# Am29PL141

Fuse Programmable Controller (FPC)

## Distinctive Characteristics

- Implements complex fuse programmable state machines
- 7 conditional inputs, 16 outputs
- 64 words of 32-bit-wide microprogram memory
- Serial Shadow Register (SSR™) diagnostics on chip (programmable option)
- 29 high-level microinstructions
  - Conditional branching
  - Conditional looping
  - Conditional subroutine call
  - Multiway branch
- 20 MHz clock rate, 28-pin DIP

## General Description

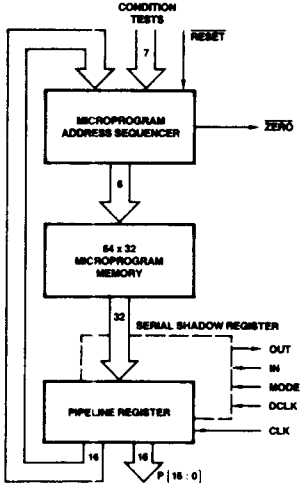
The Am29PL141 is a single-chip Fuse Programmable Controller (FPC) which allows implementation of complex state machines and controllers by programming the appropriate sequence of microinstructions. A repertoire of jumps, loops, and subroutine calls, which can be conditionally executed based on the test inputs, provides the designer with powerful control flow primitives.

The Am29PL141 FPC also allows distribution of intelligent control throughout the system. It off-loads the central controller by distributing FPCs as the control for various self-contained functional units, such as register file/ALU, I/O, interrupt, diagnostic, and bus control units.

A microprogram address sequencer is the heart of the FPC. It provides the microprogram address to an internal 64-word by 32-bit PROM. The fuse programming algorithm is almost identical to that used for AMD's Programmable Array Logic family.

As an option, the Am29PL141 may be programmed to have on chip SSR diagnostics capability. Microinstructions can be serially shifted in, executed, and the results shifted out to facilitate system diagnostics.

## Block Diagram




BDR02340

## Related Products

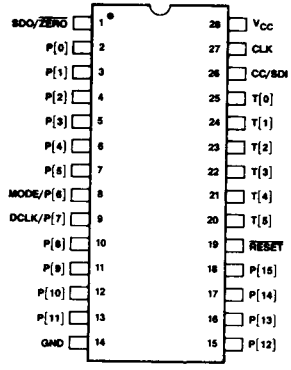
| Part No. | Description |
|---|---|
| Am2914 | Vectored Priority Interrupt Controller |
| Am29100 | Controller Family Products |

SSR is a trademark of Advanced Micro Devices, Inc.

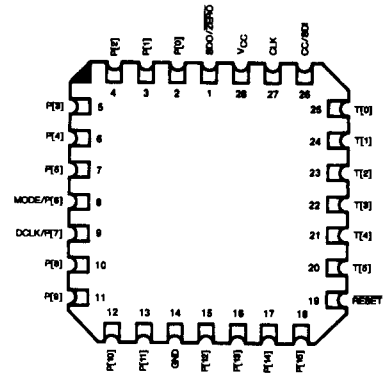
04179F/0
JANUARY 1988

## Connection Diagrams

### Top View



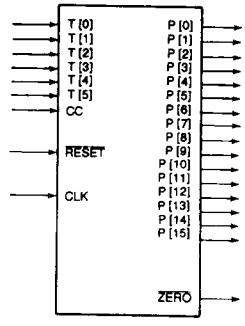

CDR04480




CD009110

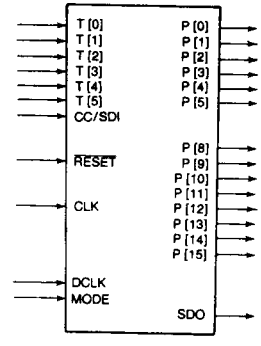
Note: Pin 1 is marked for orientation.

## Logic Symbols




LS002131

**Normal Configuration**



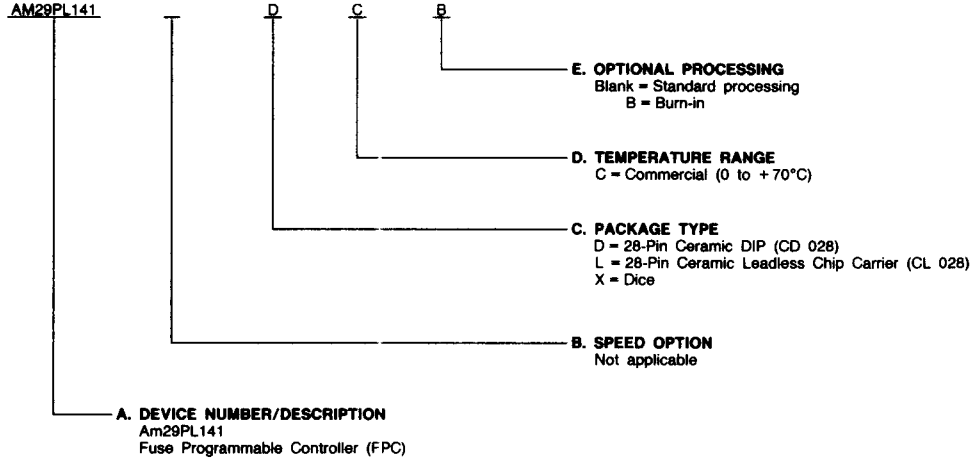

LS002140

**SSR™ Diagnostics Configuration**

## 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 | |
|---|---|
| AM29PL141 | DC, DCB, LC, XC |

**Valid Combinations**

Valid Combinations list cnfigurations 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 valid combinations, and to obtain additional data on AMD's standard military grade products.

5

# Ordering Information

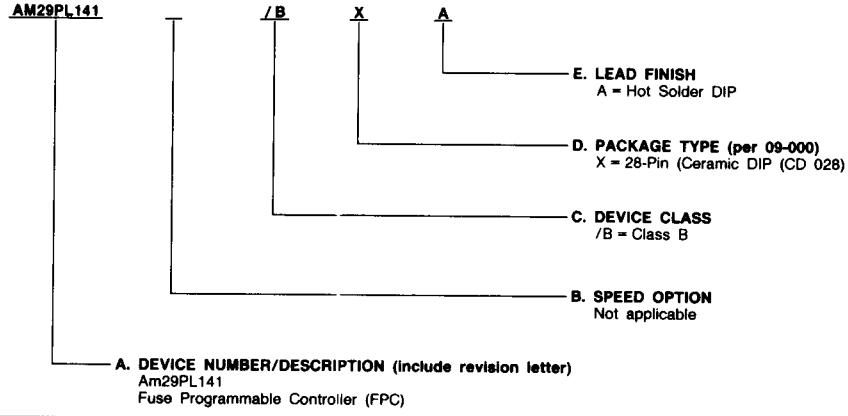
## APL and CPL Products

AMD products for Aerospace and Defense applications are available in several packages and operating ranges. APL (Approved Products List) products are fully compliant with MIL-STD-883C requirements. CPL (Controlled Products List) products are processed in accordance with MIL-STD-883C, but are inherently non-compliant because of package, solderability, or surface treatment exceptions to those specifications. The order number (Valid Combination) is formed by a combination of:

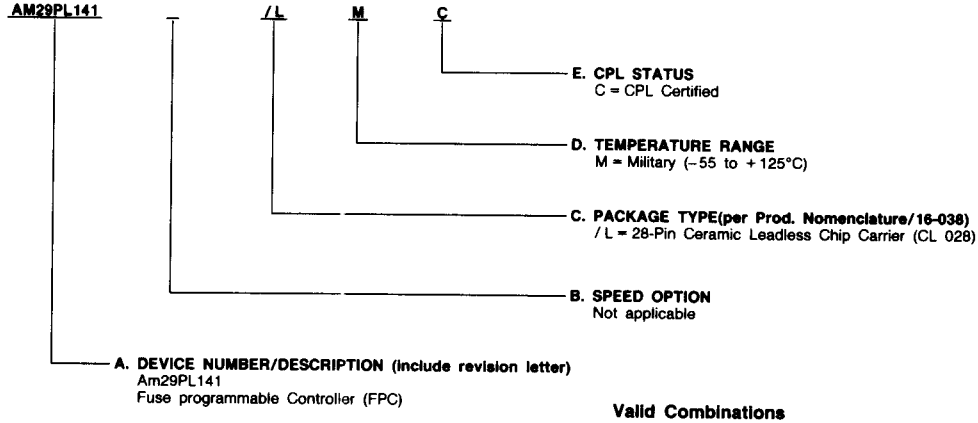
**APL Products:**
**A. Device Number**
**B. Speed Option** (if applicable)
**C. Device Class**
**D. Package Type**
**E. Lead Finish**

**CPL Products:**
**A. Device Number**
**B. Speed Option** (if applicable)
**C. Package Type**
**D. Temperature Range**
**E. CPL Status**

### APL Products




### CPL Products




| Valid Combinations | | |
|---|---|---|
| APL | Am29PL141 | /BXA |
| CPL | Am29PL141 | /LMC |

### Valid Combinations

Valid Combinations list configurations planned to be supported in volume for this device. Consult the local AMD sales office to confirm availability of specific valid combinations or to check for newly released valid combinations.

### Group A Tests

Group A tests consists of Subgroups:
1, 2, 3, 7, 8, 9, 10, 11

## Pin Description

**CC[SDI] Condition Code ((TEST) Input)**
When the TEST (P[24:22]) field of the executing microinstruction is set to 6 (binary 110), CC is selected to be the conditional input. (Note: In SSR diagnostic configuration, CC is also the Serial Data Input SDI.)

**CLK Clock (Input)**
The rising edge clocks the microprogram counter, count register, subroutine register, pipeline register, and EQ flag.

**P[15:8] (Outputs)**
Upper eight, general-purpose microprogram control outputs. They are enabled by the OE signal from the microprogram pipeline register. When OE is HIGH, P[15:8] are enabled, and when LOW, P[15:8] are three-stated.

**P[7:0] [DLCK, MODE] (Outputs)**
Lower
Lower eight, general-purpose microprogram control outputs. They are permanently enabled. (Note: in the SSR diagnostic configuration, P[7] becomes the diagnostic clock input DCLK and P[6] becomes the diagnostic control input MODE.)

**$\overline{\text{RESET}}$**
Synchronous reset input. When it is low, the output of the PC MUX is forced to the uppermost microprogram address (63). On the next rising clock edge, this address (63) is loaded into the microprogram counter, the microinstruction at location 63 is loaded into the pipeline register and the EQ flag is cleared. The CREG and SREG values are indeterminate on reset.

**T[5:0]**
Test inputs. In conditional microinstructions, the inputs can be used as individual condition codes selected by the TEST field in the pipeline register. The T[5:0] inputs can also be used as a branch address when performing a microprogram branch, or as a count value.

**$\overline{\text{ZERO}}$ [SDO]**
Zero output. A Low state indicates that the CREG value is zero. (Note: In the SSR diagnostic configuration, $\overline{\text{ZERO}}$ becomes the Serial Data output SDO. This change is only on the output pin; internally, the zero detect functions is unchanged.)

5

## Functional Description

Figure 1, the block diagram of the Am29PL141 FPC, shows logic blocks and interconnecting buses. These allow parallel performance of different operations in a single microinstruction. The FPC consists of four main logic blocks: the microprogram memory, microaddress control logic, condition code selection logic, and microinstruction decode. A fifth optional block is the Serial Shadow Register (SSR).

The microprogram memory contains the user-defined instruction flow and output sequence. The microaddress control logic addresses the microprogram memory. This control logic supports high-level microinstruction functions including conditional branches, subroutine calls and returns, loops, and multiway branches. The condition code selection logic selects the condition code input to be tested when a conditional microinstruction is executed. The polarity of the selected condition code input is controlled by the POL bit in the microword. The microinstruction decode generates the control signals necessary to perform the microinstruction specified by the microinstruction part (P[31 : 16]) of the microword. The SSR enables in-system testing that allows isolation of problems down to the IC level.

## Microprogram Memory

The FPC microprogram memory is a 64-word by 32-bit PROM with a 32-bit pipeline register at its output. The upper 16 bits (P[31 : 16]) of the pipeline register stay internal to the FPC and form the microinstruction to control address sequencing. The format for microinstructions is: a one-bit synchronous Output Enable OE, a five-bit OPCODE, a one-bit test polarity select POL, a three-bit TEST condition select field, and a six-bit immediate DATA field. The DATA field is used to provide branch addresses, test input masks, and counter values.

The lower 16 bits (P[15 : 0]) of the pipeline register are brought out as user-defined, general purpose control outputs. The upper eight control outputs (P[15 : 8]) are three-stated when OE is programmed as a LOW. The lower eight control bits (P[7 : 0]) are always enabled.

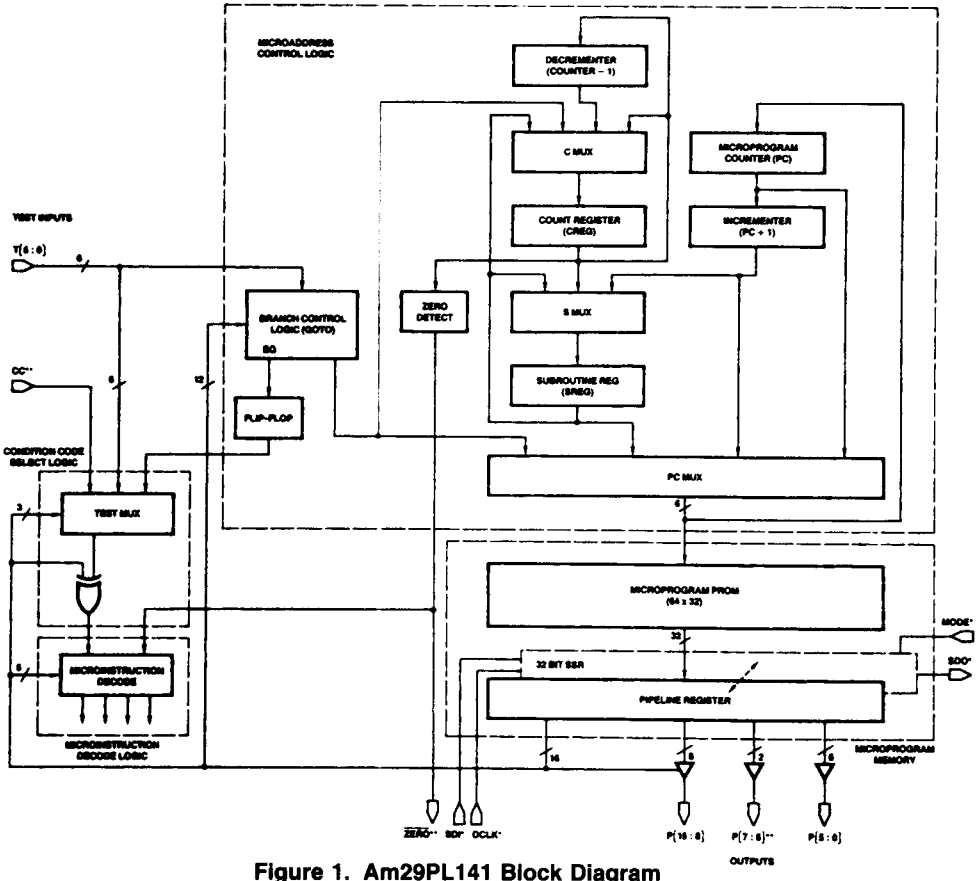



**Figure 1. Am29PL141 Block Diagram**

BDR02330

*Note: These pins available only in SSR mode.
**Note: These pins available only in normal mode.

## Microaddress Control Logic

The microaddress control logic consists of five smaller logic blocks. These are:

PC MUX – The microprogram counter multiplexer

P CNTR – Microprogram counter (PC) and incrementer (PC + 1)

SUBREG – Subroutine register (SREG) with subroutine mux (S MUX)

CNTR – Count register (CREG) with counter mux (C MUX), decrementer (COUNTER–1) and zero detect

GOTO – Specialized branch control logic

The PC MUX is a six-bit, four-to-one multiplexer. It selects either the PC, PC+1, SREG, or GOTO output as the next microaddress input to the microprogram memory and to the PC. The PC thus always contains the address of the microinstruction in the pipeline register. During a Reset, the PC MUX output is forced to all ones, selecting location 63 of the microprogram memory.

The P CNTR block consists of a six-bit register (PC) driving a six-bit combinatorial incrementer (PC+1). Either the present or the incremented values of PC can address the microprogram PROM. The incremented value of PC can be saved as a subroutine return address. The present PC value can address the microprogram PROM when waiting for a condition to become valid. PC+1 addresses the microprogram PROM for sequential microprogram flow, for unconditional microinstructions, and as a default for conditional microinstructions

The SUBREG block consists of a six-bit, three-to-one multiplexer (S MUX) driving a six-bit register (SREG). The three possible SREG inputs are PC+1, CREG, and SREG. SREG normally operates as a one-deep stack to save subroutine return addresses. PC+1 is the input source when performing subroutine calls and PC MUX is the output destination when performing return from subroutine.

The CNTR block consists of a six-bit, four-to-one multiplexer (C MUX); driving a six-bit register (CREG); a six-bit, combinatorial decrementer (COUNTER-1); and a zero detection circuit. The CNTR logic block is typically used for timing functions and iterative loop counting.

The SUBREG and CNTR can be considered as one logic block because of their unique interaction. To explain this interaction, notice that both have an additional input source and output destination not used in typical operation—each other. This allows the CREG to be an additional stack location when not used for counting, and the SREG to be a nested count location when not used as a stack location. Thus, the SREG and CREG can operate in three different modes:

1. As a separate one-deep stack and counter.
2. As a two-deep stack.
3. As a two-deep nested counter.

The GOTO logic block serves three functions:

1. It provides a six-bit count value from the DATA Field in the pipeline register (P[21 : 16]) or from the TEST inputs (T[5 : 0]) masked by the DATA Field (P[21 : 16]). (This is represented by T*M.)
2. It provides a branch address from the DATA Field in the pipeline register (P[21 : 16]) or from the TEST inputs (T[5 : 0]) masked by the DATA Field (P[21 : 16]). (This is represented by T*M.)
3. It compares the TEST inputs (T[5 : 0]) masked by the DATA Field (P[21 : 16]), called T*M, to the CONSTANT Field from the pipeline register (P[27 : 22]). If a match occurs, the EQ Flip-flop is set. EQ remains unchanged if there is no match. Constant field bits that correspond to marked test bits must be ZERO.

The EQ flag can be tested by the condition code selection logic. Multiple tests of any group of T inputs in a manner analogous to Sum-of-Products can be performed since a no match comparison does not reset the EQ flag. Any conditional branch on EQ will reset the EQ flag. Conditional returns on EQ will not change the EQ flag. $\overline{\text{RESET}}$ input LOW will reset the EQ flag.

NOTE: A zero in the DATA Field blocks the corresponding bit in the TEST Field; a one activates the corresponding bit.

The constant field bits that correspond to masked test field bits must be zero. A zero is substituted for masked test field bits. The 'POL' bit is a "don't care" when using test inputs to load registers.

## Condition Code Selection Logic

The condition code selection logic consists of an eight-to-one multiplexer. The eight test conditon inputs are the device inputs (CC and T[5 : 0]) and the EQ flag. The TEST field P[24 : 22] selects one of the eight conditions to test.

The polarity bit POL in the microinstructions allows the user to test for either a true or false conditon. Refer to Table 2 for details.

## Microinstruction Decode

The microinstruction decoder is a PLA that generates the control for 29 different microinstructions. The decoder's inputs include the OPCODE Field (P[30 : 26]), the zero detection output from the CNTR, and the selected test condition code from the conditional code selection logic.

## Am29PL141 SSR Diagnostics Option

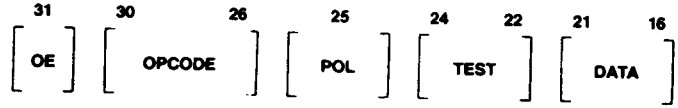
As a programmable option, the Am29PL141 FPC may be configured to contain Serial Shadow Register (SSR) diagnostics capability. SSR diagnostics is a simple, straightforward method of in-system testing that allows isolation of problems down to the IC level.

The SSR diagnostics configuration activates a 32-bit-wide, D-type register, called a "shadow" register, on the pipeline register inputs. The shadow register can be serially loaded from the SDI pin, parallel loaded from the pipeline register, or held. The pipeline register can be loaded from the microprogram memory in normal operation or from the shadow register during diagnostics. A redefinition of four device pins is required to control the different diagnostics functions. CC also functions as the Serial Data Input (SDI), $\overline{\text{ZERO}}$ becomes the Serial Data Output (SDO), P[7] becomes the diagnostic clock (DCLK), and P[6] becomes the diagnostic mode control (MODE). The various diagnostic and normal modes are shown in Table 1.

Serially loading a test microinstruction into the shadow register and parallel loading the shadow register contents into the pipeline register forces execution of the test microinstruction. The result of the test microinstruction can then be clocked into the pipeline register, as in normal operation mode, parallel loaded into the shadow register, and serially shifted out for system diagnostics.

The general microinstruction format is shown below:

**Am29PL141 General Microinstruction Format**




DFR00730

WHERE:

OE = Synchronous Output Enable for P[15:8].

OPCODE = A five-bit opcode field for selecting one of the twenty-eight single data field microinstructions.

POL = A one-bit test condition polarity select.
0 = Test for true (HIGH) condition.
1 = Test for false (LOW) condition.

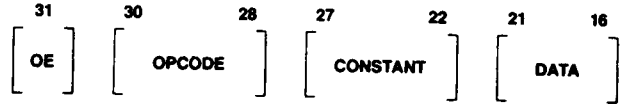
TEST = A three-bit test condition select.

| TEST[2:0] | UNDER TEST |
|---|---|
| 000 | T[0] |
| 001 | T[1] |
| 010 | T[2] |
| 011 | T[3] |
| 100 | T[4] |
| 101 | T[5] |
| 110 | CC |
| 111 | EQ |

DATA = A six-bit conditional branch microaddress, test input mask, or counter value field designated as PI in microinstruction mnemonics.

The special two data field comparison microinstruction format is shown below:

**Am29PL141 Comparison Microinstruction Format**




DFR00740

WHERE:

OE = Synchronous Output Enable for P[15:8].

OPCODE = Compare microinstruction (binary 100).

CONSTANT = A six-bit constant for equal to comparison with T*M.

DATA = A six-bit mask field for masking the incoming T[5:0] inputs.

## Table 1.

| Inputs | | | | Outputs | | | Operation |
|---|---|---|---|---|---|---|---|
| SDI | MODE | DCLK | CLK | SDO | Shadow Register | Pipeline Register | |
| D | L | ↑ | H,L,↑ | $S_O$ | $S_{i-1} \leftarrow S_i$<br>$S_{31} \leftarrow D$ | Hold | Serial Right Shift Register |
| X | L | H,L,↓ | ↑ | $S_O$ | Hold | $P_i \leftarrow PROM_i$ | Normal Load Pipeline Register from PROM |
| L | H | ↑ | H,L,↓ | L | $S_i \leftarrow P_i$ | Hold | Load Shadow Register from Pipeline* Register |
| X | H | H,L,↓ | ↑ | SDI | Hold | $P_i \leftarrow S_i$ | Load Pipeline Register from Shadow Register |
| H | H | ↑ | H,L,↓ | H | Hold | Hold | Hold Shadow Register |

*S7, S6 are undefined. $S_{15} - S_8$ load from the source driving pins P[15] – P[8]. If P[31] in the microword is a ONE, $S_{15}$-$S_8$ are loaded from the pipeline register. If P[31] in the microword is a ZERO, $S_{15} - S_8$ are loaded from an external source.

### Function Table Definitions

**Inputs**

H = HIGH X = Don't Care
L = LOW ↑ = LOW-to-HIGH transition
↓ = High-to-Low transition

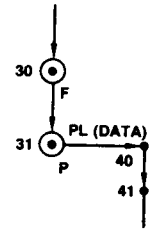
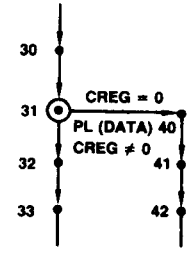
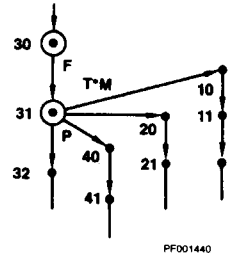
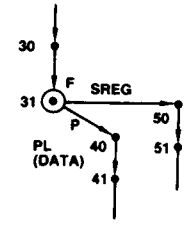
## Table 2.

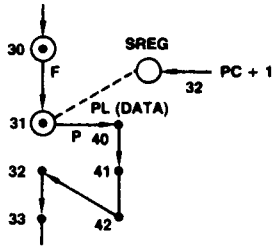
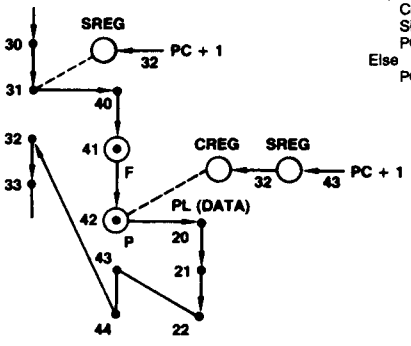
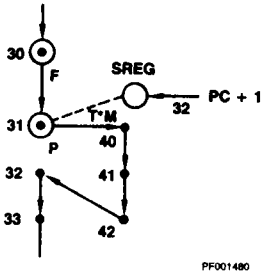
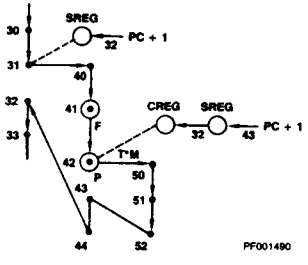
| Input Condition Being Tested | POL | Condition |
|---|---|---|
| 0 | 0 | Fail |
| 0 | 1 | Pass |
| 1 | 0 | Pass |
| 1 | 1 | Fail |

5

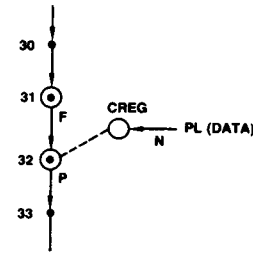
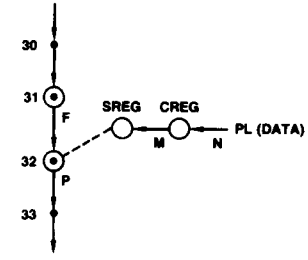
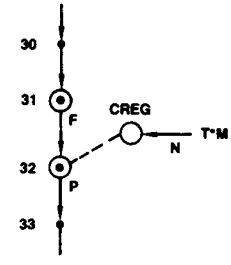
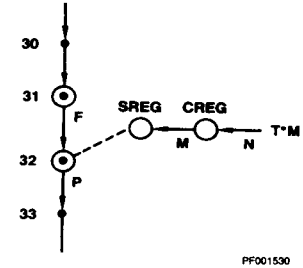
## Am29PL141 Microinstruction Set Definition

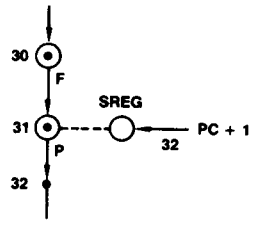
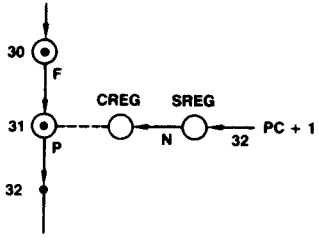
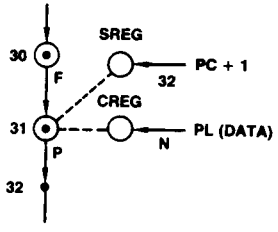
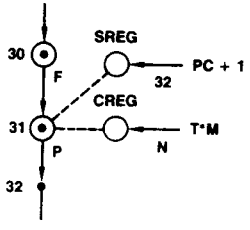
● = Other instruction
⊙ = Instruction being described
$\epsilon$ = Register in part
P = Test Pass
F = Test Fail
M,N are arbitrary values in the CREG or SREG

| Opcode | Mnemonics | Description | Execution Example | Register Transfer Description |
|---|---|---|---|---|
| 19 | GOTOPL | **If (cond) Then Go To Pipeline** Conditional branch to the address in the PL (DATA field). The EQ flag will be reset if the test field selects it and the condition passes. |   PF001420 | If ( cond = true ) Then PC = PL(data) Else PC = PC + 1 |
| 0B | GOTOPLZ | **If (CREG = 0) Then Go To Pipeline** Conditional branch, when the CREG is equal to zero, to the address in the PL (DATA field). This instruction does not depend on the pass/fail condition. The EQ flag will be reset if the test field selects it and the CREG is equal to zero. |   PF001430 | If ( CREG = 0 ) Then PC = PL(data) Else PC = PC + 1 |
| 0F | GOTOTM | **If (cond) Then Go To TM** Conditional branch to the address defined by the T*M (T[5:0] under bitwise mask from the DATA field). This microinstruction is intended for multiway branches. The EQ flag will be reset if the test field selects it and the condition passes. |   PF001440 | If ( cond = true ) Then PC = T*M Else PC = PC + 1 |
| 18 | FORK | **If (cond) Then Go To Pipeline Else Go To (SREG)** Conditional branch to the address in the PL (DATA field) or the SREG. A branch to PL is taken if the condition is true and a branch to SREG if false. The EQ flag will be reset if the test field selects it and the condition passes. |   PF001451 | If ( cond = true ) Then PC = PL(data) Else PC = SREG |

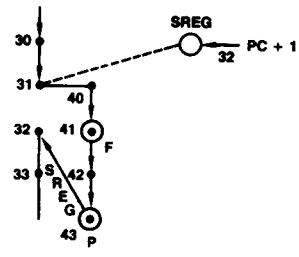
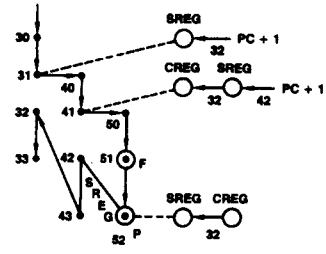
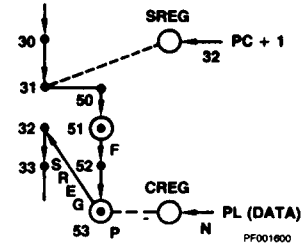
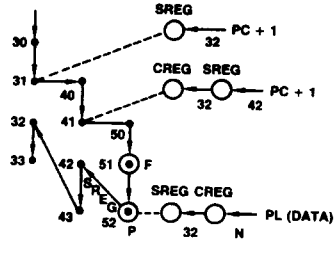
| Opcode | Mnemonics | Description | Execution Example | Register Transfer Description |
|---|---|---|---|---|
| 1C | CALPL | **If (cond) Then Call Pipeline** Conditional jump to subroutine at the address in the PL (DATA field). The PC + 1 is pushed into the SREG as the return address. The EQ flag will be reset if the test field selects it and the condition passes. |  PF001461 | If ( cond = true ) Then<br>SREG = PC + 1<br>PC = PL(data)<br>Else<br>PC = PC + 1 |
| 1D | CALPLN | **If (cond) Then Call Pipeline, Nested** Conditional jump to subroutine at the address in the PL (DATA field) nested. The SREG and CREG are treated as a two-deep stack, the PC + 1 is pushed into the SREG as the return address and the previous SREG value is transferred into the CREG as a nested return address. The EQ flag will be reset if the test field selects it and the condition passes. |  PF001470 | If ( cond = true ) Then<br>CREG = SREG<br>SREG = PC + 1<br>PC = PL(data)<br>Else<br>PC = PC + 1 |
| 1E | CALTM | **If (cond) Then Call TM** Conditional jump to subroutine at the address specified by the T*M (T[5:0] under bitwise mask from the DATA field). The PC + 1 is pushed into the SREG as the return address. The EQ flag will be reset if the test field selects it and the condition passes. |  PF001480 | If ( cond = true ) Then<br>SREG = PC + 1<br>PC = T*M<br>Else<br>PC = PC + 1 |
| 1F | CALTMN | **If (cond) Then Call TM, Nested** Conditional jump to subroutine at the address specified by the T*M (T[5:0] under bitwise mask from the DATA field) nested. The PC + 1 is pushed into the SREG as the return address and the previous SREG value is transferred into the CREG as a nested return address. The EQ flag will be reset if the test field selects it and the condition passes. |  PF001490 | If ( cond = true ) Then<br>CREG = SREG<br>SREG = PC + 1<br>PC = T*M<br>Else<br>PC = PC + 1 |

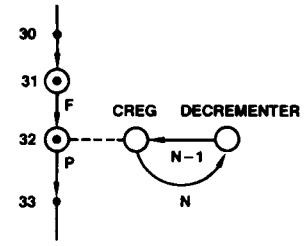
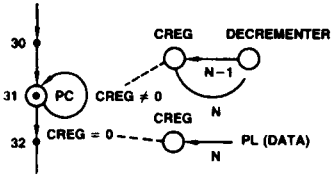
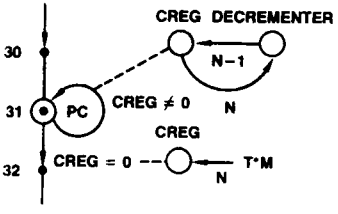
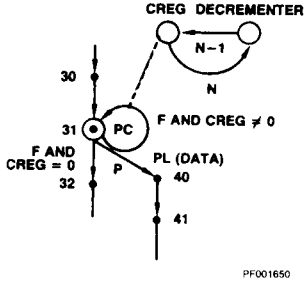
5

| Opcode | Mnemonics | Description | Execution Example | Register Transfer Description |
|---|---|---|---|---|
| 04 | LDPL | **If (cond) Then Load Pipeline**<br>Conditional Load the CREG from the PL (DATA field). |  <br>PF001510 | If ( cond = true ) Then<br>CREG = PL(data)<br>PC = PC + 1<br>Else<br>PC = PC + 1 |
| 05 | LDPLN | **If (cond) Then Load Pipeline, Nested**<br>Conditional load the CREG from the PL (DATA field) nested. The CREG and SREG are treated as a two-deep nested count register, the previous CREG value is pushed into the SREG as a nested count, and the CREG is loaded from PL. |  <br>PF001500 | If ( cond = true ) Then<br>SREG = CREG<br>CREG = PL(data)<br>PC = PC + 1<br>Else<br>PC = PC + 1 |
| 06 | LDTM | **If (cond) Then Load TM**<br>Conditional load the CREG from the T*M (T[5:0] inputs under bitwise mask from the DATA field). |  <br>PF001520 | If ( cond = true ) Then<br>CREG = T*M<br>PC = PC + 1<br>Else<br>PC = PC + 1 |
| 07 | LDTMN | **If (cond) Then Load TM, Nested**<br>Conditional load the CREG from the T*M (T[5:0] inputs under bitwise mask from the DATA field) nested. The SREG and CREG are treated as a two-deep nested count register, the previous CREG value is transferred into the SREG and the CREG is loaded from T*M. |  <br>PF001530 | If ( cond = true ) Then<br>SREG = CREG<br>CREG = T*M<br>PC = PC + 1<br>Else<br>PC = PC + 1 |

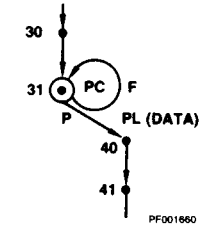
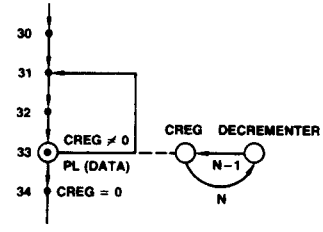
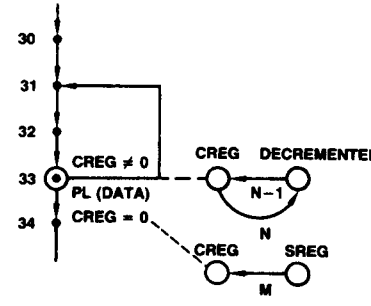
| Opcode | Mnemonics | Description | Execution Example | Register Transfer Description |
|---|---|---|---|---|
| 15 | PSH | **If (cond) Then Push**<br>Conditional push the PC + 1 into the SREG. |  <br>PF0001540 | If ( cond = true ) Then<br>SREG = PC + 1<br>PC = PC + 1<br>Else<br>PC = PC + 1 |
| 17 | PSHN | **If (cond) Then Push, Nested**<br>Conditional push the PC + 1 into the SREG nested. This microinstruction treats the SREG and CREG as a two-deep stack, PC + 1 is pushed into SREG and the previous value in SREG is transferred into the CREG. |  <br>PF001550 | If ( cond = true ) Then<br>CREG = SREG<br>SREG = PC + 1<br>PC = PC + 1<br>Else<br>PC = PC + 1 |
| 14 | PSHPL | **If (cond) Then Push, Load Pipeline**<br>Conditional push the PC + 1 into the SREG and load the CREG from the PL (DATA field). |  <br>PF001560 | If ( cond = true ) Then<br>CREG = PL(data)<br>SREG = PC + 1<br>PC = PC + 1<br>Else<br>PC = PC + 1 |
| 16 | PSHTM | **If (cond) Then Push, Load TM**<br>Conditional push the PC + 1 into the SREG and load the CREG from the T*M (T[5:0] under bitwise mask from the DATA field). |  <br>PF001570 | If ( cond = true ) Then<br>CREG = T*M<br>SREG = PC + 1<br>PC = PC + 1<br>Else<br>PC = PC + 1 |

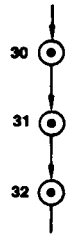
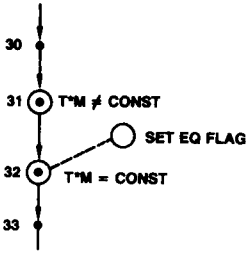
5

| Opcode | Mnemonics | Description | Execution Example | Register Transfer Description |
|---|---|---|---|---|
| 02 | RET | **If (cond) Then Return**<br>Conditional return from subroutine. The SREG provides the return from subroutine address. |  <br>PF001580 | If ( cond = true ) Then<br>PC = SREG<br>Else<br>PC = PC + 1 |
| 03 | RETN | **If (cond) Then Return Nested**<br>Conditional return from nested subroutine. This microinstruction treats the SREG and CREG as a two-deep stack providing the SREG value as a return address and the CREG value as a nested return address that is transferred into the SREG. |  <br>PF001590 | If ( cond = true ) Then<br>PC = SREG<br>SREG = CREG<br>Else<br>PC = PC + 1 |
| 00 | RETPL | **If (cond) Then Return, Load Pipeline**<br>Conditional return from subroutine and load the CREG from the PL (DATA field). The SREG provides the return from subroutine address. |  <br>PF001600 | If ( cond = true ) Then<br>CREG = PL(data)<br>PC = SREG<br>Else<br>PC = PC + 1 |
| 01 | RETPLN | **If (cond) Then Return Nested, Load Pipeline**<br>Conditional return from nested subroutine and load the CREG from the PL (DATA field). This microinstruction treats the SREG and CREG as a two-deep stack providing the SREG value as a return address and the CREG value as a nested return address that is transferred into the SREG. |  <br>PF001610 | If ( cond = true ) Then<br>PC = SREG<br>SREG = CREG<br>CREG = PL(data)<br>Else<br>PC = PC + 1 |

| Opcode | Mnemonics | Description | Execution Example | Register Transfer Description |
|---|---|---|---|---|
| 09 | DEC | **If (cond) Then Decrement**<br>Conditional decrement of the CREG. |  <br>PF001620 | If ( cond = true ) Then<br>CREG = CREG – 1<br>PC = PC + 1<br>Else<br>PC = PC + 1 |
| 0C | DECPL | **While (CREG ≠ 0) Wait Else Load Pipeline**<br>Conditional Hold until the counter is equal to zero, then load CREG from the PL (DATA field). This microinstruction is intended for timing waveform generation. If the CREG is not equal to zero, the same microinstruction is refetched while CREG is decremented. Timing is complete when the CREG is equal to zero, causing the next microinstruction to be fetched and the CREG to be reloaded from PL. This instruction does not depend on the pass/fail condition. |  <br>PF001630 | While ( CREG < > 0)<br>CREG = CREG – 1<br>PC = PC<br>End While<br>CREG = PL(data)<br>PC = PC + 1 |
| 0E | DECTM | **While (CREG ≠ 0) Wait Else Load TM**<br>Conditional Hold until the counter is equal to zero, then load CREG from the T*M (T[5:0] under bitwise mask from the DATA field). This microinstruction is intended for timing waveform generation. If the CREG is not equal to zero, the same microinstruction is refletched while the CREG is decremented. Timing is complete when the CREG is equal to zero, causing the next microinstruction to be fetched and the CREG to be reloaded from T*M. This instruction does not depend on the pass/fail condition. |  <br>PF001640 | While ( CREG < > 0 )<br>CREG = CREG – 1<br>PC = PC<br>End While<br>CREG = T*M<br>PC = PC + 1 |
| 1B | DECGOPL | **If (cond) Then Go To Pipeline Else While (CREG ≠ 0) Wait**<br>Conditional Hold/Count. The current microinstruction will be refetched and the CREG decremented until the condition under test becomes true or the counter is equal to zero. If the condition becomes true, a branch to the address in the PL (DATA field) is executed. If the counter becomes zero without the condition becoming true, a CONTINUE is executed. The EQ flag will be reset if the test field selects it and the condition passes. |  <br>PF001650 | While ( cond = false )<br>If ( CREG < > 0 )<br>CREG = CREG – 1<br>PC = PC<br>Else<br>PC = PC + 1<br>End While<br>PC = PL(data) |

5

| Opcode | Mnemonics | Description | Execution Example | Register Transfer Description |
|---|---|---|---|---|
| 1A | WAIT | **If (cond) Then Go To Pipeline Else Wait**<br>Conditional Hold. The current microinstruction will be refetched and executed until the condition under test becomes true. When true, a branch to the address in the PL (DATA field) is executed. The EQ flag will be reset if the test field selects it and the condition passes. |  <br>PF001660 | If ( cond = true ) Then<br>PC = PL(data)<br>Else<br>PC= PC |
| 08 | LPPL | **While (CREG ≠ 0) Loop to Pipeline**<br>Conditional loop to the address in the PL (DATA field). This microinstruction is intended to be placed at the bottom of an iterative loop. If the CREG is not equal to zero, it is decremented (signifying completion of an iteration), and a branch to the PL address (top of the loop) is executed. If the CREG is equal to zero, looping is complete and the next sequential microinstruction is executed. This instruction does not depend on the pass/fail condition. The EQ flag will be reset if the test field selects it and CREG is not equal to zero. |  <br>PF001670 | While ( CREG < > 0)<br>CREG = CREG – 1<br>PC = PL (data)<br>End While<br>PC = PC + 1 |
| 0A | LPPLN | **While (CREG ≠ 0) Loop to Pipeline Else Nest**<br>Conditional loop to the address in the PL (DATA field) nested. The SREG and CREG are treated as a two-deep nested count register, and the microinstruction is intended to be placed at the bottom of an "inner-nested" iterative loop. If the CREG is not equal to zero, the CREG is decremented (signifying completion of an iteration), and a branch to the PL address (top of the loop) is executed. If the CREG is equal to zero, the inner loop is complete, and the count value for the outer loop is transferred from the SREG into the CREG. This instruction does not depend on the pass/fail condition. The EQ flag will be reset if the test field selects it and CREG is not equal to zero. |  <br>PF001680 | While ( CREG < > 0)<br>CREG = CREG – 1<br>PC = PL(data)<br>End While<br>CREG = SREG<br>PC = PC + 1 |

| Opcode | Mnemonics | Description | Execution Example | Register Transfer Description |
|---|---|---|---|---|
| 0D | CONT | **Continue** The next sequential microinstruction is fetched unconditionally. |  PF001690 | PC = PC + 1 |
| 10 – 13 (100XX binary) | CMP | **Compare TM to Pipeline (DATA)** This microinstruction performs bitwise exclusive-or of T*M (T[5:0] under bitwise mask from the DATA field) with CONSTANT (P[27:22]). If T*M equals CONSTANT, the EQ flag is set to one which may be branched on in a following microinstruction. If not equal, the EQ flag is unaffected. This allows sequences of compares, in a manner analogous to sum-of-products, to be performed which can be followed by a single conditional branch if one or more of the comparisons were true. Note: The EQ flag is set to zero on reset or when EQ is selected as the condition in a branch. Conditional returns on EQ leave the flag unchanged. **Constant field bits that correspond to masked test field bits must be zero.** This instruction does not depend on the pass/fail condition. |  PF001701 | Compare T*M and PL(data)<br>EQ = ((T [5:0] .AND. DATA).<br>XNOR. CONSTANT) .OR. EQ |

5

## Microinstruction Set Table

| Code | Mnemonics | Definition | CREG Content | Pass | | | | Fail | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | | PC MUX | SREG | CREG | EQ | PC MUX | SREG | CREG | EQ |
| 00 | RETPL | Return: Load Pipeline | X | SREG | Hold | Data | NC | PC + 1 | Hold | Hold | NC |
| 01 | RETPLN | Return Nested: Load Pipeline | X | SREG | CREG | Data | NC | PC + 1 | Hold | Hold | NC |
| 02 | RET | Return | X | SREG | Hold | Hold | NC | PC + 1 | Hold | Hold | NC |
| 03 | RETN | Return Nested | X | SREG | CREG | Hold | NC | PC + 1 | Hold | Hold | NC |
| 04 | LDPL | Load Pipeline | X | PC + 1 | Hold | Data | NC | PC + 1 | Hold | Hold | NC |
| 05 | LDPLN | Load Pipeline Nested | X | PC + 1 | CREG | Data | NC | PC + 1 | Hold | Hold | NC |
| 06 | LDTM | Load T*M | X | PC + 1 | Hold | T*M | NC | PC + 1 | Hold | Hold | NC |
| 07 | LDTMN | Load T*M Nested | X | PC + 1 | CREG | T*M | NC | PC + 1 | Hold | Hold | NC |
| 08 | LPPL | Loop Pipeline | ≠ 0 | Data | Hold | DCRMT | Reset | | | | |
| | | | = 0 | PC + 1 | Hold | Hold | NC | | | | |
| 09 | DEC | Decrement | X | PC + 1 | Hold | DCRMT | NC | PC + 1 | Hold | Hold | NC |
| 0A | LPPLN | Loop Pipeline Nested | ≠ 0 | Data | Hold | DCRMT | Reset | | | | |
| | | | = 0 | PC + 1 | Hold | SREG | NC | | | | |
| 0B | GOTOPLZ | Go to Pipeline Zero | ≠ 0 | PC + 1 | Hold | Hold | NC | | | | |
| | | | = 0 | Data | Hold | Hold | Reset | | | | |
| 0C | DECPL | Count/Load Pipeline | ≠ 0 | PC | Hold | DCRMT | NC | | | | |
| | | | = 0 | PC + 1 | Hold | Data | NC | | | | |
| OD | CONT | Continue | X | PC + 1 | Hold | Hold | NC | PC + 1 | Hold | Hold | NC |
| 0E | DECTM | Count/Load T*M | ≠ 0 | PC | Hold | DCRMT | NC | | | | |
| | | | = 0 | PC + 1 | Hold | T*M | NC | | | | |
| 0F | GOTOTM | Go to T*M | X | T*M | Hold | Hold | Reset | PC + 1 | Hold | Hold | NC |
| 10 – 13 (100XX Binary) | CMP | Compare* | X | PC + 1 | Hold | Hold | Set | PC + 1 | Hold | Hold | NC |
| 14 | PSHPL | Push: Load Pipeline | X | PC + 1 | PC + 1 | Data | NC | PC + 1 | Hold | Hold | NC |
| 15 | PSH | Push | X | PC + 1 | PC + 1 | Hold | NC | PC + 1 | Hold | Hold | NC |
| 16 | PSHTM | Push: Load T*M | X | PC + 1 | PC + 1 | T*M | NC | PC + 1 | Hold | Hold | NC |
| 17 | PSHN | Push Nested | X | PC + 1 | PC + 1 | SREG | NC | PC + 1 | Hold | Hold | NC |
| 18 | FORK | Fork | X | Data | Hold | Hold | Reset | SREG | Hold | Hold | NC |
| 19 | GOTOPL | Go to Pipeline | X | Data | Hold | Hold | Reset | PC + 1 | Hold | Hold | NC |
| 1A | WAIT | Hold Pipeline | X | Data | Hold | Hold | Reset | PC | Hold | Hold | NC |
| 1B | DECGOPL | Count: Hold Pipeline | ≠ 0 | Data | Hold | Hold | Reset | PC | Hold | DCRMT | NC |
| | | | = 0 | Data | Hold | Hold | Reset | PC + 1 | Hold | Hold | NC |
| 1C | CALPL | Call Pipeline | X | Data | PC + 1 | Hold | Reset | PC + 1 | Hold | Hold | NC |
| 1D | CALPLN | Call Pipeline Nested | X | Data | PC + 1 | SREG | Reset | PC + 1 | Hold | Hold | NC |
| 1E | CALTM | Call T*M | X | T*M | PC + 1 | Hold | Reset | PC + 1 | Hold | Hold | NC |
| 1F | CALTMN | Call T*M Nested | X | T*M | PC + 1 | SREG | Reset | PC + 1 | Hold | Hold | NC |

EQ = ((T[5:0]. AND. DATA). XNOR. CONSTANT).OR. EQ
CONSTANT field bits that correspond to masked test field bits must be zero.
NC = No Change

**Notes:**
1. (/)Signifies two different operations may occur, depending on the condition.
2. (:) Signifies two parallel operations on the same condition.
3. The EQ flag will be affected only if the test field selects it, with the exception of instructions 10 – 13.

### PROGRAMMING YIELD

AMD programmable logic devices have been designed to insure extremely high programming yields ( > 98%).

AMD programmable logic devices contain many internal test features, including circuitry and extra fuses which allow AMD to test the ability of each part to perform programming before shipping, to assure high programming yields, and correct logical operation for a correctly programmed part. Programming yield losses are most likely due to poor programming socket contact, programming equipment that is out of calibration, or improper usage of said equipment.

## Programmers/Development Systems

(refer to Programmer Reference Guide, page 3-81)

## Absolute Maximum Ratings

Storage Temperature ..... −65 to +150°C
(Ambient) Temperature Under Bias ..... −55 to +125°C
Supply Voltage to Ground Potential
(Pin 28 to Pin 14) Continuous ..... −0.5 V to +7.0 V
DC Voltage Applied to Outputs
(Except During Programming) ..... −0.5 V to +$V_{CC}$ Max.
DC Voltage Applied to Outputs
During Programming ..... 21 V
DC Output Current, Into Outputs During
Programming (Max Duration of 1 sec) ..... 200 mA
DC Input Voltage ..... −0.5 V to +5.5 V
DC Input Current ..... −30 mA to +5.0 mA

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

## Operating Ranges

Commercial (C) Devices
Temperature ..... 0 to +70°C
Supply Voltage ..... +4.75 V to +5.25 V

Military (M) Devices
Temperature ..... −55 to +125°C
Supply Voltage ..... +4.5 V to +5.5 V

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

## DC Characteristics

over operating range unless otherwise specified; included in Group A, Subgroup 1, 2, 3 tests unless otherwise noted

| Parameters | Description | Test Conditions | | | | Min | Max | Units |
|---|---|---|---|---|---|---|---|---|
| $V_{OH}$ | Output HIGH Voltage | $V_{CC}$ = Min., $V_{IN}$ = $V_{IH}$ or $V_{IL}$ | | $I_{OH}$ = −3.0 mA | COM'L | 2.4 | | Volts |
| | | | | $I_{OH}$ = −1.0 mA | MIL | | | |
| $V_{OL}$ | Output LOW Voltage | $V_{CC}$ = Min., $V_{IN}$ = $V_{IH}$ or $V_{IL}$ | | $I_{OL}$ = 16 mA | COM'L | | 0.50 | Volts |
| | | | | $I_{OL}$ = 12 mA | MIL | | | |
| $V_{IH}$ (Note 1) | Input HIGH Level | Guaranteed Input Logical HIGH Voltage for All Inputs | | | | 2.0 | | Volts |
| $V_{IL}$ (Note 1) | Input LOW Level | Guaranteed Input Logical LOW Voltage for All Inputs | | | | | 0.8 | Volts |
| $I_{IL}$ | Input LOW Current | $V_{CC}$ = Max. $V_{IN}$ = 0.5 V | CLK<br>P [15:6]<br>All other Inputs | | | | −1.5<br>−0.55<br>−0.50 | mA |
| $I_{IH}$ | Input HIGH Current | $V_{CC}$ = Max. $V_{IN}$ = 2.4 V | CLK<br>P [15:6]<br>All other Inputs | | | | 150<br>100<br>25 | µA |
| $I_I$ | Input HIGH Current | $V_{CC}$ = Max., $V_{IN}$ = 5.5 V | | | | | 1.0 | mA |
| $I_{SC}$ | Output Short Circuit Current | $V_{CC}$ = Max., $V_{OUT}$ = 0.5 V (Note 2) | | | | −20 | −80 | mA |
| $I_{CC}$ | Power Supply Current | $V_{CC}$ = Max. | COM'L | $T_A$ = 0 to 70°C | | | 450 | mA |
| | | | | $T_A$ = 70°C | | | 400 | |
| | | | MIL | $T_C$ = −55 to 125°C | | | 490 | |
| | | | | $T_C$ = 125°C | | | 420 | |
| $V_I$ | Input Clamp Voltage | $V_{CC}$ = Min, $I_{IN}$ = −18 mA | | | | | −1.2 | Volts |
| $I_{OZH}$ | Output Leakage Current (Note 3) | $V_{CC}$ = MAX, $V_{IL}$ = 0.8 V $V_{IH}$ = 2.0 V | | $V_O$ = 2.4 V | | | 100 | µA |
| $I_{OZL}$ | | | | $V_O$ = 0.5 V | | | −550 | |

**Notes:**
1. These are absolute values with respect to device ground and all overshoots due to system or tester noise are included.
2. Not more than one output should be tested at a time. Duration of the short circuit should not be more than one second. $V_{OUT}$ = 0.5 V has been chosen to avoid test problems caused by tester ground degradation.
3. I/O pin leakage is the worst case of $I_{OZX}$ or $I_{IX}$ (where X = H or L).

## Switching Characteristics

over operating range unless otherwise specified; included in Group A, Subgroup 9, 10, 11 tests unless otherwise noted (APL and CPL products only)

| Parameters | | Description | Test Conditions | COMMERCIAL | | MILITARY | | Units |
|---|---|---|---|---|---|---|---|---|
| | | | | Min. | Max. | Min. | Max. | |
| $t_{PD}$ | 1 | CLK to P[15:0] | See Test Output Load Conditions | | 15 | | 20 | ns |
| | 2 | CLK to $\overline{ZERO}$ | | | 20 | | 25 | ns |
| | 3 | DCLK to SDO | | | 30 | | 35 | ns |
| | 4 | Mode to SDO | | | 30 | | 35 | ns |
| | 5 | SDI to SDO | | | 30 | | 35 | ns |
| $t_S$ | 6 | T[5:0] to CLK (Note 1) | | 40 | | 45 † | | ns |
| | 7 | CC to CLK (Note 1) | | 40 | | 45 † | | ns |
| | 8 | $\overline{RESET}$ to CLK | | 30 | | 35 | | ns |
| | 9 | Mode to CLK | | 30 | | 35 | | ns |
| | 10 | Mode to DCLK | | 30 | | 35 | | ns |
| | 11 | SDI to DCLK | | 30 | | 35 | | ns |
| | 12 | P[15:8] to DCLK | | 30 | | 35 | | ns |
| $t_H$ | 13 | T[5:0] to CLK | | 3 | | 3 | | ns |
| | 14 | CC to CLK | | 3 | | 3 | | ns |
| | 15 | $\overline{RESET}$ to CLK | | 3 | | 3 | | ns |
| | 16 | Mode to CLK | | 3 | | 3 | | ns |
| | 17 | Mode to DCLK | | 3 | | 3 | | ns |
| | 18 | SDI to DCLK | | 3 | | 3 | | ns |
| | 19 | P[15:8] to DCLK | | 3 | | 3 | | ns |
| $t_{PZX}$ | 20 | CLK to P[15:8] Enable | | | 30 | | 35 | ns |
| $t_{PXZ}$ | 21 | CLK to P[15:8] Disable | | | 30 | | 35 | ns |
| $t_{PW}$ | 22 | CLK Pulse Width (HIGH and LOW) | | 20 | | 25 | | ns |
| | 23 | DCLK Pulse Width (HIGH and LOW) | | 30 | | 35 | | ns |
| $t_P$ | 24 | CLK and DCLK Period (Note 1) | | 45 | | 50 † | | ns |

**Notes:**

1. These parameters cannot be measured directly on unprogrammed devices. They are determined as follows:
   a. Measure delay from input (CC, T[5:0], or CLK) to PROM address out in test mode. This will measure the delay through the sequence logic.
   b. Measure setup time from T[5:0] input through PROM test columns to pipeline register in verify test column mode. This will measure the delay through the PROM and register setup.
   c. Measure delay from T[5:0] input to PROM address out in verify test column mode. This will measure the delay through the logic and P[15:0] outputs.

   To calculate the desired parameter measurement the following formula is used:

   Measurement (a) + Measurement (b) – Measurement (c)

   CLK PERIOD:

   CLK (a) + (b) – (c) = CLK PERIOD

   CC to CLK Set-up time:

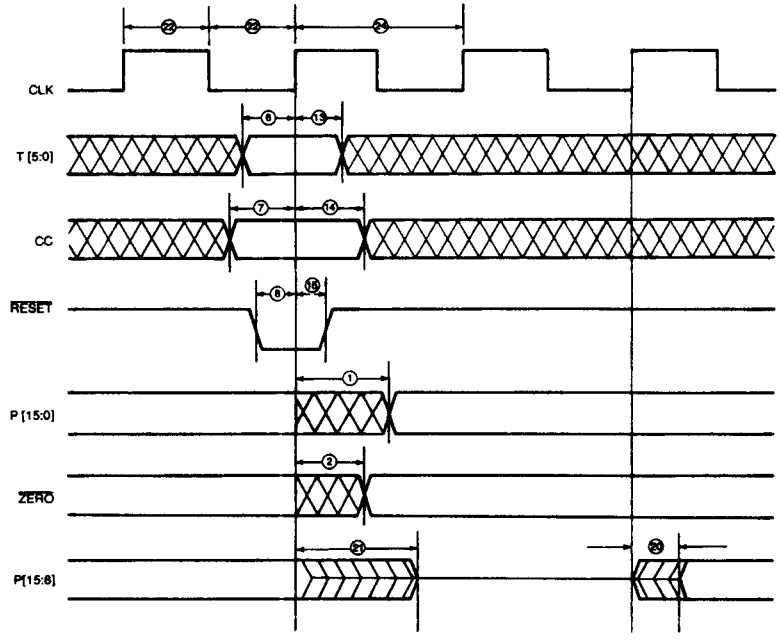
   CC (a) + (b) – (c) = CC to CLK Set-up time

   T[5:0] to CLK Set-up time:

   T[5:0] (a) + (b) – (c) = T[5:0] to CLK Set-up time

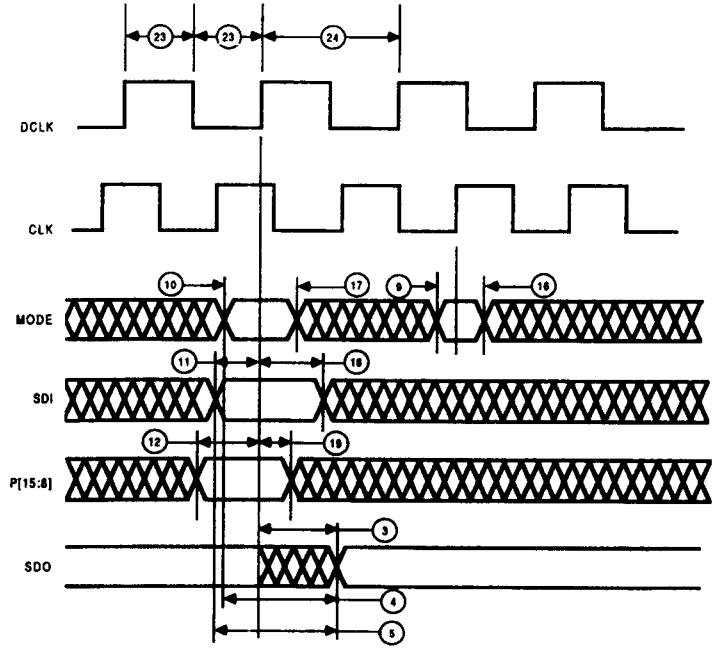
† = Not included in Group A tests

## Switching Waveforms




WF020852

**Normal Configuration**




WF023120

**SSR™ Configuration**

## 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 testing environment. The specifics of what philosophies are applied to which test are shown in the data sheet and the data-sheet reconciliation that follow.

### 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 smaller capacitive load than the associated stray capacitance. Typical examples of this are the so-called "float delays" that measure the propagation delays in to 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 setup are used to determine 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 impractical 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 determined from engineering correlations based on data taken with a bench setup and the knowledge that certain DC tests are performed in order to facilitate this correlation.

AC loads specified in the data sheet are used for bench testing. Automatic tester loads, which simulate the data-sheet loads, may be used during production testing.

### Threshold Testing

The noise associated with automatic testing, the long inductive cables, and the high gain of bipolar devices frequently give rise to oscillations when testing high-speed circuits. These oscillations are not indicative of a reject device, but instead, of an overtaxed 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.

### AC Testing

AC parameters are specified that cannot be measured accurately on automatic testers because of tester limitations. Data-input hold times fall into this category. In these cases, the parameter in question is tested by correlating the tester to bench data or oscilloscope measurements made on the tester by engineering (supporting data on file).

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.

### Output Short-Circuit Current Testing

When performing $I_{OS}$ tests on devices containing RAM or registers, great care must be taken that undershoot caused by grounding the high-state output does not trigger parasitic elements which in turn cause the device to change state. In order to avoid this effect, it is common to make the measurement at a voltage ($V_{output}$) that is slightly above ground. The $V_{CC}$ is raised by the same amount so that the result (as confirmed by Ohm's law and precise bench testing) is identical to the $V_{OUT} = 0$, $V_{CC}$ = Max. case.

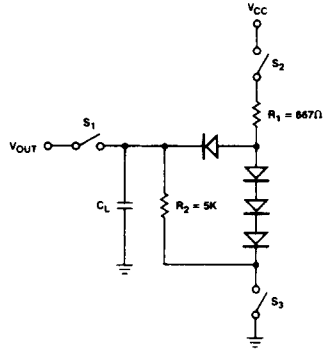
5

## 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 |

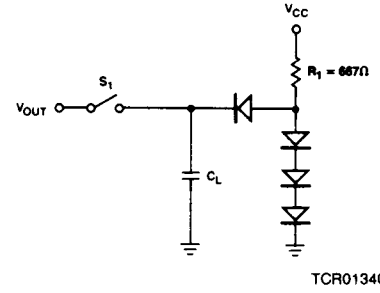
KS000010

## Switching Test Circuits




TCR01330

**A. Three State Outputs**

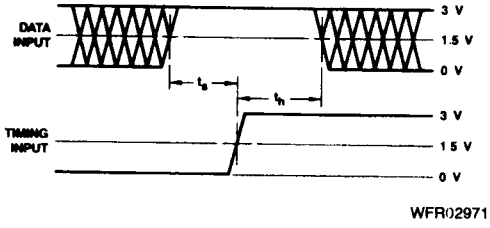



TCR01340

**B. Normal Outputs**

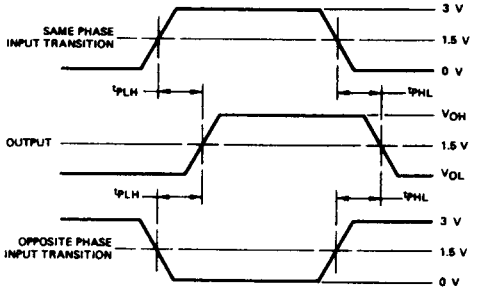
Notes: 1. CL = 50 pF includes scope probe, wiring and stray capacitances without device in test fixture.
2. $S_1$, $S_2$, and $S_3$ are closed during function tests and all AC tests except output enable tests.
3. $S_1$ and $S_3$ are closed while $S_2$ is open for $tp_{ZH}$ test.
4. $C_L$ = 5.0 pF for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

## Switching Test Waveforms




WFR02971

### Set-up, Hold, and Release Times

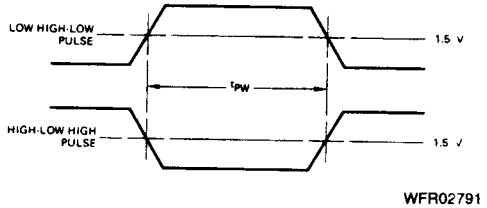



WFR02980

### Propagation Delay

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

### Pulse Width




WFR02791

### Enable and Disable Times

| Test | $V_X$ | Output Waveform – Measurement Level |
|---|---|---|
| All $t_{PD}$s | 5.0V | $V_{OH}$, $V_{OL}$, 1.5V |
| $t_{PHZ}$ | 0.0V | $V_{OH}$, 0.5V, 0.0V |
| $t_{PLZ}$ | 5.0V | 3V, $V_{OL}$, 0.5V |
| $t_{PZH}$ | 0.0V | $V_{OH}$, 0.0V, 1.5V |
| $t_{PZL}$ | 5.0V | 3V, 1.5V, $V_{OL}$ |

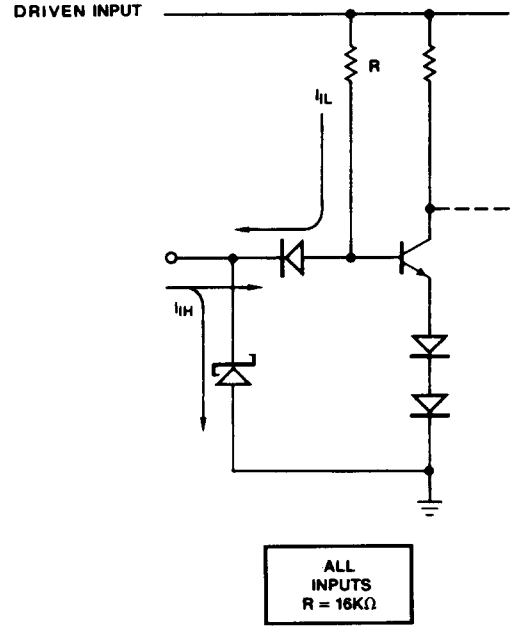
WFR02680

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.

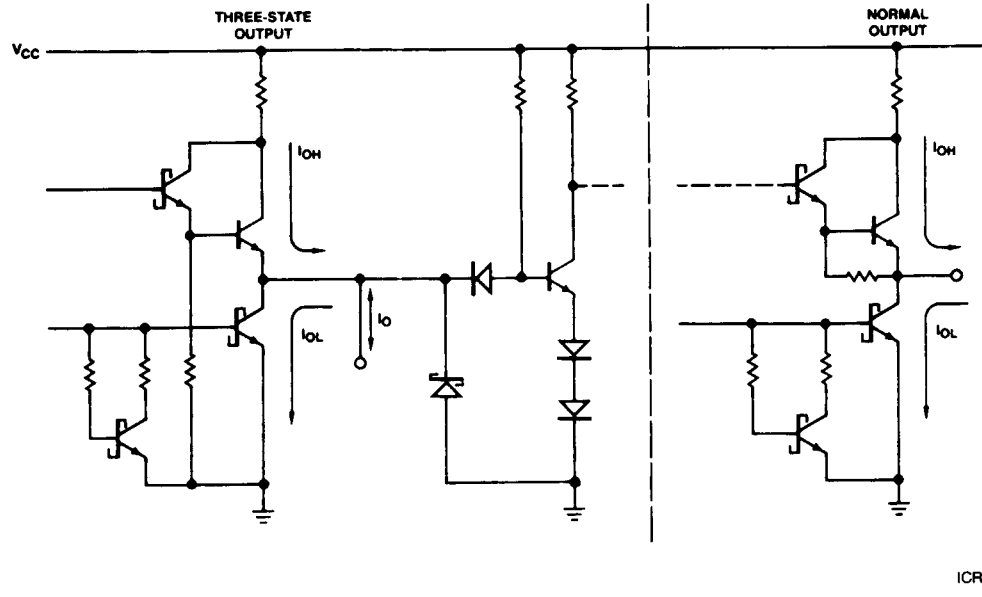
NOTE: Pulse generator for all pulses: Rate $\leq$ 1.0 MHz; $Z_0 = 50\ \Omega$: $t_r \leq 2.5$ ns.

5

## Input/Output Current Interface Conditions




$C_O \cong 5.0$ pF, all inputs




$C_O \cong 5.0$ pF, all outputs

NOTE: Actual current flow direction shown.