7 | 56 | GND, TTL | R-14 | 79 | PY ₃ | E-15 | 112 | Y ₂₃ | T-13 | 76 |
| DA ₃₀ | T-7 | 58 | GND, TTL | U-17 | 85 | RS | B-14 | 128 | Y ₂₄ | U-13 | 75 |
| DA ₃₁ | T-8 | 60 | GND, TTL | P-17 | 91 | SLAVE | C-14 | 127 | Y ₂₅ | R-13 | 74 |
| DB ₀ | A-6 | 153 | GND, TTL | K-17 | 98 | V | B-16 | 121 | Y ₂₆ | U-12 | 73 |
| DB ₁ | A-5 | 155 | GND, TTL | J-15 | 105 | V _{CC} , ECL | R-7 | 63 | Y ₂₇ | T-11 | 70 |
| DB ₂ | A-4 | 157 | GND, TTL | F-16 | 110 | V _{CC} , ECL | L-16 | 103 | Y ₂₈ | U-10 | 69 |
| DB ₃ | C-4 | 159 | GND, TTL | C-17 | 117 | V _{CC} , ECL | L-15 | 103 | Y ₂₉ | U-11 | 68 |
| DB ₄ | A-3 | 161 | HOLD | A-17 | 123 | V _{CC} , ECL | L-17 | 103 | Y ₃₀ | T-10 | 67 |
| DB ₅ | B-2 | 163 | I ₀ | C-10 | 139 | V _{CC} , ECL | C-7 | 144 | Y ₃₁ | R-10 | 66 |
| DB ₆ | A-1 | 1 | I ₁ | B-11 | 138 | V _{CC} , ECL | L-3 | 22 | Z | B-17 | 122 |

LOGIC SYMBOL



LS002911

METALLIZATION AND PAD LAYOUT



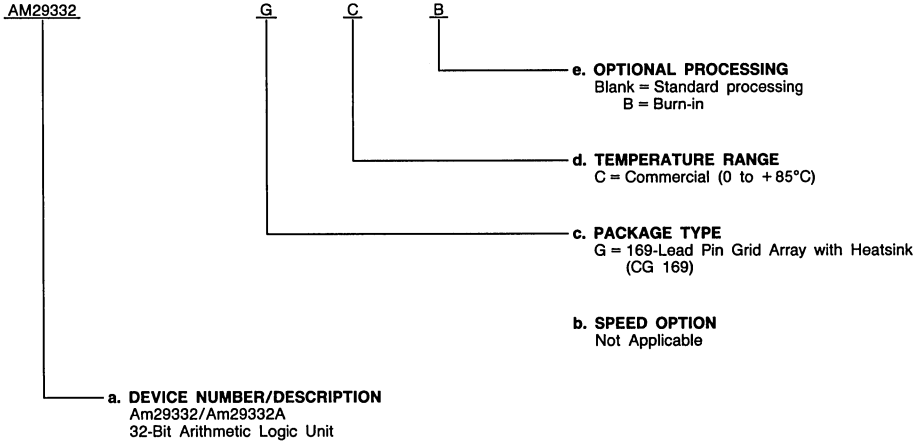
Die size: 367 x 387 mils
Gate Count: 5200

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

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.

| Valid Combinations | |
|--------------------|---------|
| AM29332 | GC, GCB |
| AM29332A | |

PIN DESCRIPTION

BOROW Borrow (Input)

When HIGH, the Carry In and Carry Out are borrows for subtract operations.

C, Z, N, V, L Status (Input/Output)

When the Register Status pin is LOW, these pins give the Carry, Zero, Negative, Overflow and Link outputs of the ALU where applicable to the instruction being executed. When not applicable to the instruction being executed, or when the Register Status pin is HIGH, these pins give the outputs of the Carry, Zero, Negative, Overflow and Link bits of the internal Status Register. In Slave mode, C, Z, N, V and L become inputs.

CP Clock Input (Input)

Clocks internal registers (status, Q) at the LOW to HIGH transition, provided HOLD input is LOW.

DA₀ - DA₃₁ Data Input for DA-bus (Input)

Data input lines for operand A.

DB₀ - DB₃₁ Data Input for DB-bus (Input)

Data input lines for operand B.

HOLD Hold (Input, Active HIGH)

When HIGH, it inhibits the update of the status and Q registers.

I₀ - I₆ Instruction Inputs (Input)

Used to select the operation to be performed.

I₇ - I₈ Byte Width Inputs (Input)

Byte width inputs for byte boundary aligned operand instructions. Selects the sources for width and position inputs for variable field bit operands. If I₇ is LOW it selects the width input from pins W₄ - W₀. If I₇ is HIGH the width input is selected from the internal width register. Similarly if I₈ is LOW it selects the position inputs from pins P₅ - P₀ and if HIGH it selects input from the internal position register.

MCin Macro Status Carry (Input)

External Carry input.

MLINK Macro Status Link (Input)

External link input.

M/ \bar{m} Macro/Micro Select (Input)

When HIGH, selects macro carry and macro link pins as input instead of micro carry and micro link from the micro-status register.

MSERR Master-Slave Error (Output)

When HIGH, this signal indicates that the master's and slave's data were not identical.

$\overline{OE-Y}$ Output Enable (Input, Active LOW)

When $\overline{OE-Y}$ is HIGH the Y-bus is disabled (three-stated).

P₀ - P₅ Position Inputs (Input)

Position input to select the position of the least significant bit of a field. Also indicates the amount by which data is to be shifted up (P₅ = LOW) or down (P₅ = HIGH) or rotated.

PA₀ - PA₃ Parity Input for DA-bus (Input)

Parity input for operand A on DA-bus (one per byte). Even parity is used for the Am29332.

PB₀ - PB₃ Parity Input for DB-bus (Input)

Parity input for operand B on DB-bus (one per byte).

PERR Parity Error (Input/Output)

When HIGH, indicates that a parity error was detected on the DA or DB inputs.

PY₀ - PY₃ Parity for Y-bus (Input/Output)

Parity output for data on Y-bus (one per byte). Even parity is used for the Am29332. In slave mode, PY₀ - PY₃ become inputs.

RS Register Status Mode Pin (Input)

Selects between ALU status (Register Status = LOW) or register status (Register Status = HIGH) on the C, Z, N, V and L outputs.

SLAVE Slave (Input)

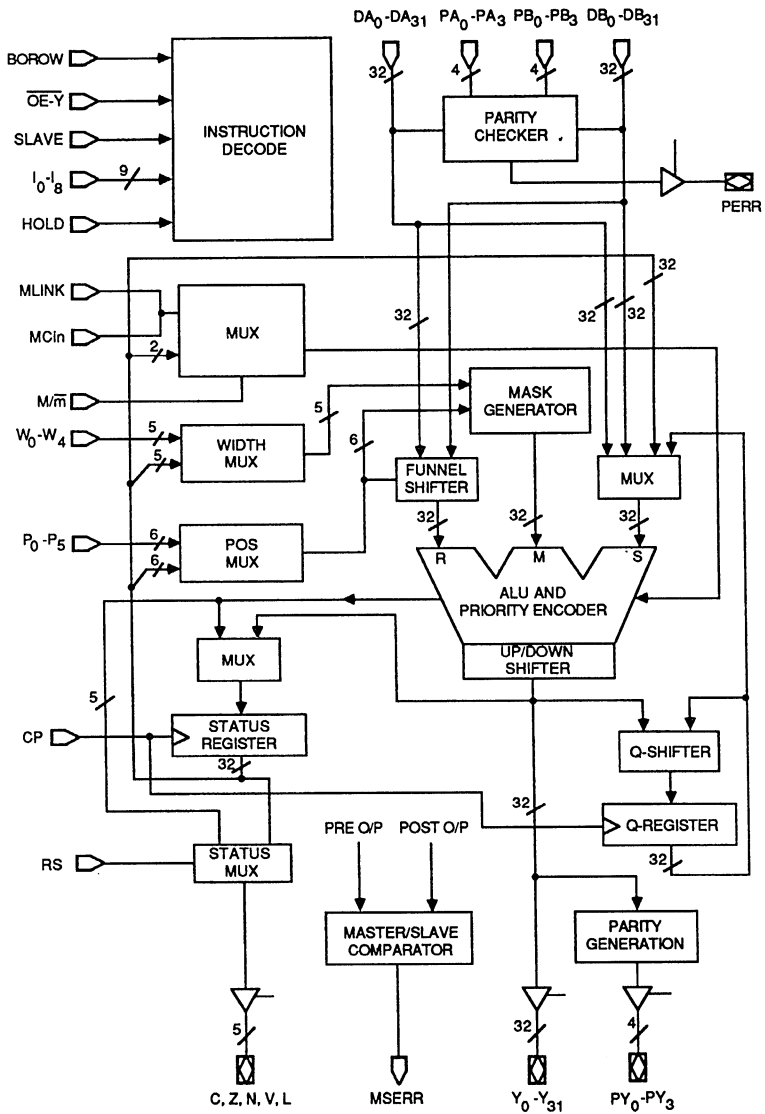
When HIGH, this pin puts the ALU in the slave mode. All output pins become input pins and signals on them are compared with the ALU's internally generated results. When $\overline{OE-Y}$ is HIGH, the Y₀ - Y₃₁ and PY₀ - PY₃ inputs are ignored. When the SLAVE pin is LOW, the ALU is put in master mode where outputs are generated as normal.

W₀ - W₄ Width Inputs (Input)

Width input to select the width of a contiguous bit field.

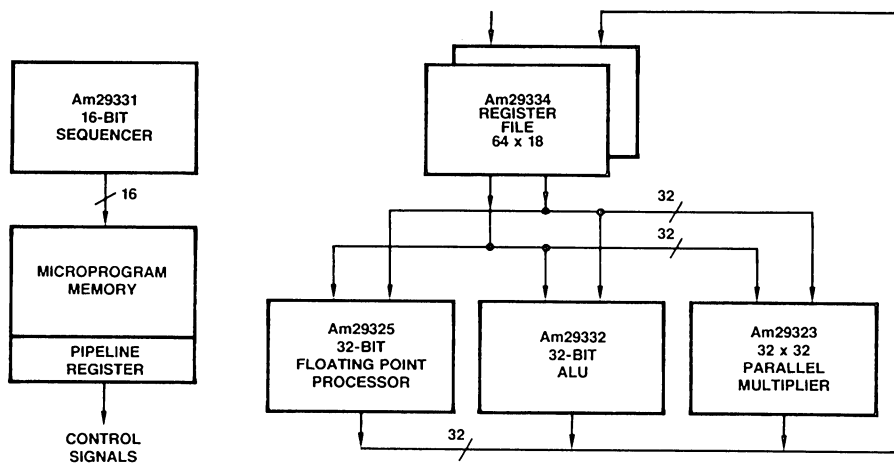
Y₀ - Y₃₁ Data Out/In Lines (Input/Output)

When $\overline{OE-Y}$ is LOW and the ALU is in the Master mode, the ALU result is enabled on the Y-bus. When $\overline{OE-Y}$ is HIGH, the Y-bus is three-stated. In Slave mode the Y-bus acts as external data input.



BD007031

Figure 1. Detailed Block Diagram



AF003480

Figure 2. Am29332 Family High-Performance System Block Diagram

PRODUCT OVERVIEW

The Am29332 is a 32-bit wide, high-performance, non-expandable Arithmetic Logic Unit (ALU). It has two 32-bit wide input ports (A and B) and one 32-bit wide output port (Y). These three ports provide flexibility and accessibility for high-performance processor designs. Dedicated input and output ports provide a flow-through architecture and avoid the penalty associated with switching the bus half-way through the cycle for input and output of data. The chip is designed for use with a dual-access RAM (Am29334) as a register file. In addition, the three-bus architecture facilitates the connection of other arithmetic units in parallel with the Am29332 for high-performance systems.

The Am29332 supports one-, two-, three-, and four-byte arithmetic operations. It also supports multiprecision arithmetic and multiple bit shifts. For logical operations, it can handle variable-length fields of up to 32 bits. The chip incorporates dedicated hardware to allow efficient implementation of a two bit-at-a-time (modified Booth) multiply algorithm, supporting signed and unsigned arithmetic data types. Similarly, hardware is provided to support a bit-at-a-time divide algorithm, also supporting signed and unsigned arithmetic data types. An internal 32-bit register (Q) is used by the multiply and divide hardware for double precision operands. For business applications, the Am29332 supports variable-length BCD arithmetic.

Field logical instructions operate on bit-fields taken from the A and B data inputs; they may be of variable width and starting position. A is normally the source input and B the destination input. In general, destination bits not falling within a specified field are passed by the ALU unchanged. Field width and position are specified either by direct inputs to the chip, or by entries in the status register. There are two kinds of field logical instructions – aligned and non-aligned. The first type of instruction assumes that source and destination fields are aligned and the operation is performed only for bits within the specified fields. In the second type of instruction, source and destination fields are normally non-aligned. However, it is always assumed that one field (either source or destination) is least-significant-bit (LSB) aligned.

If the destination field is LSB aligned then the source field is downshifted in order to make it LSB aligned as well. Down-

shifting is accomplished by making the 6-bit position input equal to the two's complement of the number of places the field is to be downshifted. If the source field is LSB aligned then it is upshifted in order to align it with the destination. Upshifting is accomplished by making the position inputs equal to the number of places the field is to be upshifted. Any other type of field operation is not allowed. Whenever the field crosses the word boundary, the portion not falling within the word boundary is ignored. This effect is useful when performing operations on fields that overlap two different words. Instructions to perform straightforward multiple-bit shifts (either up or down) are also provided. Additionally, it is possible to extract a bit-field from a word in one instruction, even if that field overlaps a word boundary.

The power and the flexibility of the processor comes partly from its ability to generate a mask to control the width of an operation for each instruction without any overhead. For all byte aligned instructions (three quarters of the instruction set), the mask is either 1, 2, 3 or 4 bytes wide and is generated from the byte width input ($I_8 - I_7$). For all field instructions the mask is of variable width and is generated from the position inputs ($P_0 - P_5$) and the width inputs ($W_0 - W_4$). Table 1 describes the position displacement from the position inputs and Table 2 the bit field from the width inputs.

TABLE 1. POSITION INPUTS AND BIT DISPLACEMENT

| Inputs | | | | | | Bit Displacement P |
|----------------|----------------|----------------|----------------|----------------|----------------|-----------------------|
| P ₅ | P ₄ | P ₃ | P ₂ | P ₁ | P ₀ | |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 2 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 0 | 1 | 1 | 1 | 1 | 1 | 31 |
| 1 | 0 | 0 | 0 | 0 | 0 | -32 |
| 1 | 0 | 0 | 0 | 0 | 1 | -31 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 1 | 1 | 1 | 1 | 1 | 1 | -1 |

TABLE 2. WIDTH INPUTS AND BIT FIELD

| Inputs | | | | | Bit Field w |
|----------------|----------------|----------------|----------------|----------------|----------------|
| W ₄ | W ₃ | W ₂ | W ₁ | W ₀ | |
| 0 | 0 | 0 | 0 | 0 | 32 |
| 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 2 |
| ⋮ | ⋮ | ⋮ | ⋮ | ⋮ | ⋮ |
| 1 | 1 | 1 | 1 | 1 | 31 |

Whenever the width of the operand is less than 32-bits, all unselected bits from the inputs of the ALU are passed to the output without any modification. Depending upon the instruction type, unselected bits are taken from different sources. For example in all single operand instructions, bits from the source operand (from either A or B input) are passed in unselected bit positions. For two operand instructions, bits from the B input are passed in unselected bit positions. There are some exceptions which are explained in the instruction set section.

The processor has a 32-bit status register to indicate the status of different operations performed. The status register is loaded at the rising edge of the clock with new status unless the HOLD signal is HIGH. The bit position for each status bit is given in the functional description. The least significant byte of the status register holds the six position bits (PR₀ – PR₅). The two most significant bits of this byte may be read or loaded but are otherwise unused by the ALU. The second byte (bits 8 to 15) consists of the five width bits (WR₀ – WR₄) and three read-only bits that are a combinational function of other status bits, and which indicate useful branch conditions. The third byte consists of ALU status bits plus bits for high-speed multiply and divide. The most significant byte holds intermediate nibble carries for BCD operations. An extract-status instruction is provided which allows a Boolean value to be formed from any selected bit. This is particularly useful in machines employing a stack architecture. Instructions to save and restore the status register are provided. As the entire status of each instruction is stored in the status register, interrupts at any microinstruction boundary are feasible.

The processor has a 32-bit wide priority encoder to support floating-point and graphics operations. The priority encoder supports all byte aligned data types – the result is dependent upon the byte width specified. The result of a priority encode is also loaded into the position bits of the status register. The result of the prioritize operation can then be used in the following clock cycle, e.g., to normalize a floating-point number or to help detect the edge of a polygon in graphics applications.

To support system diagnostics, the Am29332 has a special "Master-Slave" mode. To use this mode, two chips are connected in parallel, and hence receive the same instructions and data. The master chip is used for the normal data path. However, in the slave chip, all outputs becomes inputs. The slave compares the outputs of the master with its own internally generated result. If the two do not match, the slave will activate an error signal.

As a further diagnostic aid, byte-wise parity checking is performed at both the A and B data inputs. The "parity" signal is activated if an error is detected. Parity bits (one per byte) are generated for the 32-bit output bus.

FUNCTIONAL DESCRIPTION

A detailed description of each functional block is given in the following paragraphs.

64-Bit Funnel Shifter

The 64-bit funnel shifter is a combinatorial network. The 64-bit input is formed from a combination of the A and B inputs. This may be left-shifted by up to 31 bits before being used by the ALU. The output of the shifter is the most significant 32 bits of the result. The 64-bit shifter can be used on either the A or B operands to perform barrel shifts (either up or down) or rotates. The operation is controlled by positioning operands properly at the input of the 64-bit up-shifter.

The number "n" by which the operand is shifted comes from two sources: the microprogram memory via the P₀ – P₅ pins or the internal register (byte 0 of the status register), PR₀ – PR₅, as selected by an instruction bit.

In general, the 6-bit position input, P₀ – P₅, takes a 6-bit two's complement number representing upshifts from 0 to 31 places (positive numbers) or downshifts from 1 to 32 places (negative numbers).

Mask Generator

The mask generator logic provides the ability to generate the appropriate mask for an operand of given width and position. The generation of the mask depends upon two types of instructions. The first type has byte boundary aligned operands (widths of either 1, 2, 3 or 4 bytes) with the least significant bit aligned to bit 0. The width of an operand is specified by the byte width inputs (I₈ and I₇) as shown in Table 3. The second type of instruction has operands of variable width (1 to 32 bits) and position. The operand is specified by the width inputs (W₀ – W₄) and the position inputs (P₀ – P₅) indicating the least significant bit position of the operand. Thus, in this type of instruction the operand may or may not be least significant bit aligned. Depending upon the type of instruction, the mask generator first generates a fence of all zeros starting from the least significant bit with the width specified either by the byte width or the width input fields. This fence can be upshifted by up to 31 bits by the 32-bit mask shifter. Whenever the mask is moved up over the 32-bit boundary, it does not wrap around. Instead, ONE's are inserted from the least significant end. This configuration provides the ability to operate on a contiguous field located anywhere in a word, or across a word boundary.

The mask generator can be used as a pattern generator by allowing the mask to pass through ALU (by using the PASS-MASK instruction). For example, a single-bit wide mask can be generated and by shifting it up by different amounts can give walking ONE or walking ZERO patterns for memory tests.

TABLE 3.

| I ₈ | I ₇ | Width in Bytes |
|----------------|----------------|----------------|
| 0 | 0 | 4 |
| 0 | 1 | 1 |
| 1 | 0 | 2 |
| 1 | 1 | 3 |

Arithmetic and Logical Unit

The ALU is a three input unit which uses the mask as a second or third operand in every instruction. The mask is used to merge two operands. For all selected bits (wherever the mask is 0), the desired operation specified by the instruction input is performed, and for all unselected bits either corresponding destination bits or zeros are passed through. The status of each operation (carry, negative, zero, overflow, link) applies to the result only over the specified width. For all byte aligned arithmetic and logical operations (first three quarters of the instruction set), the status is extracted from the appropriate

byte boundary. For all field operations (last quarter of the instruction set), the operand width is assumed to be 32 bits for status generation. The ZERO flag always indicates the status of all bits selected by the mask.

The actual width of the ALU is 34 bits. There are two extra bits used for the high speed signed and unsigned multiplication instructions. These two bits are automatically concatenated to the most-significant end of the ALU depending upon the width specified for the operation. Since the modified Booth algorithm requires a two-bit down-shift each cycle, these ALU bits generate the two most-significant bits of the partial product.

The ALU is capable of shifting data down by two bits for the multiplication algorithm, up by one bit for the divide algorithm and single-bit-up-shifts.

The processor is capable of performing BCD arithmetic on packed BCD numbers. The ALU has separate carry logic for BCD operations. This logic generates nibble carries (BCD digit carry) from propagate and generate signals formed from the A and B operands. In order to simplify the hardware while maintaining throughput, the BCD add and subtract operations are performed in two cycles. In the first cycle, ordinary binary addition or subtraction is performed and BCD nibble carries are generated. These are blocked from affecting the result at this stage, but are saved in the status register to be used later for BCD correction (NC₀ – NC₇). In the second cycle all BCD numbers are adjusted by examining the previously generated nibble carries. Since all the necessary information is stored in the status register, the processor can be interrupted after the first BCD cycle.

Priority Encoder

The priority encoder is provided to support floating-point arithmetic and some graphics primitives. The priority encoder takes up to 32 bits as input and generates a 5-bit wide binary code to indicate location of the most significant one in the operand. Input to the priority encoder comes from the input multiplexer, which masks all bits that the user does not want to participate in the prioritization. The priority encoder supports 8, 16, 24 and 32-bit operations depending upon the byte width specified. For each data type the priority encoder generates the appropriate binary weighted code. For example, when a byte width of two is specified ($l_7 - l_8 = 10$), the output of the encoder is zero when bit 15 is HIGH. However, if byte width of four is specified ($l_8 - l_7 = 00$), the output of encoder is 16 (decimal) if bit 15 is HIGH and bits 31 – 16 are LOW. Table 4 shows the output for each data type. If none of the inputs are HIGH or the most significant bit of the data type specified is HIGH, then the output is zero. The difference between these two cases is indicated by the Z-flag of the status register which is HIGH only if all inputs are zero.

Q-Register

The Q-register holds dividend and quotient bits for division, and multiplier and product bits for multiplication. During division, the contents of the Q-register are shifted left, a bit at a time, with quotient bits inserted into bit 0. During multiplication, the contents of the Q-register are shifted right, two bits at

a time, with product bits inserted into the most-significant two bits (according to the selected byte width). The Q-register may be loaded from the A or B inputs and read onto the Y bus.

Master-Slave Comparator

All ALU outputs (except MSERR) employ three-state buffers. The master-slave comparator compares the input and output of each buffer. Any difference causes the MSERR signal to be made true. In Slave mode, all output buffers are disabled. Outputs from a second ALU may then be connected to the equivalent pins of the first. The comparator in the slave will then detect any difference in the results generated by the two. When the Y bus is three-stated by making Output-Enable false, the Y bus master-slave comparators are disabled.

Parity Logic

For each byte of the DA and DB inputs there is an associated parity bit (8 in all). If a parity error is detected on any byte, the Parity-Error signal is made true. Four parity signals (one per byte) are also generated for the Y bus outputs. EVEN parity is employed for the Am29332.

Status Register

All necessary information about operations performed in the ALU is stored in the 32-bit wide status register after every microcycle. Since the register can be saved, an interrupt can occur after any cycle. The status register can be loaded from either the A or B input of the chip and can be read out on the Y bus for saving in an external register file. For loading, the byte width indicates how many bytes are to be updated. The status register is only updated if the HOLD input is inactive.

Each byte of the status register holds different types of information (see Figure 3). The least significant byte (bits 0 to 7) holds eight position bits (PR₀ – PR₇) for the data shifter. The two most significant bits are not used. The next most significant byte (bits 8 to 15) holds the 5-bit width field (WR₀ – WR₄) for the mask generator. The three most-significant bits of that byte (bits 13 to 15) are read-only bits that represent three different conditions extracted from the other bits of the status register. They are $\bar{C} + Z$, $N \oplus V$, and $(N \oplus V) + Z$ for bits 13, 14 and 15 respectively. These bits can be read on the Y₀ pin by the extract-status instruction. The next byte contains all the necessary information generated by an ALU operation. The least-significant four bits (bits 16 to 19) hold carry, negative, overflow and zero flags. Bit 20 holds link information for single bit shifts and bits 21 and 22 are used by the multiply and divide instructions. The M flag holds the multiplier bit for the modified Booth algorithm or it holds the sign comparison result for the divide algorithm. The S flag holds the sign of the partial remainder for unsigned division. Both the flags (M and S) are provided as a part of the status register so that multiply and divide instructions can be interrupted at microinstruction boundaries. The most significant byte of the status register holds nibble carries for BCD arithmetic. Since BCD arithmetic is performed in two cycles, the nibble carries are saved in the first cycle and used in the second cycle. Since all the information is stored, BCD instructions are also interruptible at the microinstruction boundary.

TABLE 4.

| Highest Priority Active Bit | Encoder Output |
|------------------------------|----------------|
| l7 – l8 = 00 (32-bit) | |
| None | 0 |
| 31 | 0 |
| 30 | 1 |
| 29 | 2 |
| 28 | 3 |
| . | . |
| . | . |
| 1 | 30 |
| 0 | 31 |
| l7 – l8 = 01 (8-bit) | |
| None | 0 |
| 7 | 0 |
| 6 | 1 |
| 5 | 2 |
| . | . |
| . | . |
| 1 | 6 |
| 0 | 7 |
| l7 – l8 = 10 (16-bit) | |
| None | 0 |
| 15 | 0 |
| 14 | 1 |
| 13 | 2 |
| 12 | 3 |
| . | . |
| . | . |
| 1 | 14 |
| 0 | 15 |
| l7 – l8 = 11 (24-bit) | |
| None | 0 |
| 23 | 0 |
| 22 | 1 |
| 21 | 2 |
| 20 | 3 |
| . | . |
| . | . |
| 1 | 22 |
| 0 | 23 |

Status0-7: Position Register

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| PR7 | PR6 | PR5 | PR4 | PR3 | PR2 | PR1 | PR0 |
| 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |

Status8-12: Width Register
 Status13: $\bar{C} + Z$
 Status14: $N \oplus V$
 Status15: $(N \oplus V) + Z$ } Read Only

| | | | | | | | |
|-----------|-----------|-------------|-----|-----|-----|-----|-----|
| SIGNED LE | SIGNED LT | UNSIGNED LE | WR4 | WR3 | WR2 | WR1 | WR0 |
| 15 | 14 | 13 | 12 | 11 | 10 | 9 | 8 |

Status16: Carry
 Status17: Negative
 Status18: Overflow
 Status19: Zero
 Status20: Link
 Status21: Multiply (and divide) Bit
 Status22: Sign Flag
 Status23: 0

| | | | | | | | |
|----|----|----|----|----|----|----|----|
| 0 | S | M | L | Z | V | N | C |
| 23 | 22 | 21 | 20 | 19 | 18 | 17 | 16 |

Status24-31: Nibble Carries

| | | | | | | | |
|-----|-----|-----|-----|-----|-----|-----|-----|
| NC7 | NC6 | NC5 | NC4 | NC3 | NC2 | NC1 | NC0 |
| 31 | 30 | 29 | 28 | 27 | 26 | 25 | 24 |

Note: Overflow is defined as follows:
 $V = (\text{carry in to MSB}) \oplus (\text{carry out of MSB})$

Figure 3. ALU Status Register Bit Assignment

Am29332 INSTRUCTION SET

Data Types

The Am29332 supports the following data types:

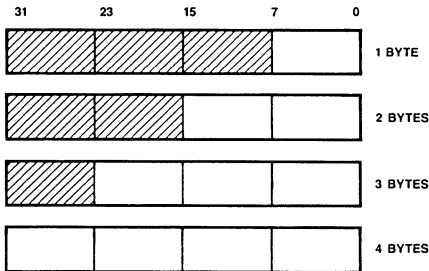
1. Integer
2. Binary-coded decimal
3. Variable-length bit field

The first two data types fall into the category of byte boundary aligned operands (Figure 4). The size of the operand could be 1 byte, 2 bytes, 3 bytes or 4 bytes. All operands are least significant bit (bit 0) aligned. The byte width is determined by bits l_8 and l_7 of the instruction as shown in Table 5.

TABLE 5.

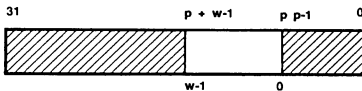
| l_8 | l_7 | Width in Bytes |
|-------|-------|----------------|
| 0 | 0 | 4 |
| 0 | 1 | 1 |
| 1 | 0 | 2 |
| 1 | 1 | 3 |

The third data type has operands of variable width (1 to 32 bits) as shown in Figure 4. The operand is specified by width inputs ($W_0 - W_4$) and position inputs ($P_0 - P_5$). The position inputs indicate the least significant bit position of the operand. Depending on bits l_8 and l_7 of the instruction, the width and position inputs can be selected from either the Status Register or the Width and Position Pins as shown in Table 6. A summary of the data types available is illustrated in Table 7.



TB000096

Byte Boundary Aligned Operands



TB000630

Variable-Length Bit Field

- p = Bit displacement of the least significant field with respect to bit 0.
 w = Width of bit field.

Figure 4. Data Types

TABLE 6.

| l_8 | l_7 | Position | | Width | |
|-------|-------|----------|-----|-------|-----|
| | | Pins | Reg | Pins | Reg |
| 0 | 0 | X | | X | |
| 0 | 1 | X | | | X |
| 1 | 0 | | X | X | |
| 1 | 1 | | X | | X |

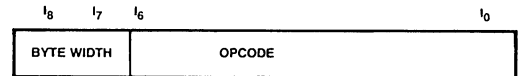
TABLE 7.

| Data Type | Size | Range | |
|-----------|-------------------------|--|-----------------|
| Integer | | Signed | Unsigned |
| 1 byte | 8 bits | -128 to +127 | 0 to 255 |
| 2 bytes | 16 bits | -2^{15} to $+2^{15}-1$ | 0 to $2^{16}-1$ |
| 3 bytes | 24 bits | -2^{23} to $2^{23}-1$ | 0 to $2^{24}-1$ |
| 4 bytes | 32 bits | -2^{31} to $2^{31}-1$ | 0 to $2^{32}-1$ |
| BCD | 1 to 4 bytes (8 digits) | Numeric, 2 digits per byte. Most-significant digit may be used for sign. | |
| Variable | 1 to 32 bits | Dependent on position and width inputs. | |

Instruction Format

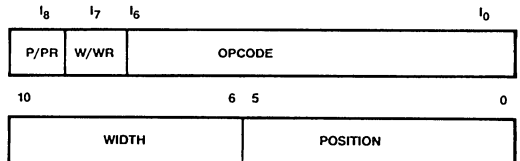
The Am29332 has two types of Instruction Formats:

1. Byte Boundary Aligned Instructions (FORMAT 1):



TB000098

2. Variable-Length Field Bit Instructions (FORMAT 2):



TB000099

For instructions that allow a field to be shifted up or down, $P_0 - P_5$ is a two's-complement number in the range -32 to $+31$ representing the direction and magnitude of the shift. For instructions that assume a fixed field position, $P_0 - P_4$ represent the position of the least-significant bit of the field and P_5 is ignored.

Instruction Classification

ALU instructions can be classified as follows:

A. Byte Boundary Aligned Operand Instructions:

1. Arithmetic
 - Binary, BCD
 - Multiply steps
 - Division steps (single and multiple precision)
2. Prioritize
3. Logical
4. Single-bit shifts
5. Data movement

B. Variable-Length Bit Field Operand Instructions:

1. N-bit shifts and rotates
2. Bit manipulations
3. Field logical operations (aligned, non-aligned, extract)
4. Mask generation

Three-fourths of the ALU instructions apply to operands that are byte boundary aligned. For these instructions, two orthogonal issues are the width of the operand (in bytes) and the contents of the high order unselected bytes on the Y bus. As mentioned earlier, the width of the operand is specified by l_6 and l_7 . With the exception of a few instructions, the unselected bytes are assigned values as follows: for single operand instructions, unselected bytes are passed unchanged from the source (A or B). For two operand instructions, unselected bytes are passed unchanged from the destination (B input).

In the last quarter of the instruction set, the width of the operand is from 1 to 32 bits (based on the width input) for field operations, 32 bits for N-bit shift operations and 1-bit for bit-oriented operations. In the case of field-aligned and single-bit operands, the position bits ($P_0 - P_4$) determine the least significant bit of the operand. In the case of N-bit shifts and field non-aligned operands, the position bits $P_0 - P_5$ is a 6-bit signed integer determining the magnitude and direction of the shift.

Flags

Byte-Aligned Instructions

The zero flag always looks only at the selected bytes:

$$Z \leftarrow (Y \text{ and bytemask (byte width)} = 0)$$

Similarly, $N \leftarrow \text{sign bit (Y, byte width)}$, where the function "sign-bit" returns bit 7, 15, 23, or 31 of the first argument for byte widths 01, 10, 11, or 00 respectively.

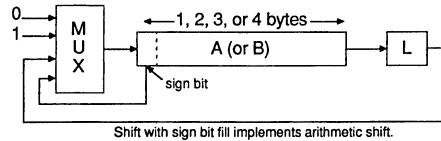
Also, $C \leftarrow \text{carry (byte width)}$ returns the carry from the appropriate byte boundary, and:

$$V \leftarrow \text{overflow (byte width)} = (\text{carry into MSB}) \oplus (\text{carry out of MSB})$$

returns the overflow from the appropriate byte boundary.

The link (L) flag is generally loaded with the bit moved out of the highest selected byte in the case of upshifts, or the bit moved out of the least significant byte for downshifts. Figure 5 shows the shift operation using link bit. Other status flags have specialized uses, explained in the following sections.

Shift Down:



Shift Up:

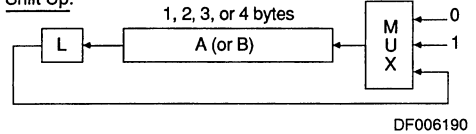


Figure 5. Upshift/Downshift Using Link Bit

Variable-Length Field Instruction:

Generally, only N and Z are affected. N takes the most-significant bit of the 32-bit result (i.e., $N \leftarrow Y_{31}$). Z detects zeros in the selected field of the result (i.e., $Z \leftarrow (Y \text{ and bitmask (position, width)} = 0)$).

Output Select

The Register Status pin, RS, may be used to switch the C, Z, N, V, and L output pins between the direct output of the ALU and the outputs of the corresponding bits in the status register. If the direct status output is selected, then for instructions that do not affect a particular flag (e.g., carry for logical arithmetic) that output will reflect the state of its corresponding bit in the status register. Similarly, when the HOLD signal is made HIGH, the C, Z, N, V and L pins will be made equal to the contents of the status register, regardless of the RS input.

INSTRUCTION SET SUMMARY

Operand Size: Variable Byte Width: 1, 2, 3, 4 Bytes

| Type | Operation | Data Type |
|-------------------|--|-------------------------------|
| Arithmetic | <ul style="list-style-type: none"> ● Increment by one, two, four ● Decrement by one, two, four ● Add, addc (carry = macro/micro) ● Sub, subr ● Subc, subrc (carry/borrow) ● BCD sum and difference correct steps | Binary Integer and BCD |
| | <ul style="list-style-type: none"> ● Negate (two's complement) ● Multiply steps (modified Booth) ● Divide steps (non-restoring) | (Signed and unsigned) |
| Prioritize | <ul style="list-style-type: none"> ● Prioritize | Binary |
| Logical | <ul style="list-style-type: none"> ● Not, OR, AND, XOR, XNOR, zero, sign | Binary |
| Single-Bit Shifts | <ul style="list-style-type: none"> ● Upshift with 0, 1, link fill ● Downshift with 0, 1, link, sign fill | (Single and double precision) |
| Data Movement | <ul style="list-style-type: none"> ● Zero extend ● Sign extend ● Pass-status, Q-Reg ● Load-status, Q-Reg ● Merge | Binary |

Operand Size: 32 Bits

| Type | Operation | Data Type |
|-------------------------------|--|-----------|
| N-Bit Shifts N-Bit Rotates | <ul style="list-style-type: none"> ● Upshift by 0 to 31 bits with 0 fill ● Downshift by 1 to 32 bits with 0, sign fill ● Rotate by 0 to 31 bits | Binary |

Operand Size: Single Bit

| Type | Operation | Data Type |
|------------------|---|-----------|
| Bit Manipulation | <ul style="list-style-type: none"> ● Extract ● Set ● Reset | Binary |

Operand Size: Variable Length Bitfield: 1 to 32 Bits

| Type | Operation | Data Type |
|---|--|-----------|
| Field Logical (aligned and non-aligned) | <ul style="list-style-type: none"> ● Not, OR, XOR, AND, extract, insert | Binary |
| Mask | <ul style="list-style-type: none"> ● Pass-mask | Binary |

INSTRUCTION SET GLOSSARY
(Sorted by Opcode in Hex Notation)

| Opcode | Name | Opcode | Name | Opcode | Name | Opcode | Name |
|--------|-----------|--------|-----------|--------|-------------|--------|-------------|
| 00 | ZERO-EXTA | 20 | DN1-0F-A | 40 | AND | 60 | NB-SN-SHA |
| 01 | ZERO-EXTB | 21 | DN1-0F-B | 41 | XNOR | 61 | NB-SN-SHB |
| 02 | SIGN-EXTA | 22 | DN1-0F-AQ | 42 | ADD | 62 | NB-0F-SHA |
| 03 | SIGN-EXTB | 23 | DN1-0F-BQ | 43 | ADDC | 63 | NB-0F-SHB |
| 04 | PASS-STAT | 24 | DN1-1F-A | 44 | SUB | 64 | NBROT-A |
| 05 | PASS-Q | 25 | DN1-1F-B | 45 | SUBC | 65 | NBROT-B |
| 06 | LOADQ-A | 26 | DN1-1F-AQ | 46 | SUBR | 66 | EXTBIT-A |
| 07 | LOADQ-B | 27 | DN1-1F-BQ | 47 | SUBRC | 67 | EXTBIT-B |
| 08 | NOT-A | 28 | DN1-LF-A | 48 | SUM-CORR-A | 68 | SETBIT-A |
| 09 | NOT-B | 29 | DN1-LF-B | 49 | SUM-CORR-B | 69 | SETBIT-B |
| 0A | NEG-A | 2A | DN1-LF-AQ | 4A | DIFF-CORR-A | 6A | RSTBIT-A |
| 0B | NEG-B | 2B | DN1-LF-BQ | 4B | DIFF-CORR-B | 6B | RSTBIT-B |
| 0C | PRIOR-A | 2C | DN1-AR-A | 4C | - | 6C | SETBIT-STAT |
| 0D | PRIOR-B | 2D | DN1-AR-B | 4D | - | 6D | RSTBIT-STAT |
| 0E | MERGEA-B | 2E | DN1-AR-AQ | 4E | SDIVFIRST | 6E | NOTF-AL-B |
| 0F | MERGED-B | 2F | DN1-AR-BQ | 4F | UDIVFIRST | 6F | PASSF-AL-B |
| 10 | DECR-A | 30 | UP1-0F-A | 50 | SDIVSTEP | 70 | NOTF-A |
| 11 | DECR-B | 31 | UP1-0F-B | 51 | SDIVLAST1 | 71 | NOTF-AL-A |
| 12 | INCR-A | 32 | UP1-0F-AQ | 52 | MPDIVSTEP1 | 72 | PASSF-A |
| 13 | INCR-B | 33 | UP1-0F-BQ | 53 | MPSDIVSTEP3 | 73 | PASSF-AL-A |
| 14 | DECR2-A | 34 | UP1-1F-A | 54 | UDIVSTEP | 74 | ORF-A |
| 15 | DECR2-B | 35 | UP1-1F-B | 55 | UDIVLAST | 75 | ORF-AL-A |
| 16 | INCR2-A | 36 | UP1-1F-AQ | 56 | MPDIVSTEP2 | 76 | XORF-A |
| 17 | INCR2-B | 37 | UP1-1F-BQ | 57 | MPUDIVSTEP3 | 77 | XORF-AL-A |
| 18 | DECR4-A | 38 | UP1-LF-A | 58 | REMCORR | 78 | ANDF-A |
| 19 | DECR4-B | 39 | UP1-LF-B | 59 | QUOCORR | 79 | ANDF-AL-A |
| 1A | INCR4-A | 3A | UP1-LF-AQ | 5A | SDIVLAST2 | 7A | EXTF-A |
| 1B | INCR4-B | 3B | UP1-LF-BQ | 5B | UMULFIRST | 7B | EXTF-B |
| 1C | LDSTAT-A | 3C | ZERO | 5C | UMULSTEP | 7C | EXTF-AB |
| 1D | LDSTAT-B | 3D | SIGN | 5D | UMULLAST | 7D | EXTF-BA |
| 1E | - | 3E | OR | 5E | SMULSTEP | 7E | EXTBIT-STAT |
| 1F | - | 3F | XOR | 5F | SMULFIRST | 7F | PASS-MASK |

TABLE 6-1. DATA MOVEMENT INSTRUCTIONS

| Mnemonics | Code | Description | Y Output | | Status | | | | | | |
|-----------|------|----------------|----------|-----------|--------|---|---|---|---|---|---|
| | | | Unsel | Sel | S | M | L | Z | V | N | C |
| ZERO-EXTA | 00 | Zero Extend | 0 | A | | | | * | | * | |
| ZERO-EXTB | 01 | | 0 | B | | | | * | | * | |
| SIGN-EXTA | 02 | Sign Extend | Sign | A | | | | * | | * | |
| SIGN-EXTB | 03 | | Sign | B | | | | * | | * | |
| MERGEA-B | 0E | Merge A with B | B | A Merge B | | | | * | | * | |
| MERGEB-A | 0F | Merge B with A | A | B Merge A | | | | * | | * | |

TABLE 6-2. DATA MOVEMENT INSTRUCTIONS

| Mnemonics | Code | Description | Y Output | | Status Register | Status | | | | | | |
|-----------|------|----------------------|----------|-----|-----------------|--------|---|---|---|---|---|---|
| | | | Unsel | Sel | | S | M | L | Z | V | N | C |
| PASS-STAT | 04 | Pass Status Register | B | S | | | | | | | | |
| LDSTAT-A | 1C | Load Status Register | S | A | A | + | + | + | + | + | + | + |
| LDSTAT-B | 1D | | S | B | B | + | + | + | + | + | + | + |

TABLE 6-3. DATA MOVEMENT INSTRUCTIONS

| Mnemonics | Code | Description | Y Output | | Q Register | Status | | | | | | |
|-----------|------|-----------------|----------|-----|------------|--------|---|---|---|---|---|---|
| | | | Unsel | Sel | | S | M | L | Z | V | N | C |
| PASS-Q | 05 | Pass Q Register | B | Q | | | | | | | | |
| LOADQ-A | 06 | Load Q | Q | A | A | | | | * | | * | |
| LOADQ-B | 07 | | Q | B | B | | | | * | | * | |

Note: 1. These instructions use the byte aligned instruction format (FORMAT 1).

- Legend: Unsel = Unselected Byte(s)
 Sel = Selected Byte(s)
 A = A Input
 B = B Input
 Q = Q Register
 + = Updated only if byte width is 3 or 4
 * = Updated

Examples:

- 2, ZERO EXTB Pass lower two bytes of B to Y with zero fill on upper two bytes
 0, LOADQ-A Load all four bytes of A into Q Register pass updated Q Resistor to Y

TABLE 7. LOGICAL INSTRUCTIONS

| Mnemonics | Code | Description | Y Output | | Status | | | | | | | |
|-----------|------|------------------|----------|-------------------|--------|---|---|---|---|---|---|--|
| | | | Unsel | Sel | S | M | L | Z | V | N | C | |
| NOT-A | 08 | One's Complement | A | \bar{A} | | | | * | | * | | |
| NOT-B | 09 | | B | \bar{B} | | | | * | | * | | |
| ZERO | 3C | Pass Zero | B | 0 | | | | 1 | | 0 | | |
| SIGN | 3D | Pass Sign | B | $0(N=0); -1(N=1)$ | | | | N | | | | |
| OR | 3E | OR | B | A OR B | | | | * | | * | | |
| XOR | 3F | EXOR | B | A XOR B | | | | * | | * | | |
| AND | 40 | AND | B | A AND B | | | | * | | * | | |
| XNOR | 41 | XNOR | B | A XNOR B | | | | * | | * | | |

Note: 1. These instructions use the byte aligned instruction format (FORMAT 1).

Legend: Unsel = Unselected Byte(s)
 Sel = Selected Byte(s)
 A = A Input
 B = B Input
 Q = Q Register
 * = Updated

Examples:

- 2, NOT-A Complement low order two bytes of A and output to Y with high order two bytes of A uncomplemented.
- 1, AND AND first byte of A and B. Output to Y with high three bytes of B.

TABLE 8-1. SINGLE-BIT SHIFT INSTRUCTIONS (SINGLE PRECISION)

| Mnemonics | Code | Description | Y Output | | Status | | | | | | | |
|-----------|------|----------------------|----------|------------------------------|--------|---|---|---|---|---|---|--|
| | | | Unsel | Sel | S | M | L | Z | V | N | C | |
| DN1-0F-A | 20 | Downshift, Zero Fill | A | $Y_i = A_{i+1}, Y_{msb} = 0$ | | | * | * | | * | | |
| DN1-0F-B | 21 | | B | $Y_i = B_{i+1}, Y_{msb} = 0$ | | | * | * | | * | | |
| DN1-1F-A | 24 | Downshift, One Fill | A | $Y_i = A_{i+1}, Y_{msb} = 1$ | | | * | * | | * | | |
| DN1-1F-B | 25 | | B | $Y_i = B_{i+1}, Y_{msb} = 1$ | | | * | * | | * | | |
| DN1-LF-A | 28 | Downshift, Link Fill | A | $Y_i = A_{i+1}, Y_{msb} = L$ | | | * | * | | * | | |
| DN1-LF-B | 29 | | B | $Y_i = B_{i+1}, Y_{msb} = L$ | | | * | * | | * | | |
| DN1-AR-A | 2C | Downshift, Sign Fill | A | $Y_i = A_{i+1}, Y_{msb} = N$ | | | * | * | | * | | |
| DN1-AR-B | 2D | | B | $Y_i = B_{i+1}, Y_{msb} = N$ | | | * | * | | * | | |
| UP1-0F-A | 30 | Upshift, Zero Fill | A | $Y_i = A_{i-1}, Y_0 = 0$ | | | * | * | * | * | | |
| UP1-0F-B | 31 | | B | $Y_i = B_{i-1}, Y_0 = 0$ | | | * | * | * | * | | |
| UP1-1F-A | 34 | Upshift, One Fill | A | $Y_i = A_{i-1}, Y_0 = 1$ | | | * | * | * | * | | |
| UP1-1F-B | 35 | | B | $Y_i = B_{i-1}, Y_0 = 1$ | | | * | * | * | * | | |
| UP1-LF-A | 38 | Upshift, Link Fill | A | $Y_i = A_{i-1}, Y_0 = L$ | | | * | * | * | * | | |
| UP1-LF-B | 39 | | B | $Y_i = B_{i-1}, Y_0 = L$ | | | * | * | * | * | | |

Note: 1. These instructions use the byte aligned instruction format (FORMAT 1).

Example:

- 2, UP1-1F-A Shift lower two bytes of A up one bit. Set LSB to 1. Fill unselected bytes to upper two bytes of A.

TABLE 8-2. SINGLE-BIT SHIFT INSTRUCTIONS (DOUBLE PRECISION)

| Mnemonics | Code | Description | Y Output & Q Register | Status | | | | | | |
|-----------|------|----------------------|-----------------------|--------|---|---|---|---|---|---|
| | | | Selected Bytes | S | M | L | Z | V | N | C |
| DN1-0F-AQ | 22 | Downshift, Zero Fill | 0 → A → Q 2) | | | * | * | | * | |
| DN1-0F-BQ | 23 | | 0 → B → Q 3) | | | * | * | | * | |
| DN1-1F-AQ | 26 | Downshift, One Fill | 1 → A → Q 2) | | | * | * | | * | |
| DN1-1F-BQ | 27 | | 1 → B → Q 3) | | | * | * | | * | |
| DN1-LF-AQ | 2A | Downshift, Link Fill | L → A → Q 2) | | | * | * | | * | |
| DN1-LF-BQ | 2B | | L → B → Q 3) | | | * | * | | * | |
| DN1-AR-AQ | 2E | Downshift, Sign Fill | N → A → Q 2) | | | * | * | | * | |
| DN1-AR-BQ | 2F | | N → B → Q 3) | | | * | * | | * | |
| UP1-0F-AQ | 32 | Upshift, Zero Fill | A ← Q ← 0 2) | | | * | * | * | * | |
| UP1-0F-BQ | 33 | | B ← Q ← 0 3) | | | * | * | * | * | |
| UP1-1F-AQ | 36 | Upshift, One Fill | A ← Q ← 1 2) | | | * | * | * | * | |
| UP1-1F-BQ | 37 | | B ← Q ← 1 3) | | | * | * | * | * | |
| UP1-LF-AQ | 3A | Upshift, Link Fill | A ← Q ← L 2) | | | * | * | * | * | |
| UP1-LF-BQ | 3B | | B ← Q ← L 3) | | | * | * | * | * | |

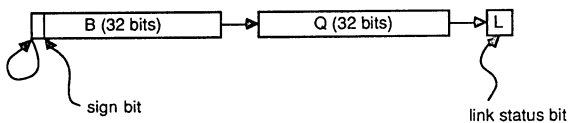
Notes: 1. These instructions use the byte aligned instruction format (FORMAT 1).
 2. Y Unselected byte from A, Q Unselected byte unchanged.
 3. Y Unselected byte from B, Q Unselected byte unchanged.

Legend: Unsel = Unselected Byte(s)
 Sel = Selected Byte(s)
 A = A Input
 B = B Input
 Q = Q Register
 * = Updated

Example:

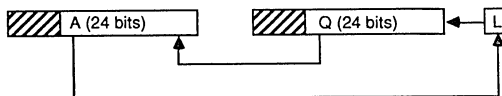
0, DN1-AR-BQ

Shift 64 bits (all 32 bits of both B and Q) down by one bit. LSB of B fills MSB of Q. MSB of B set to sign bit (bit N of status register).



3, UP1-LF-AQ

Shift 48 bits (24-bits of A and 24-bits of Q) up by one bit. MSB of 24-bit Q fills LSB of A. MSB of 24-bit A sets link status bit. LSB of Q is filled with original link value.



DF006200

TABLE 9. PRIORITIZE INSTRUCTIONS

| Mnemonics | Code | Description | Y Output | Status | | | | | | |
|-----------|------|----------------|---------------------------|--------|---|---|---|---|---|---|
| | | | | S | M | L | Z | V | N | C |
| PRIOR-A | 0C | Prioritization | Location of Highest 1 Bit | | | | * | | | |
| PRIOR-B | 0D | | | | | | * | | | |

- Notes: 1. These instructions use the byte aligned instruction format (FORMAT 1).
 2. Priority also loaded into STATUS <7:0>
 3. Refer to Table 4.

Legend: A = A Input
 B = B Input
 Q = Q Register
 * = Updated

Example:

3, PRIOR-A Value placed on Y is 2



Assume A is

| | | | |
|----------|----------|----------|----------|
| 01001011 | 00100010 | 00000000 | 00000000 |
|----------|----------|----------|----------|

TABLE 10-1. ARITHMETIC INSTRUCTIONS

| Mnemonics | Code | Description | Y Output | | Status | | | | | | |
|-----------|------|-------------------|----------|---------------|--------|---|---|---|---|---|---|
| | | | Unsel | Sel | S | M | L | Z | V | N | C |
| NEG-A | 0A | Two's Complement | A | $\bar{A} + 1$ | | | | * | * | * | * |
| NEG-B | 0B | | B | $\bar{B} + 1$ | | | | * | * | * | * |
| INCR-A | 12 | Increment by One | A | A + 1 | | | | * | * | * | * |
| INCR-B | 13 | | B | B + 1 | | | | * | * | * | * |
| INCR2-A | 16 | Increment by Two | A | A + 2 | | | | * | * | * | * |
| INCR2-B | 17 | | B | B + 2 | | | | * | * | * | * |
| INCR4-A | 1A | Increment by Four | A | A + 4 | | | | * | * | * | * |
| INCR4-B | 1B | | B | B + 4 | | | | * | * | * | * |
| DECR-A | 10 | Decrement by One | A | A - 1 | | | | * | * | * | * |
| DECR-B | 11 | | B | B - 1 | | | | * | * | * | * |
| DECR2-A | 14 | Decrement by Two | A | A - 2 | | | | * | * | * | * |
| DECR2-B | 15 | | B | B - 2 | | | | * | * | * | * |
| DECR4-A | 18 | Decrement by Four | A | A - 4 | | | | * | * | * | * |
| DECR4-B | 19 | | B | B - 4 | | | | * | * | * | * |

- Notes: 1. These instructions use the byte aligned instruction format (FORMAT 1).
 2. Borrow, rather than carry, is generated if BOROW is HIGH (borrow = carry).
 3. Nibble bits are set by these instructions. NEG-A (or NEG-B) and DIFF-CORR may be used to form 10's complement of a BCD number. Use SUM-CORR (for increment) or DIFF-CORR (for decrement) to increment or decrement a BCD number.

Legend: Unsel = Unselected Byte(s)
 Sel = Selected Byte(s)
 A = A Input
 B = B Input
 Q = Q Register
 * = Updated

Example:

2, DECR4-A Decrement lower two bytes of A by 4

TABLE 10-2. ARITHMETIC INSTRUCTIONS

| Mnemonics | Code | Description | Y Output | | Status | | | | | | |
|-------------|------|-------------------------------------|----------|-----------------------------|--------|---|---|---|---|---|---|
| | | | Unsel | Sel | S | M | L | Z | V | N | C |
| ADD | 42 | Add | B | A + B | | | | * | * | * | * |
| ADDC | 43 | Add with Carry | B | A + B + C 6) | | | | * | * | * | * |
| SUB | 44 | Subtract | B | A + \bar{B} + 1 | | | | * | * | * | * |
| SUBR | 46 | | B | B + \bar{A} + 1 | | | | * | * | * | * |
| SUBC | 45 | Subtract with Carry | B | A + \bar{B} + 1 + C 2) 6) | | | | * | * | * | * |
| SUBRC | 47 | | B | B + \bar{A} + 1 + C 2) 6) | | | | * | * | * | * |
| SUM-CORR-A | 48 | Correct BCD Nibbles for Addition | A | Corrected A 3) | | | | * | * | * | * |
| SUM-CORR-B | 49 | | B | Corrected B 3) | | | | * | * | * | * |
| DIFF-CORR-A | 4A | Correct BCD Nibbles for Subtraction | A | Corrected A 3) | | | | * | * | * | * |
| DIFF-CORR-B | 4B | | B | Corrected B 3) | | | | * | * | * | * |

- Notes: 1. These instructions use the byte aligned instruction format (FORMAT 1).
 2. BOROW is LOW. For subtract operations, a borrow rather than a carry is stored in STATUS if BOROW is HIGH. Carry is always generated for ADD regardless of BOROW.
 3. First, the nibble carries NC₀ - NC₇ are tested. Any nibble carry/borrow that is set to 1 generates "6" internally as a correction word and then the correction word is added (SUM-CORR-) or subtracted (DIFF-CORR-) from the operand. NC₀ - NC₇ are not affected by this operation.
 4. Use SUM-CORR or DIFF-CORR to add or subtract a BCD number.
 5. Use ADDC, SUBC, or SUBRC to perform operations on integers longer than 32 bits.
 6. Carry bit is obtained from MCin if M/ \bar{m} is HIGH. Otherwise, carry is obtained from the C status bit.

Legend: Unsel = Unselected Byte(s)
 Sel = Selected Byte(s)
 A = A Input
 B = B Input
 Q = Q Register

* = Updated only if byte width is 3 or 4

Example:

0, ADD Add two 32-bit two's-complement integers

TABLE 11-1. DIVIDE INSTRUCTIONS (Aligned Format)

| Name | I ₆ - I ₀ Code | Description | Source for Unselected Bytes | Output | Status | | | | | | |
|------------------------------------|--------------------------------------|---------------------------------------|-----------------------------|--------|--------|---|---|---|---|---|---|
| | | | | | S | M | L | Z | V | N | C |
| Signed Divide Steps | | | | | | | | | | | |
| SDIVFIRST | 4 E | First Instruction for Signed Divide | B | Y, Q | * | * | * | * | * | * | |
| SDIVSTEP | 5 0 | Iterate Step (#bits - 1 times) | B | Y, Q | | * | * | * | * | * | * |
| SDIVLAST1 | 5 1 | Last Divide Instruction Unless | B | Y, Q | | * | | * | | * | * |
| SDIVLAST2 | 5 A | Dividend & Remainder Negative | B | Y | | | | * | | | |
| Unsigned Divide Steps | | | | | | | | | | | |
| UDIVFIRST | 4 F | First Instruction for Unsigned Divide | B | Y, Q | | | * | * | * | * | |
| UDIVSTEP | 5 4 | Iterate Step (#bits - 1 times) | B | Y, Q | * | * | * | * | | | * |
| UDIVLAST | 5 5 | Last Instruction | B | Y, Q | 0 | * | | * | | * | * |
| Multiprecision Divide Steps | | | | | | | | | | | |
| MPDIVSTEP1 | 5 2 | First Instruction | B | Y, Q | | | | | | | |
| MPDIVSTEP2 | 5 6 | Executed 0 Times for Double | B | Y, Q | | | | | | | |
| MPSDIVSTEP3 | 5 3 | Last Instruction of Inner Loop | B | Y, Q | | | | | | | |
| MPUDIVSTP3 | 5 7 | Used for Unsigned Divide | B | Y, Q | | | | | | | |
| Correction Steps | | | | | | | | | | | |
| REMCORR | 5 8 | Correct Remainder After Divide | B | Y | | | | | | | * |
| QUOCORR | 5 9 | Correct Quotient After Divide | B | Y | | | | | * | | * |

TABLE 11-2. EXAMPLE CODING FORM (Signed Division)

| Am29331 | | | | Am29332 | | | | Am29334 | | | Am29332 Y-Out |
|---------|--------|-------------|-----------|---------|------------|-------|----------|---------|------|-------|---------------|
| OP | Branch | Cond Select | Multi Sel | B/W | OP | Width | Position | A-IN | B-IN | Y-OUT | |
| CONT | | | | 2 | LOADQ-A | | | R2 | | | 1 |
| CONT | | | | 0 | SIGN | | | | | R3 | 0 |
| FOR_D | 15 | | | 2 | SDIVFIRST | | | R4 | R3 | R3 | 0 |
| DJMP_S | | | | 2 | SDIVSTEP | | | R4 | R3 | R3 | 0 |
| CONT | | | | 2 | SDIVLAST1 | | | R4 | R3 | R3 | 0 |
| BRCC_D | DONE | Z | | | | | | | | | 1 |
| CONT | | | | 2 | SDIVLAST2A | | | R4 | R3 | R3 | 0 |
| CONT | | | | 2 | PASS-Q | | | | | R1 | 0 |
| CONT | | | | 2 | QUOCORR | | | | R1 | R1 | 0 |
| CONT | | | | 2 | REMCORR | | | R4 | R3 | R3 | 0 |

Note: Divisor in A, Dividend in A
Quotient in Q, Remainder in B

Legend: A = A Input
B = B Input
S = Status Register
Q = Q Register
R1 = Quotient
R2 = Dividend
R3 = Remainder
R4 = Divisor

TABLE 12-1. MULTIPLY INSTRUCTIONS (Aligned Format)

| Name | I ₆ - I ₀ Code | Description | Source for Unselected Bytes | Output | Status | | | | | | |
|--------------------------------|---|----------------------------------|-----------------------------------|------------------|--------|---|---|---|---|---|---|
| | | | | | S | M | L | Z | V | N | C |
| Signed Multiply Steps | | | | | | | | | | | |
| SMULFIRST | 5 F | First multiply instruction | B | Y ⁽¹⁾ | | | | | | | |
| SMULSTEP | 5 E | Iterate step (#bits/2 - 1 steps) | B | Y ⁽¹⁾ | | | | | | | |
| Unsigned Multiply Steps | | | | | | | | | | | |
| UMULFIRST | 5 B | First multiply instruction | B | Y ⁽¹⁾ | | * | | | | | |
| UMULSTEP | 5 C | Iterate step (#bits/2 - 1 steps) | B | Y ⁽¹⁾ | | * | | | | | |
| UMULLAST | 5 D | Last multiply instruction | B | Y ⁽¹⁾ | | | | * | | | |

TABLE 12-2. EXAMPLE CODING FORM (Unsigned Multiply)

| Am29331 | | | | Am29332 | | | | Am29334 | | | Am29332 Y-Out |
|---------|------------------|-------------|-----------|---------|------------|-------|----------|---------|------|-------|---------------|
| OP | Branch | Cond Select | Multi Sel | B/W | OP | Width | Position | A-IN | B-IN | Y-OUT | |
| CONT | | | | 3 | ZERO | | | | R3 | R3 | 0 |
| CONT | | | | 3 | LOADQ-A | | | R1 | | | 1 |
| FOR_D | 11 ₁₀ | | | 3 | ULMULFIRST | | | R2 | R3 | R3 | 0 |
| DJMP_S | | | | 3 | UMULSTEP | | | R2 | R3 | R3 | 0 |
| CONT | | | | 3 | UMULLAST | | | R2 | R3 | R3 | 0 |
| CONT | | | | 3 | PASS-Q | | | | | R4 | 0 |

Note: 1. Put ALU output in B.
 2. Multiplicand in A, Multiplier in Q
 Product (HIGH) in B, Product (LOW) in Q

Legend: A = A Input
 B = B Input
 S = Status Register
 Q = Q Register
 R1 = Multiplier
 R2 = Multiplicand
 R3 = Product (HIGH)
 R4 = Product (LOW)

TABLE 13. SHIFT/ROTATE INSTRUCTIONS

| Mnemonics | Code | Description | Y Output | Status | | | | | | |
|-----------|------|------------------------|----------------------------------|--------|---|---|---|---|---|---|
| | | | | S | M | L | Z | V | N | C |
| NB-OF-SHA | 62 | Field Shift, Zero Fill | $Y_{i+p} = A_i, 0$ 2) | | | | * | * | | |
| NB-OF-SHB | 63 | | $Y_{i+p} = B_i, 0$ 2) | | | | * | * | | |
| NB-SN-SHA | 60 | Field Shift, Sign Fill | $Y_{i+p} = A_i, N$ 2) | | | | * | * | | |
| NB-SN-SHB | 61 | | $Y_{i+p} = B_i, N$ 2) | | | | * | * | | |
| NBROT-A | 64 | Field Rotate | $Y_i = A_{(i-p)\text{mod}32}$ 3) | | | | * | * | | |
| NBROT-B | 65 | | $Y_i = B_{(i-p)\text{mod}32}$ 3) | | | | * | * | | |

- Notes: 1. These instructions use the field instruction format (FORMAT 2).
 2. "p" stands for bit displacement from P₀-P₅ or from PR₀-PR₅ (-32 ≤ p ≤ 31).
 If p is positive, Y_{p-1} to Y₀ are equal to the fill bit.
 If p is negative, Y₃₁ to Y_{31+p+1} are equal to the fill bit.
 3. The sign of the position input is ignored for this instruction and P₀-P₄ are treated as a positive magnitude for a circular upshift.

Legend: A = A Input
 B = B Input
 Q = Q Register
 * = Updated

Examples: *
 NB-OF-SHA,,4 Shift A up 4 bits and zero fill
 NB-OF-SHB,,-17 Shift B down 17 bits and sign fill

*Width field not used

TABLE 14-1. BIT-MANIPULATION INSTRUCTIONS

| Mnemonics | Code | Description | Y Output | | Status | | | | | | |
|-------------|------|-------------|----------|---|--------|---|---|---|---|---|---|
| | | | Unsel | Sel | S | M | L | Z | V | N | C |
| SETBIT-A | 68 | Bit Set | A | $Y_i = A_i, Y_p = 1$ | | | | * | * | | |
| SETBIT-B | 69 | | B | $Y_i = B_i, Y_p = 1$ | | | | * | * | | |
| RSTBIT-A | 6A | Bit Reset | A | $Y_i = A_i, Y_p = 0$ | | | | * | * | | |
| RSTBIT-B | 6B | | B | $Y_i = B_i, Y_p = 0$ | | | | * | * | | |
| EXTBIT-A | 66 | Bit Extract | 0 | if p > 0, Y ₀ = A _p 2) if p < 0, Y ₀ = A _p | | | | * | * | | |
| EXTBIT-B | 67 | | 0 | if p > 0, Y ₀ = B _p 2) if p < 0, Y ₀ = B _p | | | | * | * | | |
| EXTBIT-STAT | 7E | | 0 | if p > 0, Y ₀ = S _p 2) if p < 0, Y ₀ = S _p | | | | * | | | |

- Notes: 1. These instructions use the field instruction format (FORMAT 2).
 2. Y₃₁ to Y₁ are set to zero. "p" stands for the bit displacement from P₀-P₄ or from PR₀-PR₅. The sign of the position input is ignored.

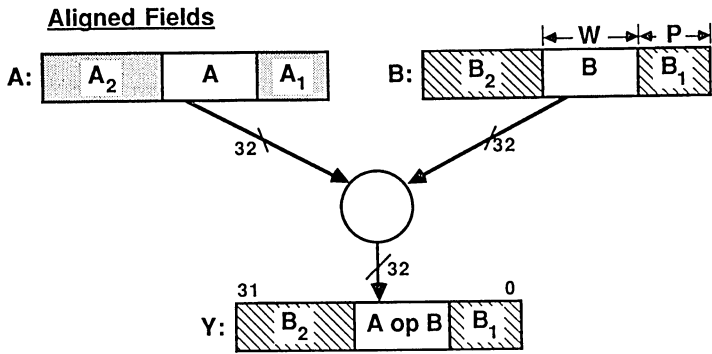
TABLE 14-2. BIT-MANIPULATION INSTRUCTIONS

| Mnemonics | Code | Description | Status Register | Y Output | Status | | | | | | |
|-------------|------|----------------|--------------------|----------|--------|---|---|---|---|---|---|
| | | | | | S | M | L | Z | V | N | C |
| SETBIT-STAT | 6C | Status Bit Set | S _p = 1 | S | * | * | * | * | * | * | * |
| RSTBIT-STAT | 6D | | S _p = 0 | S | * | * | * | * | * | * | * |

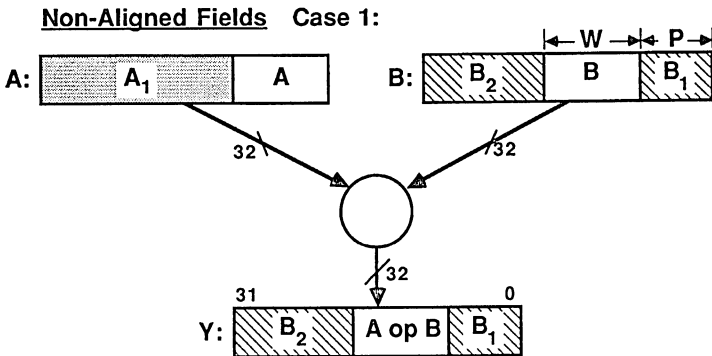
- Notes: 1. These instructions use the Field instruction format (FORMAT 2).
 2. "p" stands for the bit displacement from P₀-P₅ or from PR₀-PR₅.

Legend: Unsel = Unselected field
 Sel = Selected field
 A = A Input
 B = B Input
 Q = Q Register
 * = Updated

Examples:
 RSTBIT-B,,3 3rd bit is set to 0 in B
 EXTBIT-STAT,,-4 4th bit in status register is extracted and inverted.

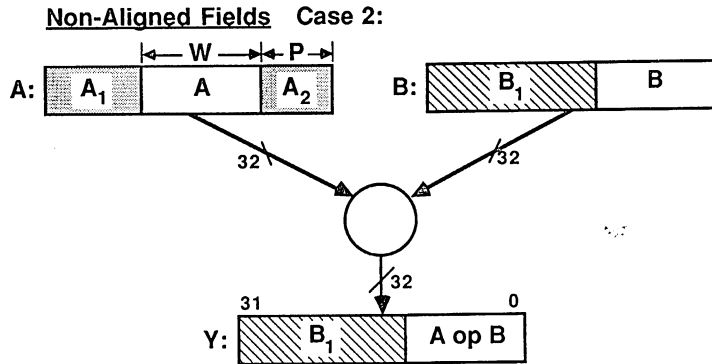


LD000140



If position (P_0 - P_5) ≥ 0 , A is LSB aligned
Width (W_0 - W_4) = 1 to 32

LD000151



If position (P_0 - P_5) < 0 , B is LSB aligned
Width (W_0 - W_5) = 1 to 32

LD000161

Figure 6. Field Logical Operations

TABLE 15. FIELD LOGICAL INSTRUCTIONS

| Mnemonics | Code | Description | | Y Output | | Status | | | | | |
|------------|------|------------------|-------|----------|---|--------|---|---|---|---|---|
| | | | | Unsel | Sel | S | M | L | Z | V | N |
| PASSF-AL-A | 73 | Field Pass | 3) | B | $Y_i = A_i$ | | | | * | * | |
| PASSF-AL-B | 6F | | 3) | B | $Y_i = B_i$ | | | | * | * | |
| PASSF-A | 72 | | 4) | B | if $p \geq 0$, $Y_i = A_{i-p}$ if $p < 0$, $Y_{i- p } = A_i$ | | | | * | * | |
| NOTF-AL-A | 71 | Field Complement | 3) | B | $Y_i = \bar{A}_i$ | | | | * | * | |
| NOTF-AL-B | 6E | | 3) | B | $Y_i = \bar{B}_i$ | | | | * | * | |
| NOTF-A | 70 | | 4) | B | if $p \geq 0$, $Y_i = \bar{A}_{i-p}$ if $p < 0$, $Y_{i- p } = \bar{A}_i$ | | | | * | * | |
| ORF-AL-A | 75 | Field OR | 3) | B | $Y_i = A_i \text{ OR } B_i$ | | | | * | * | |
| ORF-A | 74 | | 4) | B | if $p \geq 0$, $Y_i = A_{i-p} \text{ OR } B_i$ if $p < 0$, $Y_{i- p } = A_i \text{ OR } B_{i- p }$ | | | | * | * | |
| XORF-AL-A | 77 | Field XOR | 3) | B | $Y_i = A_i \text{ XOR } B_i$ | | | | * | * | |
| XORF-A | 76 | | 4) | B | if $p \geq 0$, $Y_i = A_{i-p} \text{ XOR } B_i$ if $p < 0$, $Y_{i- p } = A_i \text{ XOR } B_{i- p }$ | | | | * | * | |
| ANDF-AL-A | 79 | Field AND | 3) | B | $Y_i = A_i \text{ AND } B_i$ | | | | * | * | |
| ANDF-A | 78 | | 4) | B | if $p \geq 0$, $Y_i = A_{i-p} \text{ AND } B_i$ if $p < 0$, $Y_{i- p } = A_i \text{ AND } B_{i- p }$ | | | | * | * | |
| EXTF-A | 7A | Field Extract | 4) 5) | 0 | if $p \geq 0$, $Y_i = A_{i-p}$ if $p < 0$, $Y_{i- p } = A_i$ | | | | * | * | |
| EXTF-B | 7B | | | 0 | if $p \geq 0$, $Y_i = B_{i-p}$ if $p < 0$, $Y_{i- p } = B_i$ | | | | * | * | |
| EXTF-AB | 7C | | 0 | 6) | | | | * | * | | |
| EXTF-BA | 7D | | 0 | 7) | | | | * | * | | |

- Notes: 1. These instructions use the field instruction format (FORMAT 2).
 2. $p \leq i \leq p + w - 1$. "p" stands for position displacement from $P_0 - P_5$ or from $PR_0 - PR_5$ and "w" for the width of the bit field from $W_0 - W_4$ or $WR_0 - WR_4$. Whenever $p + w > 32$, operation takes place only over the portion of the field up to the end of the word. No wraparound occurs.
 3. This instruction uses the aligned format (see Figure 6).
 4. This instruction uses the unaligned field format (see Figure 6).
 $p \geq 0$: Case 1
 $p < 0$: Case 2
 5. If p is positive, the input is LSB aligned and Y output aligned at position.
 If p is negative, the input is aligned at |p| and Y output at LSB.
 6. Firstly, the concatenation of A(High Word) and B(Low Word) is rotated by the amount specified by the position (p). If p is positive, left-rotate is performed. If p is negative, right-rotate is performed. Secondly, the least significant bits on the Y output specified by the width (w) are extracted.
 7. Same as 6) except that B input is taken as a high word and A input as a low word.

Legend: Unsel = Unselected Field
 Sel = Selected Field
 A = A Input
 B = B Input
 Q = Q Register
 * = Updated

For all examples, assume STATUS (7:0) is -7 and STATUS (12:8) is 3.

1. 0,PASSF-AL-B,11,20 Pass B to Y and test if B_{20} to B_{30} are all zero. Set Z status if so.

B: 10000000000000000101011100110100

Z set to 1 in this case

2. 3,XORF-A,, Exclusive-OR bits $A_7 - A_9$ with bits $B_0 - B_2$ and output to $Y_0 - Y_2$. Pass $B_3 - B_{31}$ to $Y_3 - Y_{31}$. Width and position values are obtained from STATUS(12:0).

A: 01101110001001000010111001101011

B: 00011100001010001100101001001001

$A_9 - 7 \oplus B_{2-0} = Y$: 00011100001010001100101001001101

TABLE 16. MASK INSTRUCTION

| Mnemonics | Code | Description | Y Output | | Status | | | | | | | |
|-----------|------|---------------|----------|-------------------|--------|---|---|---|---|---|---|--|
| | | | Unsel | Sel | S | M | L | Z | V | N | C | |
| PASS-MASK | 7F | Generate Mask | p_5 | $Y_i = \bar{p}_5$ | | | | | | | | |

Notes: 1. This instruction uses the field instruction format (FORMAT 2).

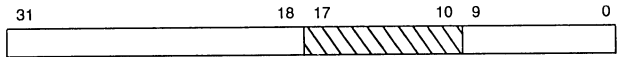
2. $p \leq i \leq p + w - 1$. "p" stands for the position displacement and "w" for the width of bit field.

- Legend: Unsel = Unselected Field
 Sel = Selected Field
 A = A Input
 B = B Input
 Q = Q Register
 * = Updated

Example:

Generates an 8-bit field mask pattern starting from bit position 10.

0, PASS-MASK, 8, 10



APPLICATIONS

Suggestions for Power and Ground Pin Connections

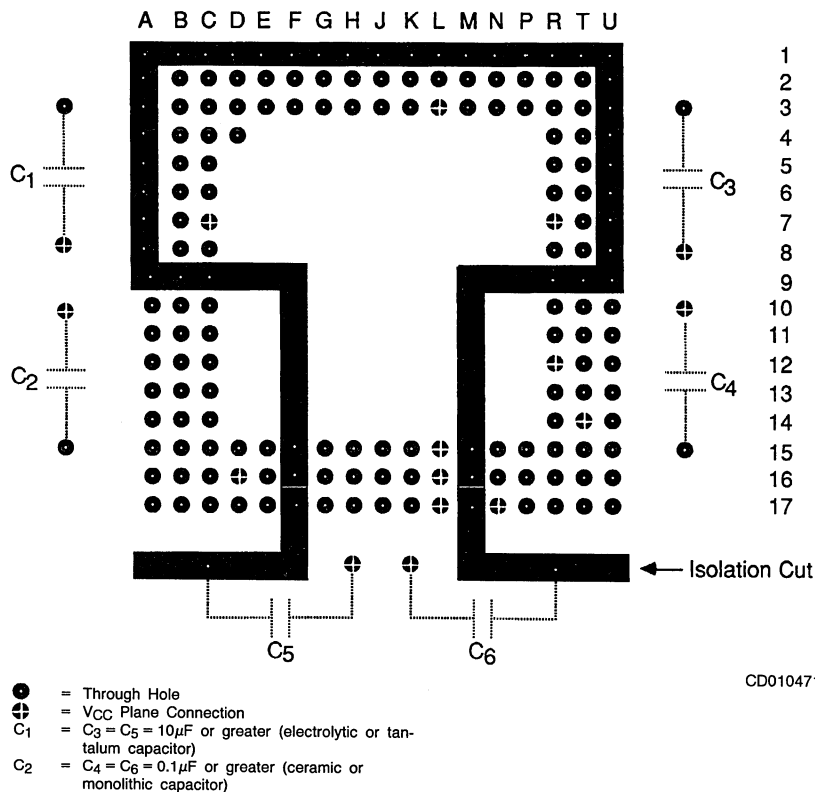
The Am29332 operates in an environment of fast signal rise times and substantial switching currents. Therefore, care must be exercised during circuit board design and layout, as with any high-performance component. The following is a suggested layout, but since systems vary widely in electrical configuration, an empirical evaluation of the intended layout is recommended.

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

Printed Circuit-Board Layout Suggestions

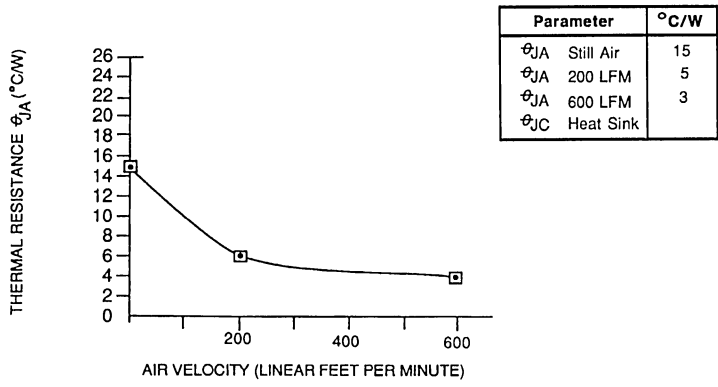
1. Use of a multi-layer PC board with separate power, ground, and signal planes is highly recommended.
2. All V_{CCE} and V_{CCT} pins should be connected to the V_{CC} plane. V_{CCT} pins should be isolated from V_{CCE} pins by means of a slot cut in the V_{CCE} plane; see Figure 7. By physically separating the V_{CCE} and V_{CCT} pins, coupled noise will be reduced.
3. All $GNDE$ and $GNDT$ pins should be connected directly to the ground plane.
4. The V_{CCT} pins should be decoupled to ground with a $0.1\text{-}\mu\text{F}$ ceramic capacitor and a $10\text{-}\mu\text{F}$ electrolytic capacitor, placed as closely to the Am29332 as is practical. V_{CCE} pins should be decoupled to ground in a similar manner.

A suggested layout is shown in Figure 7.



CD010471

Figure 7. Suggested Printed Circuit-Board Layout



OP002241

Figure 8. Am29332 Thermal Characteristics (Typical)

ABSOLUTE MAXIMUM RATINGS

| | |
|---|---------------------------------|
| Storage Temperature | -65 to +150°C |
| Temperature Under Bias - T _C | -55 to +125°C |
| Supply Voltage to Ground Potential | |
| Continuous | -0.5 to +7.0 V |
| DC Voltage Applied to Outputs | |
| for HIGH State | -0.5 V to +V _{CC} Max. |
| DC Input Voltage | -0.5 to +5.5 V |

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) Case Devices | |
| Temperature (T _C) | 0 to +85°C |
| Supply Voltage V _{CC} | +4.75 V to +5.25 V |

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

DC CHARACTERISTICS over operating range

| Parameter Symbol | Parameter Description | Test Conditions (Note 1) | | Min. | Max. | Units |
|------------------|--|--|--------------------------------|------|-------|-------|
| V _{OH} | Output HIGH Voltage | V _{CC} = 4.75 V, V _{IN} = V _{IH} or V _{IL} , I _{OH} = -1.2 mA | All Outputs | -2.4 | | Volts |
| V _{OL} | Output LOW Voltage | V _{CC} = 4.75 V, V _{IN} = V _{IH} or V _{IL} , I _{OL} = 8 mA | All Outputs | | 0.5 | Volts |
| V _{IH} | Input HIGH Level (Guaranteed Logic HIGH Voltage) | | All Inputs | 2.0 | | Volts |
| V _{IL} | Input LOW Level (Guaranteed Logic LOW Voltage) | | All Inputs | | 0.8 | Volts |
| V _I | Input Clamp Voltage | V _{CC} = 4.75 V, I _{IN} = -18 mA | All Inputs | | -1.5 | Volts |
| I _{IL} | Input LOW Current | V _{CC} = 5.25 V, V _{IN} = 0.5 V | PY0-3, Y0-31 | | -0.55 | mA |
| | | | I4-6 | | -1.50 | |
| | | | I7-8 | | -1.00 | |
| | | | SLAVE | | -3.00 | |
| | | | OE-Y | | -2.50 | |
| | | | CLK | | -2.00 | |
| | | | C, Z, V, N, L; PERR | | -0.55 | |
| | | | Other | | -0.50 | |
| I _{IH} | Input HIGH Current | V _{CC} = 5.25 V, V _{IN} = 2.4 V | PY0-3, Y0-31 | | 100 | μA |
| | | | I4-6 | | 150 | |
| | | | I7-8 | | 100 | |
| | | | SLAVE | | 300 | |
| | | | OE-Y | | 250 | |
| | | | CLK | | 200 | |
| | | | C, Z, V, N, L; PERR | | 100 | |
| | | | Other | | 50 | |
| I _I | Input HIGH Current | V _{CC} = 5.25 V, V _{IN} = 5.5 V | All Inputs | | 1.0 | mA |
| I _{OZH} | Off State Output Current | V _{CC} = 5.25 V, V _O = 2.4 V | All Outputs Except MSERR | | 100 | μA |
| I _{OZL} | | V _{CC} = 5.25 V, V _O = 0.5 V | | | -550 | |
| I _{OS} | Output Short-Circuit Current (Note 2) | V _{CC} = 5.75 V, V _O = 0.5 V | | -15 | -50 | mA |
| I _{CC} | Power Supply Current (Note 3) | V _{CC} = 5.25 V | T _C = 0 to 85°C | | 1800 | mA |
| | | | T _C = 85°C | | 1690 | mA |

- Notes: 1. For conditions shown as Min. or Max., use the appropriate value specified under Operating Ranges for the applicable device type.
 2. Not more than one output should be shorted at a time. Duration of the short circuit test should not exceed one second.
 3. Measured with all inputs HIGH and outputs disabled.

SWITCHING CHARACTERISTICS over operating range

A. COMBINATIONAL PROPAGATION DELAYS

| No. | From | To | Am29332 | Am29332A* | Unit |
|-----|---|-----------------------------------|------------|------------|------|
| | | | Max. Delay | Max. Delay | |
| 1 | PA ₀ - PA ₃ , PB ₀ - PB ₃ | PERR | 19 | 16 | ns |
| 2 | DA ₀ - DA ₃₁ , DB ₀ - DB ₃₁ | PERR | 28 | 24 | ns |
| 3 | DA ₀ - DA ₃₁ , DB ₀ - DB ₃₁ | PY ₀ - PY ₃ | 42 | 36 | ns |
| 4 | DA ₀ - DA ₃₁ , DB ₀ - DB ₃₁ | Y ₀ - Y ₃₁ | 35 | 30 | ns |
| 5 | DA ₀ - DA ₃₁ , DB ₀ - DB ₃₁ | C, Z, V, N, L | 43 | 37 | ns |
| 6 | DA ₀ - DA ₃₁ , DB ₀ - DB ₃₁ | MSERR | 49 | 42 | ns |
| 7 | I ₀ - I ₈ | PY ₀ - PY ₃ | 53 | 45 | ns |
| 8 | I ₀ - I ₈ | Y ₀ - Y ₃₁ | 47 | 40 | ns |
| 9 | I ₀ - I ₈ | C, Z, V, N, L | 48 | 41 | ns |
| 10 | I ₀ - I ₈ | MSERR | 55 | 47 | ns |
| 11 | W ₀ - W ₄ | PY ₀ - PY ₃ | 40 | 34 | ns |
| 12 | W ₀ - W ₄ | Y ₀ - Y ₃₁ | 34 | 29 | ns |
| 13 | W ₀ - W ₄ | C, Z, V, N, L | 35 | 30 | ns |
| 14 | W ₀ - W ₄ | MSERR | 41 | 35 | ns |
| 15 | P ₀ - P ₅ | PY ₀ - PY ₃ | 48 | 41 | ns |
| 16 | P ₀ - P ₅ | Y ₀ - Y ₃₁ | 42 | 36 | ns |
| 17 | P ₀ - P ₅ | C, Z, V, N, L | 43 | 37 | ns |
| 18 | P ₀ - P ₅ | MSERR | 45 | 39 | ns |
| 19 | CP | PY ₀ - PY ₃ | 47 | 40 | ns |
| 20 | CP | Y ₀ - Y ₃₁ | 41 | 35 | ns |
| 21 | CP | C, Z, V, N, L | 42 | 36 | ns |
| 22 | CP | STATUS REG. | 20 | 17 | ns |
| 23 | RS | C, Z, V, N, L | 16 | 14 | ns |
| 24 | MC _{in} | Y ₀ - Y ₃₁ | 31 | 27 | ns |
| 25 | MC _{in} | C, Z, V, N, L | 34 | 29 | ns |
| 26 | MC _{in} | MSERR | 37 | 32 | ns |
| 27 | MLINK | Y ₀ - Y ₃₁ | 33 | 28 | ns |
| 28 | MLINK | C, Z, V, N, L | 37 | 32 | ns |
| 29 | MLINK | MSERR | 38 | 33 | ns |
| 30 | M/ \bar{m} | Y ₀ - Y ₃₁ | 33 | 28 | ns |
| 31 | M/ \bar{m} | C, Z, V, N, L | 37 | 32 | ns |
| 32 | M/ \bar{m} | MSERR | 38 | 33 | ns |
| 33 | BOROW | Y ₀ - Y ₃₁ | 33 | 28 | ns |
| 34 | BOROW | C, Z, V, N, L | 37 | 32 | ns |
| 35 | BOROW | MSERR | 38 | 33 | ns |
| 36 | HOLD | C, Z, V, N, L | 22 | 19 | ns |
| 37 | HOLD | MSERR | 29 | 25 | ns |
| 38 | PY ₀ - PY ₃ | MSERR | 20 | 17 | ns |
| 39 | Y ₀ - Y ₃₁ | MSERR | 19 | 16 | ns |
| 40 | C, Z, V, N, L | MSERR | 21 | 18 | ns |
| 41 | PERR | MSERR | 20 | 17 | ns |

*ADVANCED INFORMATION

SWITCHING CHARACTERISTICS (Cont'd.)

B. SETUP AND HOLD TIMES

| No. | Parameter (Note 2) | For | With Respect To | Am29332 | Am29332A* | Unit |
|-----|--------------------|---|-----------------|------------|------------|------|
| | | | | Max. Value | Max. Value | |
| 42 | Input Data Setup | DA ₀ - DA ₃₁ , DB ₀ - DB ₃₁ | CP ↑ | 31 | 31 | ns |
| 43 | Input Data Hold | DA ₀ - DA ₃₁ , DB ₀ - DB ₃₁ | CP ↑ | 1 | 1 | ns |
| 44 | Byte Width Setup | I ₇ - I ₈ | CP ↑ | 30 | 30 | ns |
| 45 | Byte Width Hold | I ₇ - I ₈ | CP ↑ | 1 | 1 | ns |
| 46 | Instruction Setup | I ₀ - I ₆ | CP ↑ | 37 | 37 | ns |
| 47 | Instruction Hold | I ₀ - I ₆ | CP ↑ | 2 | 2 | ns |
| 48 | Width Setup | W ₀ - W ₄ | CP ↑ | 28 | 28 | ns |
| 49 | Width Hold | W ₀ - W ₄ | CP ↑ | 0 | 0 | ns |
| 50 | Position Setup | P ₀ - P ₅ | CP ↑ | 28 | 28 | ns |
| 51 | Position Hold | P ₀ - P ₅ | CP ↑ | 0 | 0 | ns |
| 52 | Borrow Setup | BOROW | CP ↑ | 22 | 22 | ns |
| 53 | Borrow Hold | BOROW | CP ↑ | 1 | 1 | ns |
| 54 | Macro Carry Setup | MCin | CP ↑ | 21 | 21 | ns |
| 55 | Macro Carry Hold | MCin | CP ↑ | 0 | 0 | ns |
| 56 | Macro Link Setup | MLINK | CP ↑ | 22 | 22 | ns |
| 57 | Macro Link Hold | MLINK | CP ↑ | 1 | 1 | ns |
| 58 | Macro/Micro Setup | M/m | CP ↑ | 22 | 22 | ns |
| 59 | Macro/Micro Hold | M/m | CP ↑ | 1 | 1 | ns |
| 60 | Hold Mode Setup | HOLD | CP ↑ | 11 | 11 | ns |
| 61 | Hold Mode Hold | HOLD | CP ↑ | 1 | 1 | ns |

C. MINIMUM CLOCK REQUIREMENTS

| No. | Description | Am29332 | Am29332A* | Unit |
|-----|-------------------------|------------|------------|------|
| | | Max. Value | Max. Value | |
| 62 | Minimum Clock LOW Time | 20 | 20 | ns |
| 63 | Minimum Clock HIGH Time | 20 | 20 | ns |

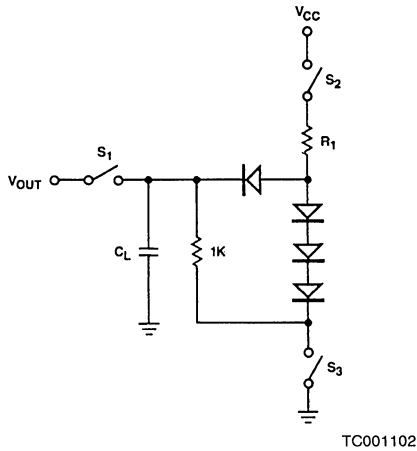
D. ENABLE AND DISABLE TIMES

| No. | From | To | Description | Am29332 | Am29332A* | Unit |
|-----|--------|---|-------------------------|------------|------------|------|
| | | | | Max. Delay | Max. Delay | |
| 64 | OE - Y | Y ₀ - Y ₃₁ , PY ₀ - PY ₃ | Output Enable Time | 25 | 25 | ns |
| 65 | OE - Y | Y ₀ - Y ₃₁ , PY ₀ - PY ₃ | Output Disable Time | 25 | 25 | ns |
| 66 | SLAVE | C, Z, V, N, L PERR | Slave Mode Enable Time | 25 | 25 | ns |
| 67 | SLAVE | Y ₀ - Y ₃₁ , PY ₀ - PY ₃ C, Z, V, N, L PERR | Slave Mode Disable Time | 25 | 25 | ns |

Notes: 1. It is the responsibility of the user to maintain a case temperature of 85°C or less. AMD recommends an air velocity of at least 200 linear feet per minute over the heatsink.

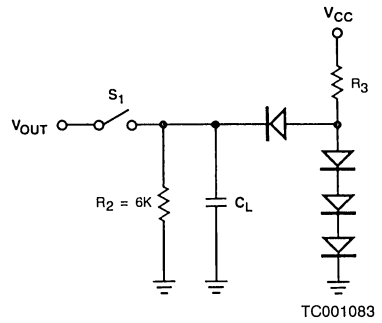
2. See timing diagram for desired mode of operation to determine clock edge to which these setup and hold times apply.

SWITCHING TEST CIRCUITS



$$R_1 = \frac{5.0 - V_{BE} - V_{OL}}{I_{OL} + \frac{V_{OL}}{1K}}$$

A. Three-State Outputs



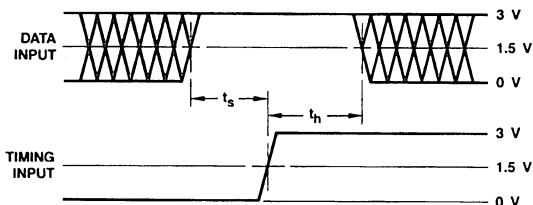
$$R_2 = \frac{2.4 V}{I_{OH}}$$

$$R_1 = \frac{5.0 - V_{BE} - V_{OL}}{I_{OL} + \frac{V_{OL}}{R_2}}$$

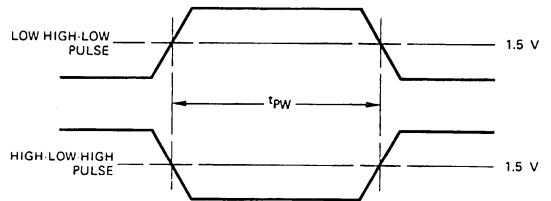
B. Normal Outputs

- 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 tp_{ZH} test.
 4. S_1 and S_2 are closed while S_3 is open for tp_{ZL} test.
 5. $C_L = 5.0$ pF for output disable tests.

SWITCHING TEST WAVEFORMS



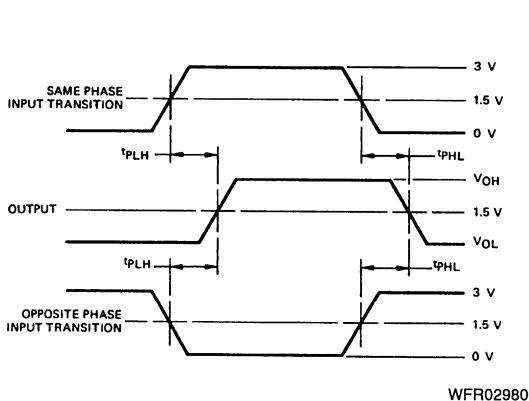
Setup, Hold, and Release Times



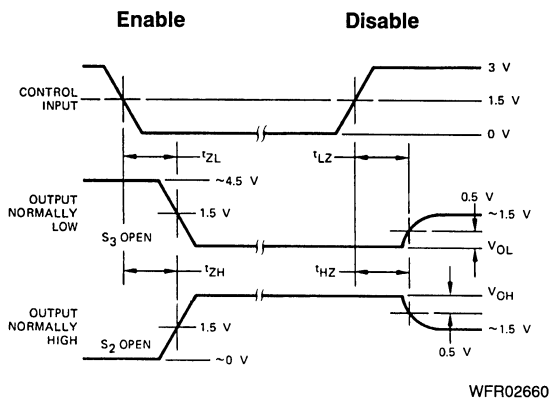
Pulse Width

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

SWITCHING TEST WAVEFORMS (Cont'd.)



Propagation Delay



Enable and Disable Times

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.

Test Philosophy and Methods

The following points give the general philosophy that we apply to tests that must be properly engineered if they are to be implemented in an automatic environment. The specifics of what philosophies applied to which test are shown.

1. Ensure the part is adequately decoupled at the test head. Large changes in supply current when the device switches may cause function failures due to V_{CC} changes.
2. Do not leave inputs floating during any tests, as they may oscillate at high frequency.
3. Do not attempt to perform threshold tests at high speed. Following an input 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 input levels for AC tests. Many inputs may be changed at once, so there will be significant noise at the device pins that may not actually reach V_{IL} or V_{IH} until the noise has settled. AMD recommends using $V_{IL} \leq 0$ V and $V_{IH} \geq 3$ 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 that call for a smaller capacitive load than the associated stray capacitance. Typical examples of this are the so-called "float delays" which measure the propagation delays into and out of 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 set up 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 set up and the knowledge that certain DC measurements (I_{OH} , I_{OL} , for example) have already been taken and are within specification. In some cases, special DC tests are performed in order to facilitate this correlation.

7. Threshold Testing

The noise associated with automatic testing, the long, inductive cables, and the high gain of bipolar devices when in the vicinity of the actual device threshold, frequently give rise to oscillations when testing high-speed 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 data from precise bench measurements in conjunction with the knowledge that certain DC parameters have already been measured and are within specification.

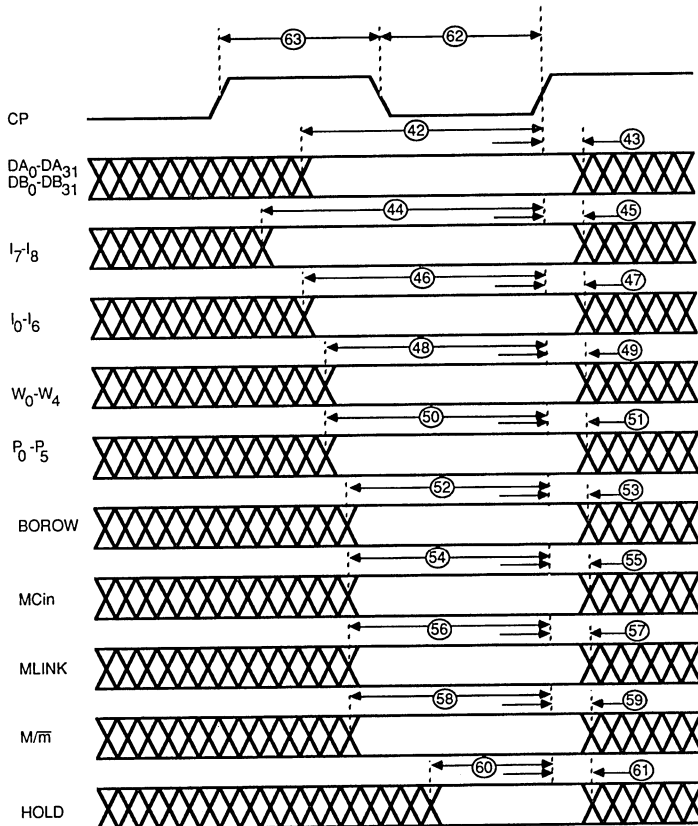
In some cases, certain AC tests are redundant since they can be shown to be predicted by other tests that have already been performed. In these cases, the redundant tests are not performed.

SWITCHING WAVEFORMS

KEY TO SWITCHING WAVEFORMS

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

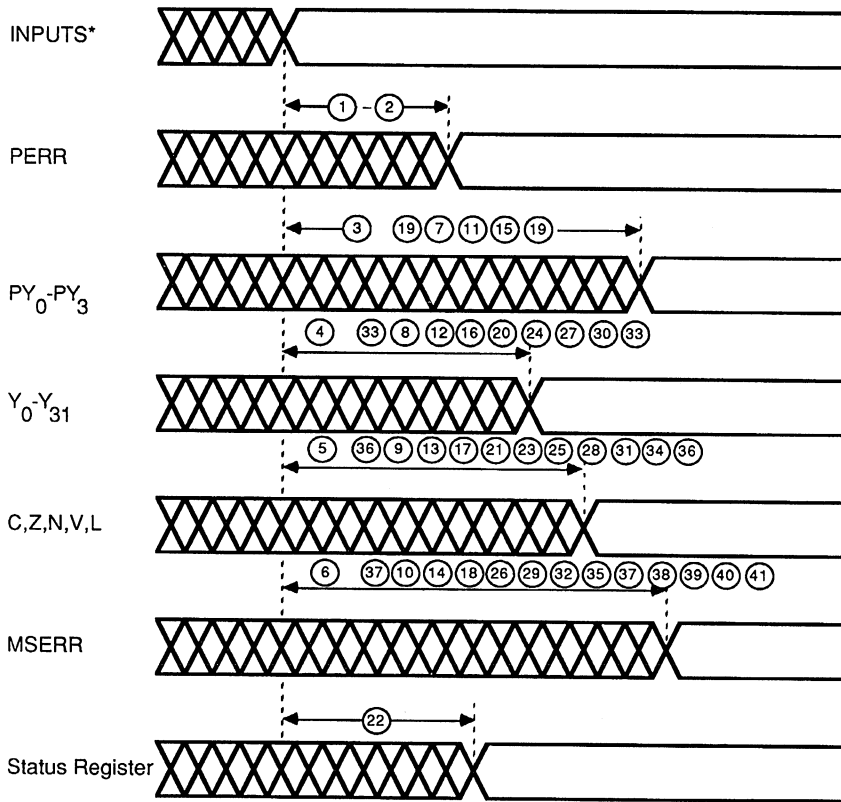
KS000010



WF023680

Setup and Hold Timing

SWITCHING WAVEFORMS (Cont'd.)

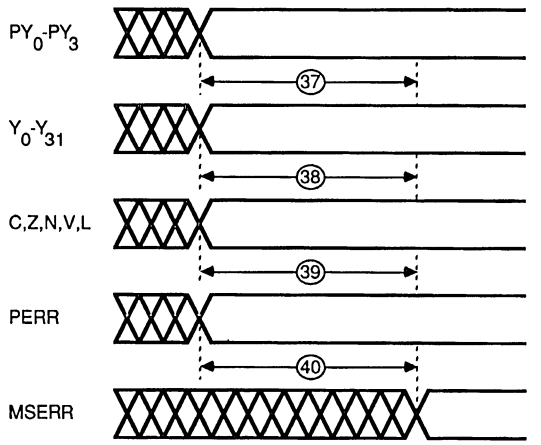


WF023691

Propagation Delays (SLAVE = LOW)

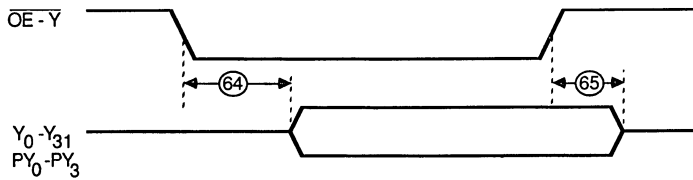
Inputs: PA₀-PA₃, PB₀-PB₃, DA₀-DA₃₁, DB₀-DB₃₁, I₀-I₈, W₀-W₄, P₀-P₅, CP, RS, MCin, MLINK, M/ \bar{m} , BOROW, HOLD

SWITCHING WAVEFORMS (Cont'd.)



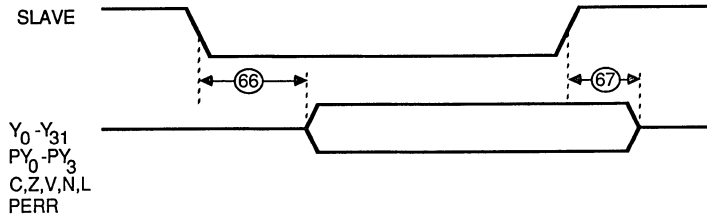
WF023700

Propagation Delay (SLAVE = HIGH)



WF023710

Enable/Disable I (SLAVE = HIGH)

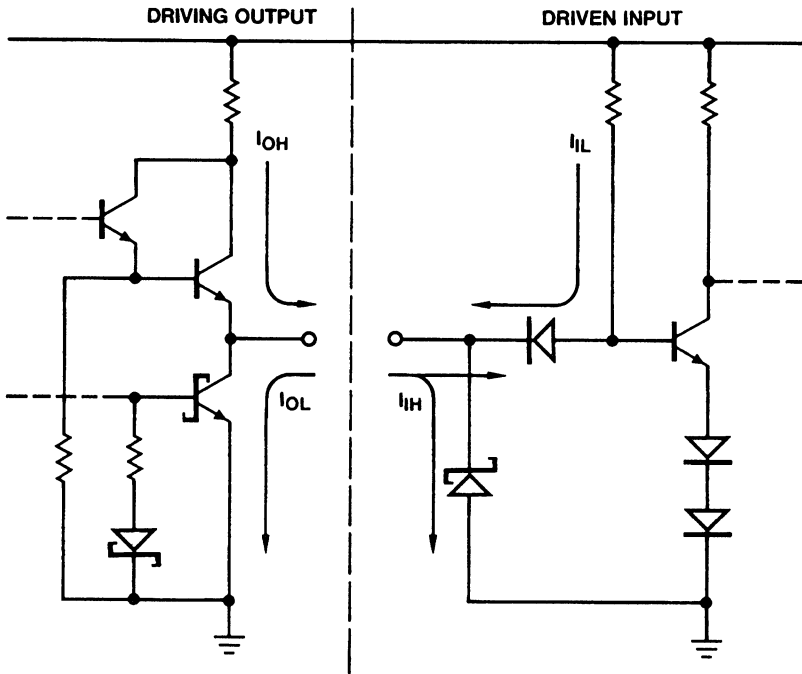


WF023720

Enable/Disable II ($\overline{OE-Y}$ = LOW)

INPUT/OUTPUT CIRCUIT DIAGRAM

(All Devices)

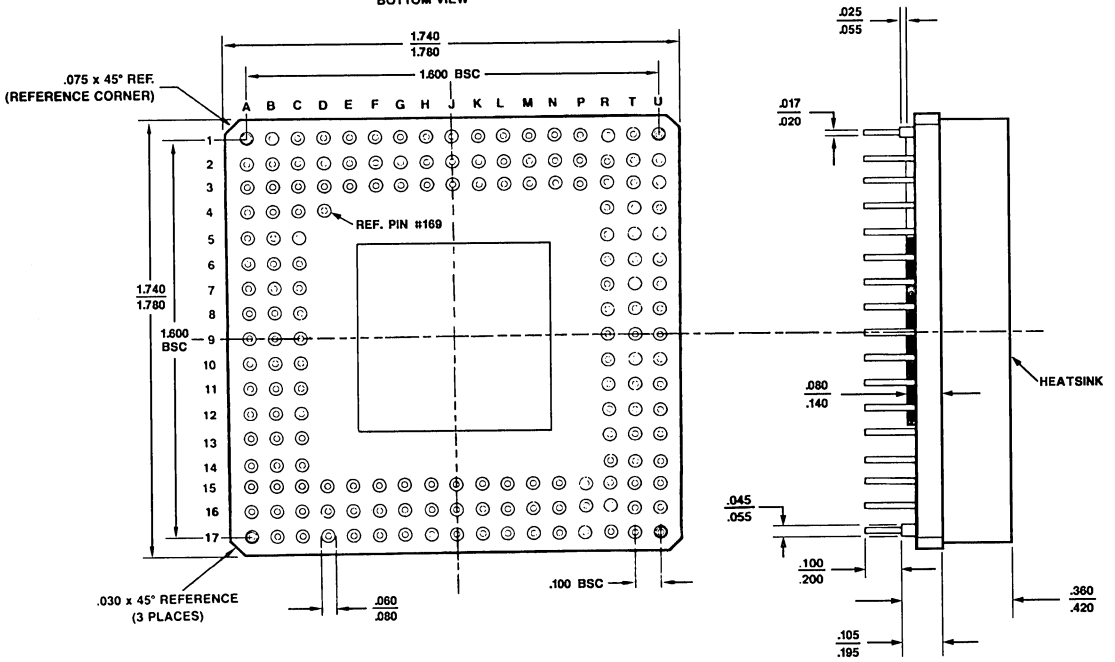


ICR00480

PHYSICAL DIMENSIONS*

CG 169

BOTTOM VIEW



* For reference only.

ADVANCED MICRO DEVICES DOMESTIC SALES OFFICES

| | | | |
|----------------------|----------------|-----------------------|----------------|
| ALABAMA | (205) 882-9122 | MASSACHUSETTS | (617) 273-3970 |
| ARIZONA, | | MINNESOTA | (612) 938-0001 |
| Tempe | (602) 242-4400 | MISSOURI | (314) 275-4415 |
| CALIFORNIA, | | NEW JERSEY | (201) 299-0002 |
| Culver City | (213) 645-1524 | NEW YORK, | |
| Newport Beach | (714) 752-6262 | Liverpool | (315) 457-5400 |
| San Diego | (619) 560-7030 | Poughkeepsie | (914) 471-8180 |
| Santa Clara | (408) 727-3270 | Woodbury | (516) 364-8020 |
| Woodland Hills | (818) 992-4155 | NORTH CAROLINA, | (919) 847-8471 |
| COLORADO | (303) 741-2900 | OREGON | (503) 245-0080 |
| CONNECTICUT | (203) 264-7800 | OHIO | (614) 891-6455 |
| FLORIDA, | | PENNSYLVANIA, | |
| Clearwater | (813) 530-9971 | Allentown | (215) 398-8006 |
| Ft. Lauderdale | (305) 484-8600 | Willow Grove | (215) 657-3101 |
| Melbourne | (305) 729-0496 | TEXAS, | |
| Orlando | (305) 859-0831 | Austin | (512) 346-7830 |
| GEORGIA | (404) 449-7920 | Dallas | (214) 934-9099 |
| ILLINOIS | (312) 773-4422 | Houston | (713) 785-9001 |
| INDIANA | (317) 244-7207 | WASHINGTON | (206) 455-3600 |
| KANSAS | (913) 451-3115 | WISCONSIN | (414) 782-7748 |
| MARYLAND | (301) 796-9310 | | |

INTERNATIONAL SALES OFFICES

| | | | | |
|---------------------|------------|-------------------------|------------|-------------------|
| BELGIUM, | | ITALY, Milano | TEL: | (02) 3390541 |
| Bruxelles | TEL: | | FAX: | (02) 3498000 |
| | | | TLX: | 315286 |
| CANADA, Ontario, | | JAPAN, | | |
| Kanata | TEL: | Tokyo | TEL: | (03) 345-8241 |
| Willowdale | TEL: | | FAX: | 3425196 |
| | | | TLX: | J24064 AMDTKOJ |
| | | Osaka | TEL: | 06-243-3250 |
| FRANCE, | | | FAX: | 06-243-3253 |
| Paris | TEL: | KOREA, Seoul | TEL: | 82-733-1021/7 |
| | | | FAX: | 82-733-1028 |
| | | | TLX: | K22652 |
| GERMANY, | | LATIN AMERICA, | | |
| Hannover area | TEL: | Ft. Lauderdale, | TEL: | (305) 484-8600 |
| | | | FAX: | (305) 485-9736 |
| | | | TLX: | 5109554261 AMDFTL |
| München | TEL: | SWEDEN, Stockholm | TEL: | (08) 733 03 50 |
| | | | FAX: | (08) 733 22 85 |
| | | | TLX: | 11602 |
| Stuttgart | TEL: | UNITED KINGDOM, | | |
| | | Manchester area | TEL: | (0925) 828008 |
| | | | FAX: | (0925) 827693 |
| | | | TLX: | 628524 |
| HONG KONG, | | London area | TEL: | (04862) 22121 |
| Kowloon | TEL: | | FAX: | (04862) 22179 |
| | | | TLX: | 859103 |

NORTH AMERICAN REPRESENTATIVES

| | | | |
|-----------------------------------|----------------------|--------------------------------|----------------|
| CALIFORNIA | | NEBRASKA | |
| I ² INC | OEM (408) 988-3400 | LORENZ SALES | (402) 475-4660 |
| | DISTI (408) 496-6868 | NEW MEXICO | |
| IDAHO | | THORSON DESERT STATES | (505) 293-8555 |
| INTERMOUNTAIN TECH MKGT | (208) 888-6071 | NEW YORK | |
| INDIANA | | NYCOM, INC | (315) 437-8343 |
| ELECTRONIC MARKETING CONSULTANTS, | | OHIO | |
| INC | (317) 253-1668 | Dayton | |
| IOWA | | DOLFUSS ROOT & CO | (513) 433-6776 |
| LORENZ SALES | (319) 377-4666 | Strongsville | |
| KANSAS | | DOLFUSS ROOT & CO | (216) 238-0300 |
| LORENZ SALES | (913) 384-6556 | PENNSYLVANIA | |
| MICHIGAN | | DOLFUSS ROOT & CO | (412) 221-4420 |
| SAI MARKETING CORP | (313) 750-1922 | UTAH | |
| MISSOURI | | R ² MARKETING | (801) 595-0631 |
| LORENZ SALES | (314) 997-4558 | | |

Advanced Micro Devices reserves the right to make changes in its product without notice in order to improve design or performance characteristics. The performance characteristics listed in this document are guaranteed by specific tests, correlated testing, guard banding, design and other practices common to the industry. For specific testing details, contact your local AMD sales representative. The company assumes no responsibility for the use of any circuits described herein.

