

# Am29C111

CMOS 16-Bit Microprogram Sequencer

PRELIMINARY

Am29C111

Advanced Micro Devices

## DISTINCTIVE CHARACTERISTICS

- **16-Bit Address up to 64K Words**
- **Supports a 70-ns system cycle time**
- **Speed Select**  
Supports an 80-ns system cycle time
- **Real-Time Interrupt Support**  
Micro-trap and interrupts are handled transparently at any microinstruction boundary.
- **Built-In Conditional Test Logic**  
Has nine external test inputs, four of which are used to internally generate four additional test conditions. Test multiplexer selects one out of 13 test inputs.
- **Break-Point Logic**  
Built-in address comparator allows break-points in the microcode for debugging and statistics collection.
- **33-Level Stack**  
Provides support for interrupts, loops, and subroutine nesting. It can be accessed through the D-bus to support diagnostics.

## GENERAL DESCRIPTION

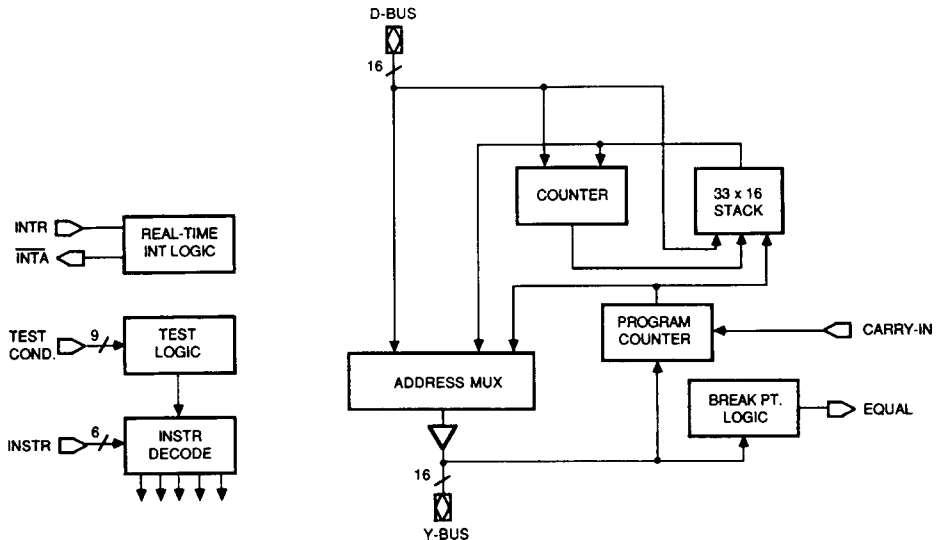
The Am29C111 is a 16-bit wide, high-speed single-chip sequencer designed to control the execution sequence of microinstructions stored in the microprogram memory. The instruction set is designed to resemble high-level language constructs, thereby bringing high-level language programming to the micro level.

The Am29C111 is interruptible at any microinstruction boundary to support real-time interrupts. Interrupts are handled transparently to the microprogrammer as an unexpected procedure call. Traps are also handled transparent-

ly at any microinstruction boundary. This feature allows re-execution of the prior microinstruction. The 33-deep stack provides the ability to support interrupts, loops, and subroutine nesting. The stack can be read through the D-bus to support diagnostics or to implement multitasking at the micro-architecture level.

Fabricated using Advanced Micro Devices' 1.2 micron CMOS process, the Am29C111 is powered by a single 5-volt supply. The device is housed in a 68-pin PLCC, or a 68-lead PGA package.

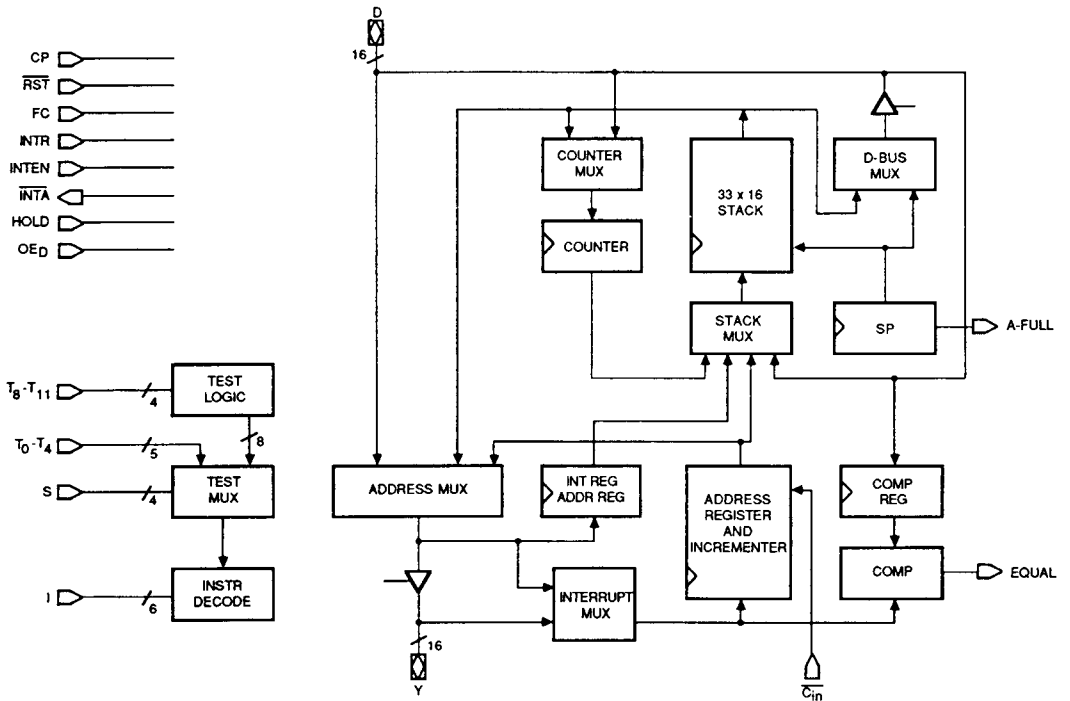
## SIMPLIFIED BLOCK DIAGRAM



BD007521

## RELATED AMD PRODUCTS

Part No.	Description
Am29C101	CMOS 16-Bit Microcontroller Slice
Am29114	Vectored Priority Interrupt Controller
Am29116	High-Performance Bipolar 16-Bit Microprocessor
Am29C116	High-Performance CMOS 16-Bit Microprocessor
Am29PL141	Field-Programmable Controller
Am29C323	CMOS 32-Bit Parallel Multiplier
Am29325	32-Bit Floating-Point Processor
Am29C325	CMOS 32-Bit Floating-Point Processor
Am29332	32-Bit Extended Function ALU
Am29C332	CMOS 32-Bit Extended Function ALU
Am29334	64 x 18 Four-Port, Dual-Access Register File
Am29C334	CMOS 64 x 18 Four-Port, Dual-Access Register File
Am29337	16-Bit Bounds Checker
Am29338	Byte Queue



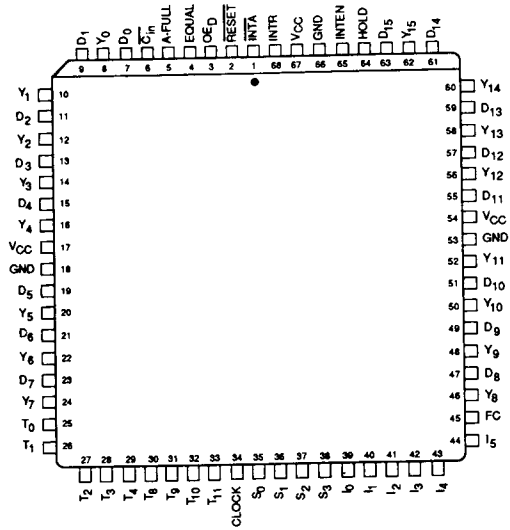
BD007510

Figure 1. Am29C111 Detailed Block Diagram

# CONNECTION DIAGRAMS

## PLCC

### (Top View)

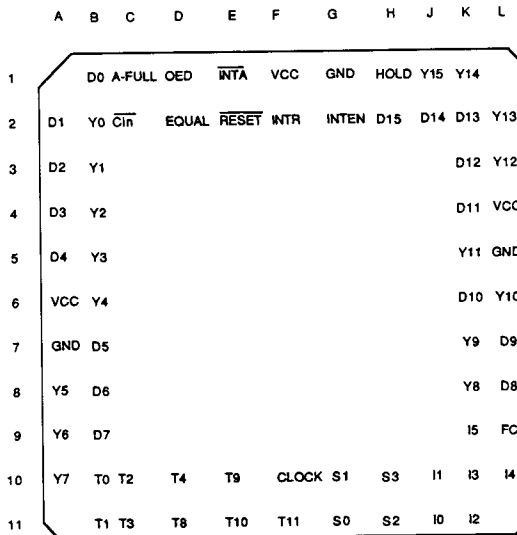


CD011160

Note: Pin 1 is marked for orientation.

## PGA

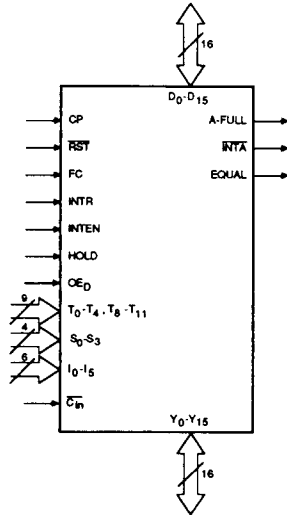
### (Bottom View)



CD011081

Note: Notch indicates orientation.

## LOGIC SYMBOL



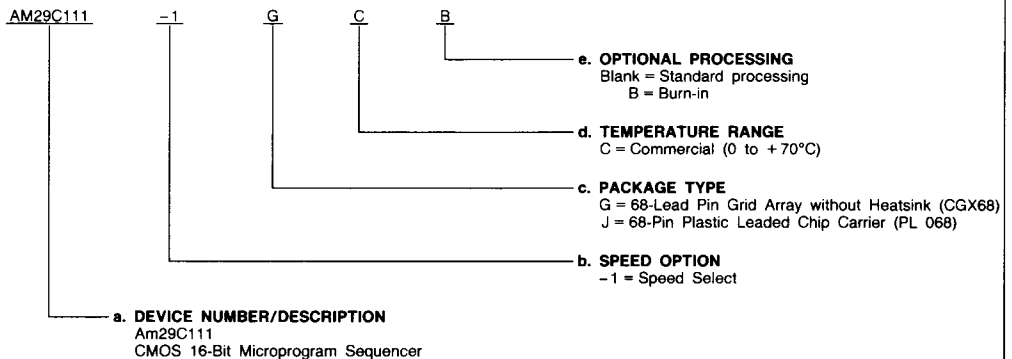
CD011090

## 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 valid combinations, and to obtain additional data on AMD's standard military grade products.

Valid Combinations	
AM29C111	GC, GCB, JC, JCB
AM29C111-1	

## PIN DESCRIPTION

### **A-FULL Almost Full (Output; Three-State)**

Indicates that  $28 \leq SP \leq 63$  (meaning there are five or less empty locations left on stack). Also active during stack-under flow.

### **$\overline{C}_{in}$ Carry In (Input; Active LOW)**

Carry-in to the incrementer.

### **CP Clock Pulse (Input)**

Clocks sequencer at the LOW-to-HIGH transition.

### **D<sub>0</sub> - D<sub>15</sub> Data (Bidirectional; Three-State)**

Input to address multiplexer, counter, stack, and comparator register. Output for stack and stack pointer.

### **EQUAL Equal (Output; Three-State)**

Indicates that the address comparator is enabled and has found a match.

### **FC Force Continue (Input)**

Overrides instruction with CONTINUE.

### **HOLD Hold (Input)**

Stops the sequencer and three-states the outputs.

### **I<sub>0</sub> - I<sub>5</sub> Instruction (Input)**

Selects one of 64 instructions.

### **INTA Interrupt Acknowledge (Output; Three-State, Active LOW)**

Indicates that an interrupt is accepted.

### **INTEN Interrupt Enable (Input)**

Enables interrupts.

### **INTR Interrupt Request (Input)**

Requests the sequencer to interrupt execution.

### **OE<sub>D</sub> Output Enable — D-Bus (Input)**

Enables the D-bus driver, provided that the sequencer is not in the hold or slave mode.

### **RST Reset (Input; Active LOW)**

Resets the sequencer.

### **S<sub>0</sub> - S<sub>3</sub> Select (Input)**

Selects one of 16 test conditions.

### **T<sub>0</sub> - T<sub>4</sub>, T<sub>8</sub> - T<sub>11</sub> Test (Input)**

Provides external test inputs.

### **Y<sub>0</sub> - Y<sub>15</sub> Address (Bidirectional; Three-State)**

Output of microcode address. Input for interrupt address.

## FUNCTIONAL DESCRIPTION

### Architecture

The major blocks of the sequencer are the address multiplexer, the address register (AR), the stack (with the top of stack denoted TOS), the counter (C), the test multiplexer with logic, and the address comparison register (R) (Figure 1). The bidirectional D-bus provides branch addresses and iteration counts; it also allows access to the stack from the outside. The bidirectional Y-bus either outputs microprogram addresses or inputs interrupt addresses. The buses are all 16 bits wide. Figure 1 shows a detailed block diagram of the sequencer.

### Address Multiplexer

The address multiplexer can select an address from any of three sources:

- 1) A branch address supplied by the D-bus
- 2) A return or loop address from the top of stack
- 3) The next sequential address from the incrementer

### Address Register and Incrementer

The address register contains the current address. It is loaded from the interrupt multiplexer and feeds the incrementer. The incrementer is inhibited if  $\overline{C}_{in}$  is taken HIGH.

### Stack

A 33-word-deep and 16-bit-wide stack provides first-in last-out storage for return addresses, loop addresses, and counter values. Items to be pushed come from the incrementer, the interrupt-return-address register, the counter, or the D-bus. Items popped go to the address multiplexer, the counter, or the D-bus.

The access to the stack via the D-bus may be used for context switching, stack extension, or diagnostics. As the stack is only accessible from the top, stack extension is done by temporarily storing the whole or some lower part of the stack outside the sequencer. The save and the later restore are done with pop and push operations, respectively, at balanced points in the microprogram; for example, points with the same stack depth. The internal D-bus driver must be turned on when popping an item to the D-bus; if the driver is off, the item will be unstacked

instead. The driver is normally turned on when the Output Enable signal is asserted and the sequencer is not being reset ( $OE_D = 1$ ,  $RST = 1$ ).

The stack pointer is a modulo 64 counter, which is incremented on each push and decremented on each pop. The stack pointer is reset to zero when the sequencer is reset, but the pointer may also be reset by instruction. Thus, the stack pointer indicates the number of items on the stack as long as stack overflow or underflow has not occurred. Overflow happens when an item is pushed onto a full stack, whereby the item at the bottom of the stack is overwritten. Underflow happens when an item is popped from an empty stack; in this case the item is undefined.

In the case of stack overflow, the SP is incremented for every push after overflow. Thus, immediately after the first occurrence of stack overflow, the SP will be equal to 34. Subsequent pushes will increment the SP to 35, 36...61, 62, 73, 1, 0, etc. In the case of the stack underflow, the SP is decremented for every pop after underflow. Thus, immediately after the first occurrence of stack underflow, the SP will be equal to 63. Subsequent pops will decrement the SP to 62, 61...2, 1, 0, etc.

The contents of the stack pointer are present on the D-bus for all instructions except POP D, provided the driver is turned on. The output signal, A-FULL, is active under the following conditions:  $28 \leq SP \leq 63$ .

### Counter

The counter may be used as a loop counter. It may be loaded from the D-bus, or via a pop from the stack. Its contents may also be pushed onto the stack.

A normal for-loop is set up by a FOR instruction, which loads the counter from the D- or A-bus with the desired number of iterations; the instruction also pushes onto the stack a loop address that points to the next sequential instruction. The end of the loop is given by an unconditional END FOR instruction, which tests the counter value against the value one and then decrements the counter. If the values differ, the loop is repeated by selecting the address at the stack as the next address. If the values are equal, the loop is terminated by popping the stack, thereby removing the loop address, and

selecting the address from the incrementer as the next address. The number of iterations is a 16-bit unsigned number, except that the number zero corresponds to 65,536 iterations. By pushing and popping counter values it is possible to handle nested loops.

### Address Comparison

The sequencer is able to compare the address from the interrupt multiplexer with the contents of the comparator register. The instruction SET loads the comparator register with the address on the D-bus and enables the comparison, while CLEAR disables it. The comparison is disabled at reset. A HIGH is present at the output EQUAL if the comparison is enabled and the two addresses are equal. The comparison is useful for detection of a break point or counting the number of times a microinstruction at a specific address is executed.

### Instruction Set

The sequencer has 64 instructions that are divided into four classes of 16 instructions each. The instruction lines  $I_0 - I_5$  use  $I_5$  and  $I_4$  to select a class, and  $I_0 - I_3$  to select an instruction within a class. The classes are:

$I_5$	$I_4$	Classes
0	0	Conditional sequence control,
0	1	Conditional sequence control with inverted polarity,
1	0	Unconditional sequence control, and
1	1	Special function with implicit continue.

Note that for the first three classes  $I_5$  forces the condition to be true and  $I_4$  inverts the condition. The basic instructions of the first three classes are shown in Table 1 and the instructions of the fourth class in Table 2.

Structured microprogramming is supported by sequencer instructions that singly or in pairs correspond to high-level language control constructs. Examples are FOR  $I: = D$  DOWN TO 1 DO . . . END FOR and CASE N OF . . . END CASE. The instructions have been given high-level language names where appropriate. Figure 2 shows how to microprogram important control constructs; the high-level language is on the left and the microcode on the right.

### Test Conditions

The condition for a conditional instruction is supplied by a test multiplexer, which selects one out of sixteen tests with the select lines  $S_0 - S_3$ . Nine of these are supplied directly by the inputs  $T_0 - T_4$  and  $T_8 - T_{11}$ , while the remaining four tests are generated by the test logic from the inputs  $T_8 - T_{11}$ . The following table shows the assignments.  $T_5 - T_7$  are internally set as true.

$(S_3 - S_0)_H$	Test	Intended Use
0 - 4	$T_0 - T_4$	General
5 - 7	$T_5 - T_7$	True
8	$T_8$	C (Carry)
9	$T_9$	N (Negative)
A	$T_{10}$	V (Overflow)
B	$T_{11}$	Z (Zero or equal)
C	$T_8 + T_{11}$	C + Z (Unsigned less than or equal, borrow mode)
D	$\overline{T_8} + T_{11}$	$\overline{C} + Z$ (Unsigned less than or equal)
E	$T_9 \oplus T_{10}$	$N \oplus V$ (Signed less than)
F	$(T_9 \oplus T_{10}) + T_{11}$	$(N \oplus V) + Z$ (Signed less than or equal)

### Force Continue

The sequencer has a force continue (FC) input, which overrides the instruction inputs  $I_0 - I_5$  with a CONTINUE instruction. This makes it possible to share the microinstruction field for the sequencer instruction with some other control or to initialize a writable control store.

### Reset

In order to start a microprogram properly, the sequencer must be reset. The reset works like an instruction overriding both the instruction input and the force continue input. The reset selects the address 0 at the address multiplexer, forces the EQUAL output to LOW, and disregards a potential interrupt request. It synchronously disables the address comparison and initializes the stack pointer to 0. The contents of the stack are invalid after a reset.

**TABLE 1. INSTRUCTION SET for I<sub>5</sub>I<sub>4</sub> = 00, 01, 10**

I <sub>5</sub> - I <sub>0</sub>	Instruction	Cond.: Fail		Cond.: Pass		Counter	Comp.	D-Mux
		Y	Stack	Y	Stack			
00, 10, 20	Goto D	INC	-	D	-	-	-	SP
01, 11, 21	Call D	INC	-	D	Push INC	-	-	SP
02, 12, 22	Exit D	INC	-	D	Pop	-	-	SP
03, 13, 23	End for D, C ≠ 1	INC	-	D	-	C←C-1	-	SP
	End for D, C = 1	INC	-	INC	-	C←C-1	-	SP
0C, 1C, 2C	End Loop	INC	Pop	TOS	-	-	-	SP
0D, 1D, 2D	Call Coroutine	INC	-	TOS	Pop & Push INC	-	-	SP
0E, 1E, 2E	Return	INC	-	TOS	Pop	-	-	SP
0F, 1F, 2F	End for, C ≠ 1	INC	Pop	TOS	-	C←C-1	-	SP
	End for, C = 1	INC	Pop	INC	Pop	C←C-1	-	SP

Cond. = (Test [S] OR I<sub>5</sub>) XOR I<sub>4</sub>

: = Concatination

C = Counter

INC = Output of Incrementer = AR + 1 (if  $\overline{C_{in}}$  = LOW)

Note: For unconditional instructions, the action marked under Cond.:Pass is taken.

**TABLE 2. INSTRUCTION SET for I<sub>5</sub>I<sub>4</sub> = 11**

I <sub>5</sub> - I <sub>0</sub>	Instruction	Y	Stack	Counter	Comp.	D-Mux
30	Continue	INC	-	-	-	SP
31	For D	INC	Push INC	C←D	-	SP
32	Decrement	INC	-	C←C-1	-	SP
33	Loop	INC	Push INC	-	-	SP
34	Pop D	INC	Pop	-	-	TOS
35	Push D	INC	Push D	-	-	SP
36	Reset SP	INC	SP←0	-	-	SP
38	Pop C	INC	Pop	C←TOS	-	SP
39	Push C	INC	Push C	-	-	SP
3A	Swap	INC	TOS←C	C←TOS	-	SP
3B	Push C Load D	INC	Push C	C←D	-	SP
3C	Load D	INC	-	C←D	-	SP
3E	Set	INC	-	-	R←D, Enable	SP
3F	Clear	INC	-	-	Disable	SP

R = Comp. Register

## Interrupts

The sequencer may be interrupted at the completion of the current microcycle by asserting the interrupt request input INTR. The return address of the interrupted routine is saved on the stack so that nested interrupts can be easily implemented. An interrupt is accepted if interrupts are enabled and the sequencer is not being reset or held (INTEN = HIGH,  $\overline{RST}$  = HIGH, and HOLD = LOW). The interrupt-acknowledge output ( $\overline{INTA}$ ) goes LOW when an interrupt is accepted.

When there is no interrupt, addresses go from the address multiplexer to the Y-bus via the driver, and to the address register and the comparator via the interrupt multiplexer. When there is an interrupt, the driver of the sequencer is turned off, an external driver is turned on, and the interrupt multiplexer is switched. The interrupt address is supplied via the external driver to the Y-bus, the address register, and the comparator (Figure 3). In order to save the address from the address multiplexer, the address is stored in the interrupt return address register, which for simplicity is clocked every cycle. The next microinstruction is the first microinstruction of the interrupt routine (Figure 4).

In this cycle the address in the interrupt return address register is automatically pushed onto the stack. Therefore the microinstruction in this cycle must not use the stack; if a stack operation is programmed, the result is undefined. The instructions that do not use the stack are GOTO D, GOTO A, GOTO M, CONTINUE, DECREMENT, LOAD D, LOAD A, SET and CLEAR. A RETURN instruction terminates the interrupt routine and the interrupted routine is resumed. Interrupts only work with a single-level control path.

## Traps

A trap is an unexpected situation linked to current microinstruction that must be handled before the microinstruction completes and changes the state of the system. An example of such a situation is an attempt to read a word from memory across a word boundary in a single cycle. When a trap occurs, the current microinstruction must be aborted and re-executed after the execution of a trap routine, which in the meantime will take corrective measures. An interrupt, on the other hand, is

not linked directly to the current microinstruction that can complete safely before an interrupt routine is executed.

Execution of a trap requires that the sequencer ignores the current microinstruction, selects the trap return address at the address multiplexer, and initiates an interrupt. This will save the trap return address on the stack and issue the trap address from an external source (Figure 5). The address register contains the address of the microinstruction in the pipeline register, thus the address register already contains the trap return address when a trap occurs. This address can be selected by the address multiplexer by disabling the incrementer ( $\overline{C_{in}} = 1$ ), and using the force continue mode (FC = 1). In this mode the sequencer ignores the current microinstruction. The remaining part of the trap handling is done by the interrupt (Figure 6), thus the section on interrupts also applies to traps. There is one exception, however. The interrupt enable cannot be used as a trap enable as it does not control the force continue mode and the carry-in to the incrementer.

## Hold Mode

The sequencer has a hold mode in which the operation is suspended.

The outputs (Y,  $\overline{INTA}$ , A-FULL & EQUAL) are disabled and the sequencer enters the hold mode immediately after the Hold signal goes active. While the sequencer is in this mode, the internal state is left unchanged and the D-bus is disabled. The outputs (Y,  $\overline{INTA}$ , A-FULL & EQUAL) are enabled again and the sequencer leaves the hold mode immediately after the Hold signal goes inactive.

In a time-multiplexed multimicroprocess system there may be one sequencer for all processes with microprogrammed context save and restore, or there may be one sequencer per microprocess permitting fast process switch. In the latter case the Y-buses of the sequencers are tied together and connected to a single microprogram store. A control unit decides on a cycle-by-cycle basis what sequencer should be running, and activates the HOLD signal to the remaining sequencers. The hold mode has higher priority than interrupts, and works independently of the reset. The hold mode can only be used with a single-level control path.

## High-Level Language Constructs

An example of high-level language constructs using Am29C111 instructions is given in Figure 2 (2-1, 2-2, and 2-3).

```

REPEAT          LOOP
-              -
-              -
UNTIL CC       END LOOP NOT CC

WHILE CC DO    LOOP
-              IF NOT CC THEN EXIT L
-              -
-              -
END WHILE      END LOOP
                L:

LOOP           LOOP
-             -
IF CC THEN EXIT IF CC THEN EXIT L
-             -
END LOOP      END LOOP
                L:
    
```

**Figure 2-1. Loops with Unknown Number of Iterations**

```

FOR CNT: = 10 DOWN TO 1 DO  FOR D 10
-                             -
-                             -
END FOR                      END FOR
    
```

**Figure 2-2. Loop with Known Number of Iterations**

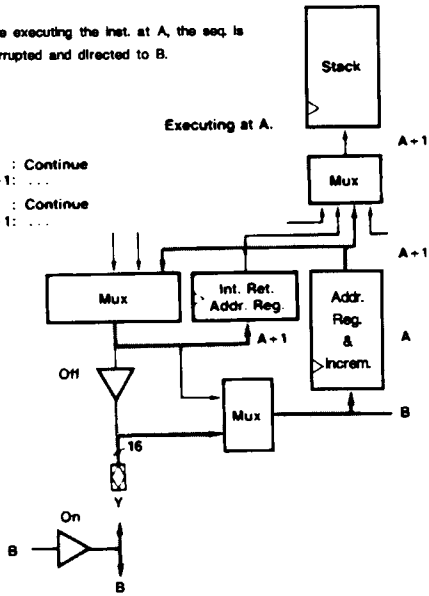
```

IF X THEN          PUSH D C
IF Y THEN          IF NOT X THEN GOTO A
-                 IF NOT Y THEN GOTO B
-                 -
ELSE               -, RETURN (TO C)
-                 B:
-                 -, RETURN (TO C)
END IF
ELSE               A:
IF Z THEN          IF NOT Z THEN GOTO D
-                 -
-                 -, RETURN (TO D)
ELSE               D:
-                 -
-                 -, RETURN (TO C)
END IF
END IF            C:
    
```

**Figure 2-3. Double-Nested If Statement**

While executing the inst. at A, the seq. is interrupted and directed to B.

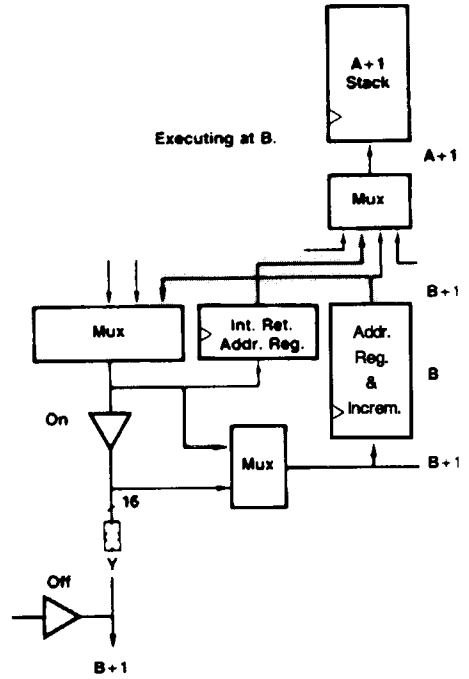
A : Continue  
 A+1: ....  
 B : Continue  
 B+1: ....



AF004193

Figure 3. Am29C111 Interrupt Cycle 1

Executing at B.

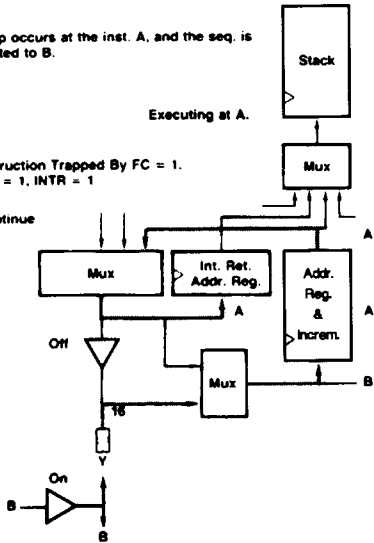


AF004213

Figure 4. Am29C111 Interrupt Cycle 2

A trap occurs at the inst. A, and the seq. is directed to B.

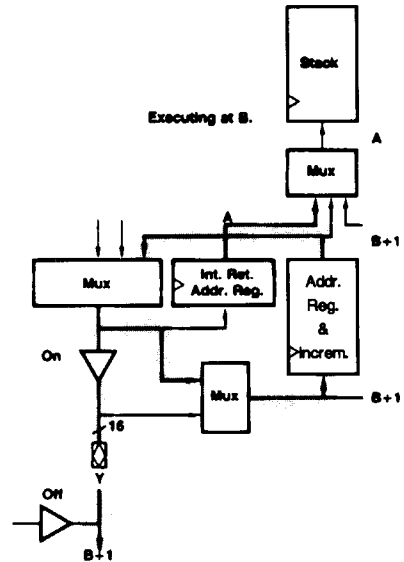
A : Instruction Trapped By FC = 1.  
 $C_{IN} = 1, INTR = 1$   
 A+1: ....  
 B : Continue  
 B+1: ....



AF004202

Figure 5. Am29C111 Traps Cycle 1

Executing at B.



AF004183

Figure 6. Am29C111 Traps Cycle 2

# Instruction Set Definition

Legend: ● = Other instruction  
 ○ = Instruction being described  
 CC = (Test [S<sub>3</sub> - S<sub>0</sub>])

P = Test pass  
 F = Test fail  
 ○ = Register in part

Opcode (I <sub>5</sub> - I <sub>0</sub> )	Mnemonics	Description	Execution Example
20H	BRA_D	GOTO D Unconditional branch to the address specified by the D inputs. The D port must be disabled to avoid bus contention.	
2CH	BRA_S	GOTO TOS Unconditional branch to the address on the top of the stack.	
PF001730			
00H	BRCC_D	IF CC THEN GOTO D ELSE CONTINUE If CC is HIGH (pass), branch to the address specified by D. If CC is LOW (fail), continue. The D port must be disabled to avoid bus contention.	
0CH	BRCC_S	IF CC THEN GOTO TOS ELSE POP STACK CONTINUE If CC is HIGH (pass), branch to the address on the top of the stack. If CC is LOW (fail), pop the stack and continue.	
PF001740			
10H	BRNC_D	IF NOT CC THEN GOTO D ELSE CONTINUE If CC is LOW (pass), branch to the address specified by D. If CC is HIGH (fail), continue. The D Port must be disabled to avoid Bus contention.	
1CH	BRNC_S	IF NOT CC THEN GOTO TOS ELSE POP STACK CONTINUE If CC is LOW (pass), branch to the address on the top of the stack. If CC is HIGH (fail), pop the stack and continue.	
PF001750			

Note: Opcode numbers are in hexadecimal notation.

Opcode (I <sub>5</sub> - I <sub>0</sub> )	Mnemonics	Description	Execution Example
21H	CALL_D	<b>CALL D</b> Unconditional branch to the subroutine specified by the D inputs. Push the return address (address Reg. + 1) on the stack. The D port must be disabled to avoid bus contention.	
	2DH	<b>CALL TOS</b> Unconditional branch to the subroutine specified by the address on the top of the stack. The stack is popped and the return address (Address Reg. + 1) is then pushed onto the stack.	<p style="text-align: right;">PF001760</p>
01H	CCC_D	<b>IF CC, THEN CALL D</b> <b>ELSE CONTINUE</b> If CC is HIGH (pass), call the subroutine specified by the D inputs. Push the return address (Address Reg. + 1) on the stack. If CC is LOW (fail), continue. The D port must be disabled to avoid bus contention.	
	0DH	<b>IF CC, THEN CALL TOS</b> <b>ELSE CONTINUE</b> If CC is HIGH (pass), call the subroutine specified by the address on the top of the stack. The stack is popped and the return address (Address Reg. + 1) is pushed onto the stack. If CC is LOW (fail), continue.	<p style="text-align: right;">PF001770</p>
11H	CNC_D	<b>IF NOT CC, THEN CALL D</b> <b>ELSE CONTINUE</b> If CC is LOW (pass), call the subroutine specified by the D inputs. Push the return address (Address Reg. + 1) on the stack. If CC is HIGH (fail), continue. The D port must be disabled to avoid bus contention.	
	1DH	<b>IF NOT CC, THEN CALL TOS</b> <b>ELSE CONTINUE</b> If CC is LOW (pass), call the subroutine specified by the address on the top of the stack. The stack is popped and the return address (Address Reg. + 1) is pushed onto the stack.	<p style="text-align: right;">PF001780</p>

Note: Opcode numbers are in hexadecimal notation.

Opcode (I <sub>5</sub> - I <sub>0</sub> )	Mnemonics	Description	Execution Example
22H	EXIT_D	EXIT TO D Unconditional branch to the address specified by the D inputs and pop the stack. The D port must be disabled to avoid bus contention.	
2EH	EXIT_S	EXIT TO TOS Unconditional branch to the address on the top of the stack and pop the stack. Also used for unconditional returns.	
02H	XTCC_D	IF CC, THEN EXIT TO D ELSE CONTINUE If CC is HIGH (pass), exit to the address specified by the D inputs and pop the stack. If CC is LOW (fail), continue with no pop. The D port must be disabled to avoid bus contention.	
0EH	XTCC_S	IF CC, THEN EXIT TO TOS ELSE CONTINUE If CC is HIGH (pass), exit to the address on the top of the stack and pop the stack. If CC is LOW (fail), continue with no pop. Also used for conditional returns.	
12H	XTNC_D	IF NOT CC, THEN EXIT TO D ELSE CONTINUE If CC is LOW (pass), exit to the address specified by the D inputs and pop the stack. If CC is HIGH (fail), continue with no pop. The D port must be disabled to avoid bus contention.	
1EH	XTNC_S	IF NOT CC, THEN EXIT TO TOS ELSE CONTINUE If CC is LOW (pass), exit to the address on the top of the stack and pop the stack. If CC is HIGH (fail), continue with no pop. Also used for conditional returns.	

Note: Opcode numbers are in hexadecimal notation.

Opcode (I <sub>5</sub> - I <sub>0</sub> )	Mnemonics	Description	Execution Example
23H	DJMP_D	<p>IF CNT ≠ 1 THEN CNT: = CNT - 1 GOTO D ELSE CNT: = CNT - 1 CONTINUE</p> <p>If the counter is not equal to one, decrement the counter and branch to the address specified by the D inputs. If the counter is equal to one, then decrement the counter and continue. The D port must be disabled to avoid bus contention.</p>	
2FH	DJMP_S	<p>IF CNT ≠ 1 THEN CNT: = CNT - 1 GOTO TOS ELSE CNT: = CNT - 1 POP STACK CONTINUE</p> <p>If the counter is not equal to one, decrement the counter and branch to the address on the top of the stack. If the counter is equal to one, then decrement the counter, pop the stack and continue.</p>	
03H	DJCC_D	<p>IF CC AND CNT ≠ 1 THEN CNT: = CNT - 1 GOTO D ELSE CNT: = CNT - 1 CONTINUE</p> <p>If CC is HIGH (pass) and the counter is not equal to one, decrement the counter and branch to the address specified by the D inputs. If CC is LOW (fail) or the counter is equal to one, then decrement the counter and continue. The D port must be disabled to avoid bus contention.</p>	
0FH	DJCC_S	<p>IF CC AND CNT ≠ 1 THEN CNT: = CNT - 1 GOTO TOS ELSE CNT: = CNT - 1 POP STACK CONTINUE</p> <p>If CC is HIGH (pass) and the counter is not equal to one, decrement the counter and branch to the address on the top of the stack. If CC is LOW (fail) or the counter is equal to one, then decrement the counter, pop the stack and continue.</p>	
13H	DJNCC_D	<p>IF NOT CC AND CNT ≠ 1 THEN CNT: = CNT - 1 GOTO D ELSE CNT: = CNT - 1 CONTINUE</p> <p>If CC is LOW (pass) and the counter is not equal to one, decrement the counter and branch to the address specified by the D inputs. If CC is HIGH (fail) or the counter is equal to one, then decrement the counter and continue. The D port must be disabled to avoid bus contention.</p>	
1FH	DJNCC_S	<p>IF NOT CC AND CNT ≠ 1 THEN CNT: = CNT - 1 GOTO TOS ELSE CNT: = CNT - 1 POP STACK CONTINUE</p> <p>If CC is LOW (pass) and the counter is not equal to one, decrement the counter and branch to the address on the top of the stack. If CC is HIGH (fail) or the counter is equal to one, then decrement the counter, pop the stack and continue.</p>	

PF001820

PF001830

PF001840

Note: Opcode numbers are in hexadecimal notation.

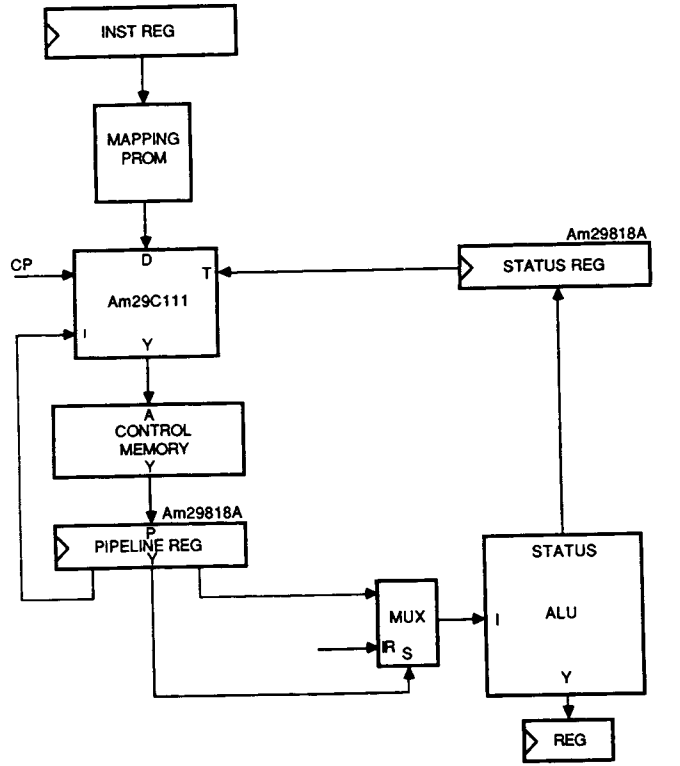
Opcode (I <sub>5</sub> - I <sub>0</sub> )	Mnemonics	Description	Execution Example
2EH	RET	RETURN Unconditional return from subroutine. The return address is popped from the stack.	
0EH	RETCC	IF CC THEN RETURN ELSE CONTINUE If CC is HIGH (pass), return from subroutine. The return address is popped from the stack. If CC is LOW (fail), continue.	
1EH	RETNC	IF NOT CC THEN RETURN ELSE CONTINUE If CC is LOW (pass), return from subroutine. The return address is popped from the stack. If CC is HIGH (fail), continue.	
			PF001850
31H	FOR_D	INITIALIZE LOOP Push the Address Reg. + 1 on the stack, load the counter from the D inputs and continue. Use with DJUMP_S for FOR ... NEXT loops. The D port must be disabled to avoid bus contention.	
33H	LOOP	INITIALIZE LOOP Push the Address Reg. + 1 on the stack and continue. Use with BRCC_S for REPEAT ... UNTIL loops, or with XTCC_D and BRA_S for WHILE ... END WHILE loops.	
			PF001860
34H	POP_D	Pop the stack and output the value on the D outputs and continue. The D port must be enabled.	
38H	POP_C	Pop the stack and store the value in the counter and continue.	
35H	PUSH_D	Push the D inputs on the stack and continue. The D port must be disabled to avoid bus contention.	
39H	PUSH_C	Push the counter on the stack and continue.	
3AH	SWAP	Exchange the counter and the top of stack and continue.	
			PF001870

Note: Opcode numbers are in hexadecimal notation.

Opcode (I <sub>5</sub> - I <sub>0</sub> )	Mnemonics	Description	Execution Example
3B <sub>H</sub>	STACK_C	Push the counter on the stack and load the counter with the value of the D inputs and continue.	
3C <sub>H</sub>	LOAD_C	Load the counter with the value of the D inputs and continue. The D port must be disabled to avoid bus contention.	
PF001880			
30 <sub>H</sub>	CONT	Continue.	
32 <sub>H</sub>	DECR	Decrement the counter and continue.	
36 <sub>H</sub>	RESET_SP	Reset the stack pointer and continue.	
PF001890			
3E <sub>H</sub>	SET	Load the comparison register with the value of the D inputs, enable the comparator and continue.	
3F <sub>H</sub>	CLEAR	Disable the comparator and continue.	
PF001900			

Note: Opcode numbers are in hexadecimal notation.

# APPLICATIONS



BD007680

**System Block Diagram**

## CONTROL PATH TIMING ANALYSIS (Preliminary)

				29C111	29C10A	29C111-1	29C10A-1	Type	
I.	<b>Pipeline Register</b>	(29818A)							
	Mapping PROM	(27S190A)							
	Register	(29818A)	CP-Q	12 ns	12 ns	12 ns	12 ns	Branch Map	
	Sequencer		D-Y	18	20	15	18		
	Control Memory	(99C59-25)	tAA	25	25	25	25		
Pipeline Register	(29818A)	Setup	4	4	4	4			
<b>Cycle Time:</b>				59 ns	61 ns	56 ns	59 ns		
II.	<b>Pipeline Register</b>	(29818A)	CP-Q	12 ns	12 ns	12 ns	12 ns	Branch	
	Buffer Enable	(2959)	OE-Y	NA	20	NA	20		
	Sequencer		I, D-Y	20	35	17	33		
	Control Memory	(99C59-25)	tAA	25	25	25	25		
	Pipeline Register	(29818A)	Setup	4	4	4	4		
	<b>Cycle Time:</b>				61 ns	96 ns	58 ns	94 ns	
	<b>Pipeline Register</b>	(29818A)	CP-Q	12 ns	12 ns	12 ns	12 ns	Conditional Branch Using External Status Register	
	CC-MUX	(2923)	Sel-W	NA	15	NA	15		
	Polarity	(74S86)	D-Y	NA	11	NA	11		
	Sequencer		T, CC-Y	22	30	18	26		
Control Memory	(99C59-25)	tAA	25	25	25	25			
Pipeline Register	(29818A)	Setup	4	4	4	4			
<b>Cycle Time:</b>				63 ns	97 ns	59 ns	93 ns		
IV.	<b>Pipeline Register</b>	(29818A)	CP-Q	12 ns	12 ns	12 ns	12 ns	Instruction to Output Path	
	Sequencer		I-Y	20	35	17	33		
	Control Memory	(99C59-25)	tAA	25	25	25	25		
	Pipeline Register	(29818A)	Setup	4	4	4	4		
<b>Cycle Time:</b>				61 ns	76 ns	58 ns	74 ns		
V.	<b>Sequencer</b>	(29818A)	CP-Y	38 ns	24 ns	31 ns	33 ns	Clock to Output Path	
	Control Memory	(99C59-25)	tAA	25	25	25	25		
	Pipeline Register	(29818A)	Setup	4	4	4	4		
<b>Cycle Time:</b>				67 ns	53 ns	60 ns	62 ns		

## ABSOLUTE MAXIMUM RATINGS

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

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

## OPERATING RANGES

Commercial (C) Devices	
Ambient Temperature ( $T_A$ )	0 to +70°C
Supply Voltage ( $V_{CC}$ )	+4.75 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.	Unit
$V_{OH}$	Output HIGH Voltage	$V_{CC} = \text{Min.}$ , $V_{IN} = V_{IL}$ or $V_{IH}$	$I_{OH} = 0.4$ mA	2.4		V
$V_{OL}$	Output LOW Voltage	$V_{CC} = \text{Min.}$ , $V_{IN} = V_{IL}$ or $V_{IH}$	$I_{OL} = 8$ mA for Y-Bus $I_{OL} = 4$ mA for All other		0.5	V
$V_{IH}$	Input HIGH Level	Guaranteed Input Logical HIGH Voltage for All Inputs		2.0		V
$V_{IL}$	Input LOW Level	Guaranteed Input Logical LOW Voltage for All Inputs			0.8	V
$I_{IL}$	Input LOW Current	$V_{CC} = \text{Max.}$ , $V_{IN} = 0$			-10	$\mu\text{A}$
$I_{IH}$	Input HIGH Current	$V_{CC} = \text{Max.}$ , $V_{IN} = V_{IH}$			10	$\mu\text{A}$
$I_I$	Input HIGH Current	$V_{CC} = \text{Max.}$ , $V_{IN} = V_{CC} - 0.5$ V			10	mA
$I_{OZH}$	Off-State High Impedance Output Current	$V_{CC} = \text{Max.}$		$V_O = 2.4$ V	10	$\mu\text{A}$
$I_{OZL}$	Off-State Low Impedance Output Current			$V_O = 0.5$ V	-10	
$I_{CC}$	Static Power Supply Current (Note 3)	$V_{CC} = \text{Max.}$		CMOS $V_{IN} = V_{OL}$ or GND (Note 4)	20	mA
				TTL $V_{IN} = 0.5$ V or 2.4 V (Note 4)	35	
$C_{PD}$	Power Dissipation Capacitance (Note 5)	$V_{CC} = 5.0$ V $T_A = 25^\circ\text{C}$ No Load		1600 pF Typical		

- Notes: 1.  $V_{CC}$  conditions shown as Min. or Max. refer to the commercial ( $\pm 5\%$ )  $V_{CC}$  limits.  
 2. These input levels provide zero-noise immunity and should only be statically tested in a noise-free environment (not functionally tested).  
 3. Worst-case  $I_{CC}$  is measured at the lowest temperature in the specified operating range.  
 4. Use CMOS  $I_{CC}$  when the device is driven by CMOS circuits and TTL  $I_{CC}$  when the device is driven by TTL circuits.  
 5.  $C_{PD}$  determines the dynamic current consumption:  
 $I_{CC}(\text{Total}) = I_{CC}(\text{Static}) + (C_{PD} + n C_L) \frac{f}{2}$ , where  $f$  is the clock frequency,  $C_L$  output load capacitance, and  $n$  number of loads.

**SWITCHING CHARACTERISTICS** over operating range

**A. COMBINATIONAL PROPAGATION DELAYS**

No.	From	To	29C111	29C111-1	Unit
			Max. Delay	Max. Delay	
1	D <sub>15-0</sub>	Y <sub>15-0</sub>	18	15	ns
	D <sub>15-0</sub>	EQUAL	28	23	ns
	Y <sub>15-0</sub>	EQUAL	27	23	ns
2	I <sub>5-0</sub>	Y <sub>31-0</sub>	20	17	ns
	I <sub>5-0</sub>	D <sub>15-0</sub>	25	21	ns
3	I <sub>5-0</sub>	EQUAL	31	25	ns
	T <sub>11-0</sub>	Y <sub>15-0</sub>	22	18	ns
4	T <sub>11-0</sub>	EQUAL	31	26	ns
	S <sub>3-0</sub>	Y <sub>15-0</sub>	21	18	ns
	S <sub>3-0</sub>	EQUAL	32	27	ns
	CP	Y <sub>15-0</sub>	38	31	ns
5	CP	D <sub>15-0</sub>	24/Z	20/Z	ns
	CP	A-FULL	23	19	ns
7	CP	EQUAL	48	38	ns
	RST	Y <sub>15-0</sub>	32/Z	26/Z	ns
8	RST	D <sub>15-0</sub>	Z	Z	ns
	RST	INTA	18	15	ns
	RST	EQUAL	40	31	ns
10	FC	Y <sub>15-0</sub>	18	15	ns
	FC	D <sub>15-0</sub>	23	19	ns
11	FC	EQUAL	28	23	ns
	INTR	Y <sub>15-0</sub>	Z	Z	ns
12	INTR	INTA	12	11	ns
	INTR	EQUAL	(Note 1)	(Note 1)	ns
	INTEN	Y <sub>15-0</sub>	Z	Z	ns
	INTEN	INTA	12	11	ns
13	INTEN	EQUAL	(Note 1)	(Note 1)	ns
	HOLD	Y <sub>15-0</sub>	Z	Z	ns
	HOLD	INTA	Z	Z	ns
	HOLD	A-FULL	Z	Z	ns
	HOLD	EQUAL	26/Z	23/Z	ns
	OEP	D <sub>15-0</sub>	Z	Z	ns
	C <sub>in</sub>	Y <sub>15-0</sub>	19	16	ns
C <sub>in</sub>	EQUAL	30	24	ns	

Notes: See notes following Table D.

**B. OUTPUT DISABLE TIME**

No.	From	To	Description	Typ. (Note 4)	Max. (Note 3)	Max. (Note 3)	Unit
					29C111	29C111-1	
37	RST	Y <sub>15-0</sub>	Reset-to-Address Enable	-	25	21	ns
	RST	Y <sub>15-0</sub>	Reset-to-Address Disable	13	32	25	ns
38	INTR	Y <sub>15-0</sub>	INTR-to-Address Enable	-	21	18	ns
	INTR	Y <sub>15-0</sub>	INTR-to-Address Disable	13	39	36	ns
39	INTEN	Y <sub>15-0</sub>	INTEN-to-Address Enable	-	21	18	ns
	INTEN	Y <sub>15-0</sub>	INTEN-to-Address Disable	13	39	36	ns
40	HOLD	Y <sub>15-0</sub>	HOLD-to-Address Enable	-	19	17	ns
	HOLD	Y <sub>15-0</sub>	HOLD-to-Address Disable	13	39	36	ns
41	OEP	D <sub>15-0</sub>	OEP-to-Data Enable	-	23	21	ns
	OEP	D <sub>15-0</sub>	OEP-to-Data Disable	13	31	29	ns
42	RST	D <sub>15-0</sub>	Reset-to-Data Enable	-	26	23	ns
	RST	D <sub>15-0</sub>	Reset-to-Data Disable	13	33	30	ns
43	CP	D <sub>15-0</sub>	Clock-to-Data Enable	-	32	27	ns
	CP	D <sub>15-0</sub>	Clock-to-Data Disable	13	34	31	ns
44	HOLD	INTA	HOLD-to-INTA Enable	-	19	17	ns
	HOLD	INTA	HOLD-to-INTA Disable	13	27	25	ns
45	HOLD	A-FULL	HOLD-to-A-FULL Enable	-	20	18	ns
	HOLD	A-FULL	HOLD-to-A-FULL Disable	13	27	26	ns
46	HOLD	EQUAL	HOLD-to-EQUAL Enable	-	20	18	ns
	HOLD	EQUAL	HOLD-to-EQUAL Disable	13	28	26	ns

Notes: See notes following Table D.

# SWITCHING CHARACTERISTICS (Cont'd.)

## C. SETUP AND HOLD TIMES

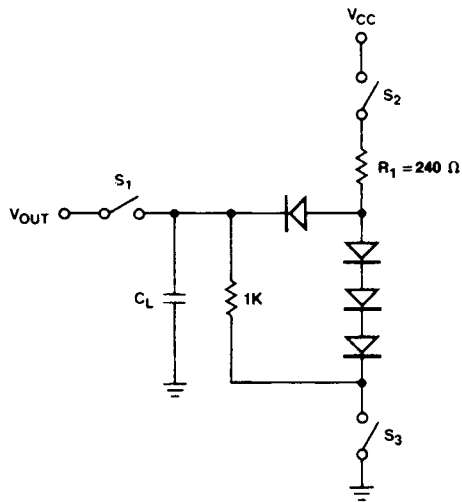
No.	Parameter	For	With Respect To	29C111	29C111-1	Unit
				Max. Value	Max. Value	
15	Data Setup	D <sub>15-0</sub>	CP ↑	18	16	ns
16	Data Hold	D <sub>15-0</sub>	CP ↑	0	0	ns
17	Address Setup	Y <sub>15-0</sub>	CP ↑	16	13	ns
18	Address Hold	Y <sub>15-0</sub>	CP ↑	0	0	ns
19	Instruction Setup	I <sub>5-0</sub>	CP ↑	11	16	ns
20	Instruction Hold	I <sub>5-0</sub>	CP ↑	0	0	ns
21	Forced Continue Setup	FC	CP ↑	18	15	ns
22	Forced Continue Hold	FC	CP ↑	0	0	ns
23	Test Setup	T <sub>11-0</sub>	CP ↑	19	16	ns
24	Test Hold	T <sub>11-0</sub>	CP ↑	0	0	ns
25	Select Setup	S <sub>3-0</sub>	CP ↑	21	17	ns
26	Select Hold	S <sub>3-0</sub>	CP ↑	0	0	ns
27	Reset Setup	RST	CP ↑	34	28	ns
28	Reset Hold	RST	CP ↑	0	0	ns
29	Interrupt Request Setup	INTR	CP ↑	29	23	ns
30	Interrupt Request Hold	INTR	CP ↑	0	0	ns
31	Interrupt Enable Setup	INTEN	CP ↑	28	23	ns
32	Interrupt Enable Hold	INTEN	CP ↑	0	0	ns
33	Hold Mode Setup	HOLD	CP ↑	19	17	ns
34	Hold Mode Hold	HOLD	CP ↑	0	0	ns
35	Carry-In Setup	C <sub>in</sub>	CP ↑	17	14	ns
36	Carry-In Hold	C <sub>in</sub>	CP ↑	0	0	ns

## D. MINIMUM CLOCK REQUIREMENT

No.	Description	29C111	29C111-1	Unit
		Max. Value	Max. Value	
47	Minimum Clock LOW Time	20	16	ns
48	Minimum Clock HIGH Time	13	12	ns

- Notes: 1. (INTR, INTEN)-to-EQUAL is the sum of (INTR, INTEN)-to-Y disable time and Y-to-EQUAL delay time. This is not tested due to bus turnaround in Master/Slave mode.
2. Z = Three-state output path; use Table B.
3. The maximum disable times are tested with load capacitance of  $C_L = 50$  pF.
4. The typical disable times with the load capacitance of  $C_L = 5$  pF are derived at  $V_{CC} = 5$  V under room temperature. These should be used as a reference.

## SWITCHING TEST CIRCUIT

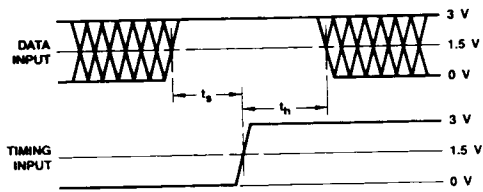


TC003420

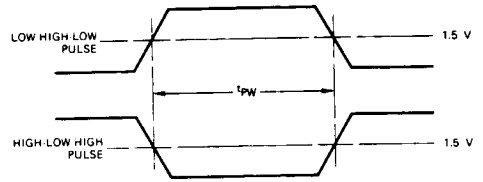
### A. Three-State Outputs

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

# SWITCHING TEST WAVEFORMS



WFR02970

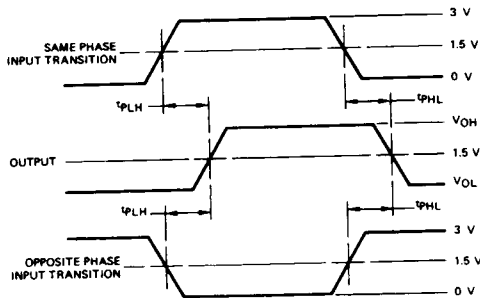


WFR02790

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

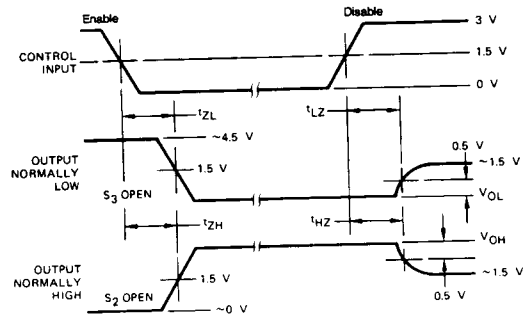
## Setup, Hold, and Release Times

## Pulse Width



WFR02980

## Propagation Delay



WFR02663

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

## Enable and Disable Times

## 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. Current level may vary from product to product.
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 which 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 which varies from one type of tester to another, but is generally around 50 pF. This makes it impossible to make direct measurements of parameters which 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 setup are used to predict the result at the lower capacitance.

Similarly, a product may be specified at more than one capacitive load. Since the typical automatic tester is not capable of switching loads in mid-test, it is impossible to make measurements at both capacitances even though they may both be greater than the stray capacitance. In

these cases, a measurement is made at one of the two capacitances. The result at the other capacitance is predicted from engineering correlations based on data taken with a bench setup and the knowledge that certain DC measurements ( $I_{OH}$ ,  $I_{OL}$ , for example) have already been taken and are within 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 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 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.

### 9. 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  $V_{OUT} = 0$ ,  $V_{CC} = \text{Max.}$  case.

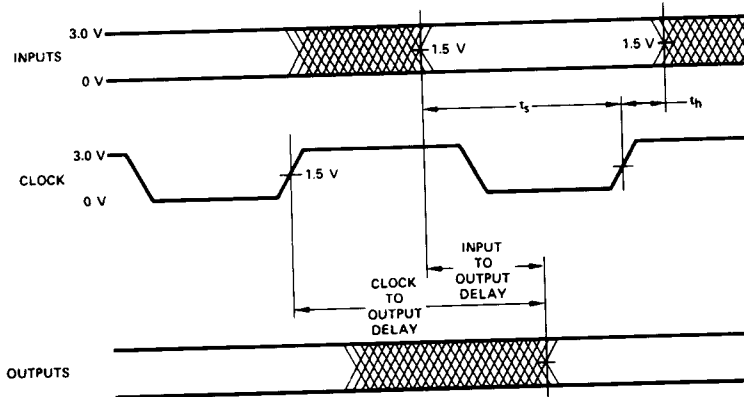
## SWITCHING WAVEFORMS

### KEY TO SWITCHING WAVEFORMS

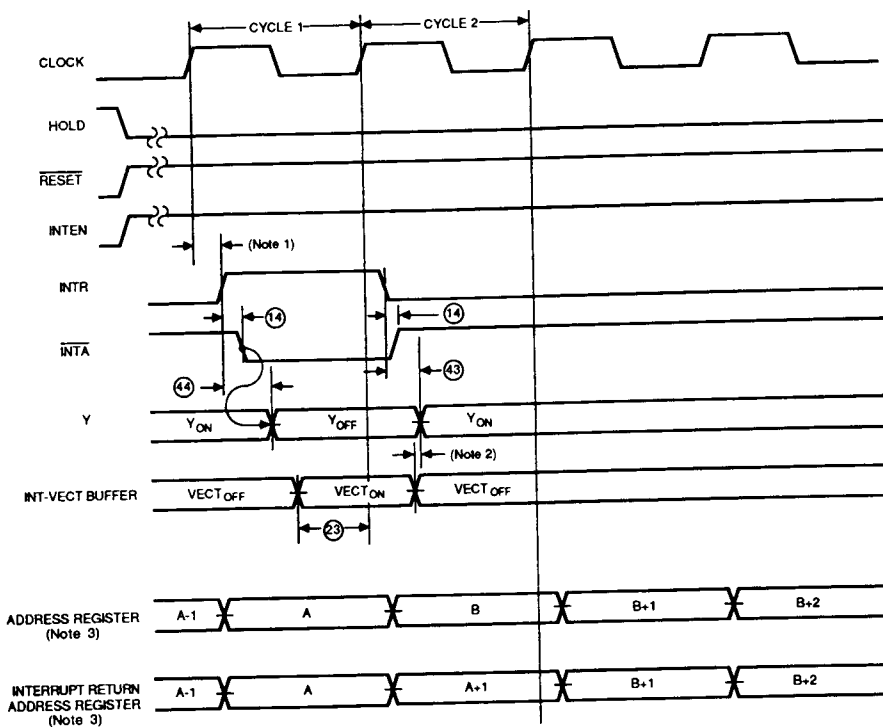
WAVEFORM	INPUTS	OUTPUTS
	MUST BE STEADY	WILL BE STEADY
	MAY CHANGE FROM H TO L	WILL BE CHANGING FROM H TO L
	MAY CHANGE FROM L TO H	WILL BE CHANGING FROM L TO H
	DON'T CARE ANY CHANGE PERMITTED	CHANGING STATE UNKNOWN
	DOES NOT APPLY	CENTER LINE IS HIGH IMPEDANCE "OFF" STATE

KS000010

## SWITCHING WAVEFORMS (Cont'd.)



WFR02990

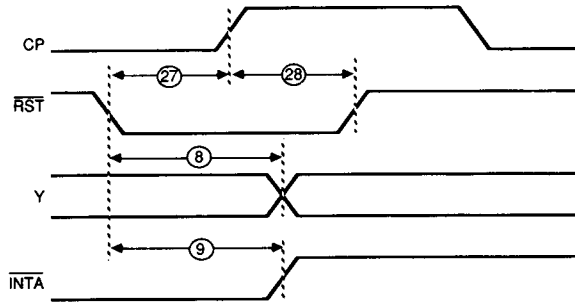


WF025100

### Interrupt Timing

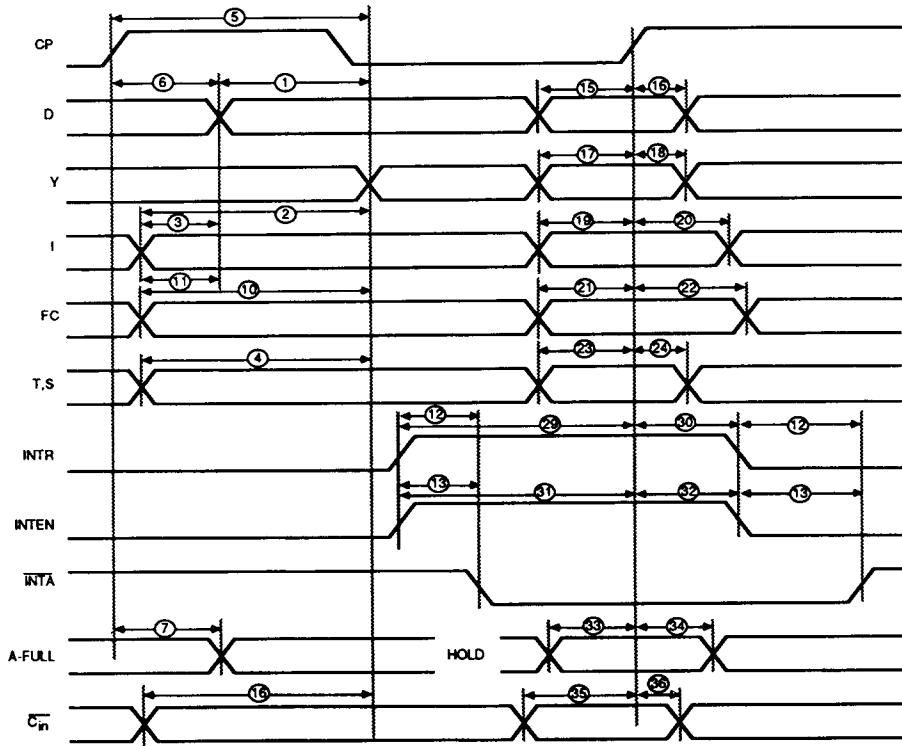
- Notes:
1. Interrupt Request comes from an interrupt-controller register. It reflects the CP 1 to INTR time of the interrupt controller.
  2. During Cycle 2, there may be contention on the Y-bus if the Y-bus is turned ON before the INT-VECT buffer is turned OFF.
  3. Refer to Figures 4 and 5 for definition of A and B.

# SWITCHING WAVEFORMS (Cont'd.)



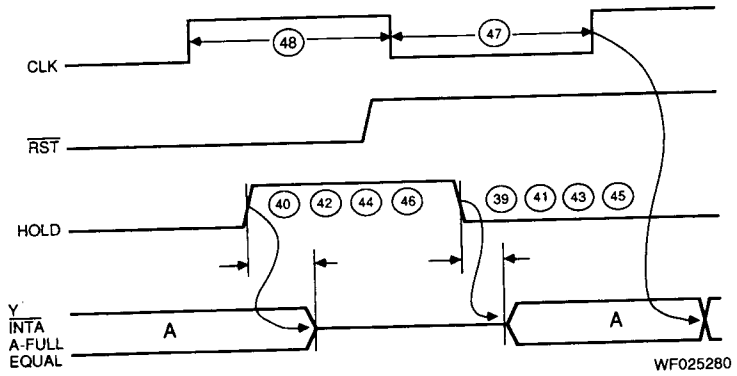
WF025270

## Reset Timing



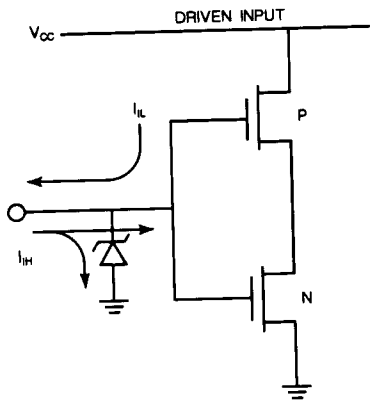
WF025440

### SWITCHING WAVEFORMS (Cont'd.)

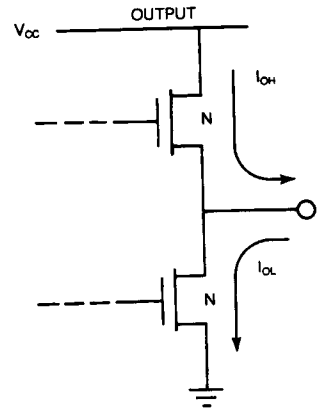


Hold Timing

### INPUT/OUTPUT CIRCUIT DIAGRAMS



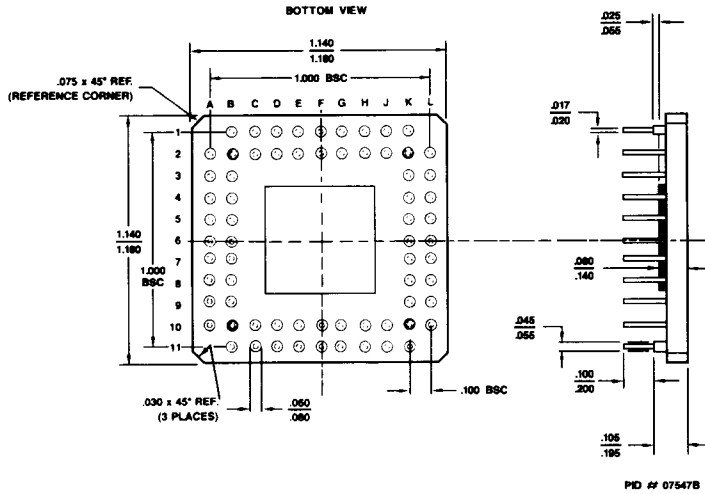
$C_I \approx 5.0$  pF, all inputs



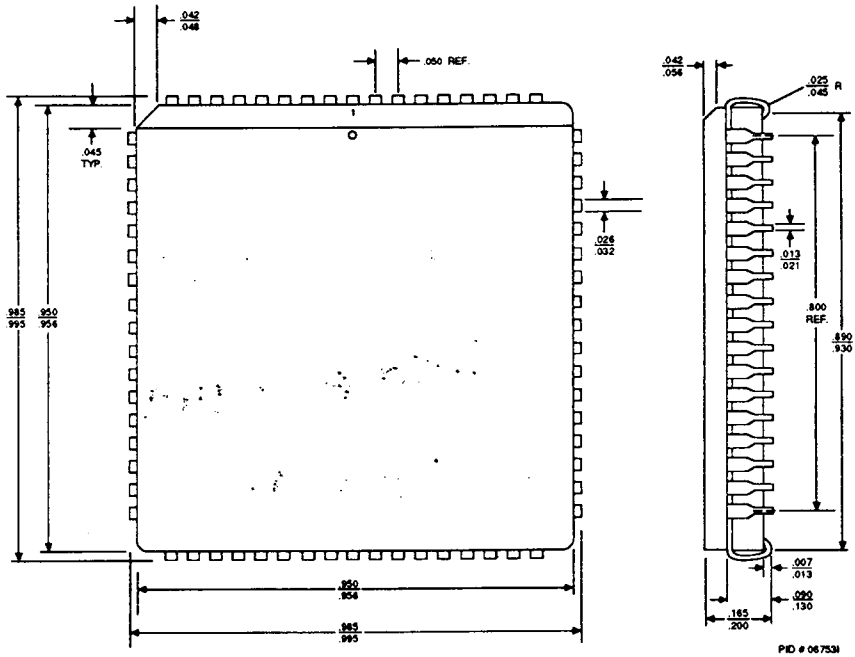
$C_O \approx 5.0$  pF, all outputs

PHYSICAL DIMENSIONS\*

CGX068



PL 068



\* For reference only.

008632

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.

**ADVANCED MICRO DEVICES** 901 Thompson Pl., P.O. Box 3453, Sunnyvale, CA 94088, USA  
 TEL: (408) 732-2400 • TWX: 910-339-9280 • TELEX: 34-6306 • TOLL FREE: (800) 538-8450

© 1988 Advanced Micro Devices, Inc.  
 Printed in U.S.A. AIS-WCP-12M-1/88-0