Am29C10A

CMOS Microprogram Controller

DISTINCTIVE CHARACTERISTICS

- **Low power**
  The CMOS Am29C10A is a plug-in replacement for the bipolar Am2910A. The Am29C10A dissipates 15% of the power of the equivalent bipolar part.
- **High-Speed CMOS**
  The Am29C10A is the bipolar-equivalent 20-MHz part; a speed-selected Am29C10A-1 runs at 25 MHz.
- **Twelve bits wide**
  Addresses up to 4096 words of microcode with one chip. All internal elements are a full 12 bits wide.
- **Internal loop counter**
  Pre-settable 12-bit down-counter for repeating instructions and counting loop iterations.
- **Four address sources**
  Microprogram address may be selected from microprogram counter, branch address bus, 9-level PUSH/POP stack, or internal holding register.
- **Sixteen powerful microinstructions**
  Executes 16 sequence control instructions, most of which are conditional on external condition input, state of internal loop counter, or both.
- **Output Enable controls three branch-address sources**
  Built-in decoder function to enable external devices onto branch address bus. Eliminates external decoder.

GENERAL DESCRIPTION

The Am29C10A Microprogram Controller is an address sequencer intended for controlling the sequence of execution of microinstructions stored in microprogram memory. Besides the capability of sequential access, it provides conditional branching to any microinstruction within its 4096-microword range. A last-in, first-out stack provides microsubroutine return linkage and looping capability; there are nine levels of nesting of microsubroutines. Microinstruction loop count control is provided with a count capacity of 4096.

During each microinstruction, the Microprogram Controller provides a 12-bit address from one of four sources: 1) the Microprogram Address Counter/Register (μPC), which usually contains an address one greater than the previous address; 2) an external (Direct) input (D); 3) a Register/counter (R) retaining data loaded during a previous microinstruction; or 4) a nine-deep last-in, first-out stack/File (F).

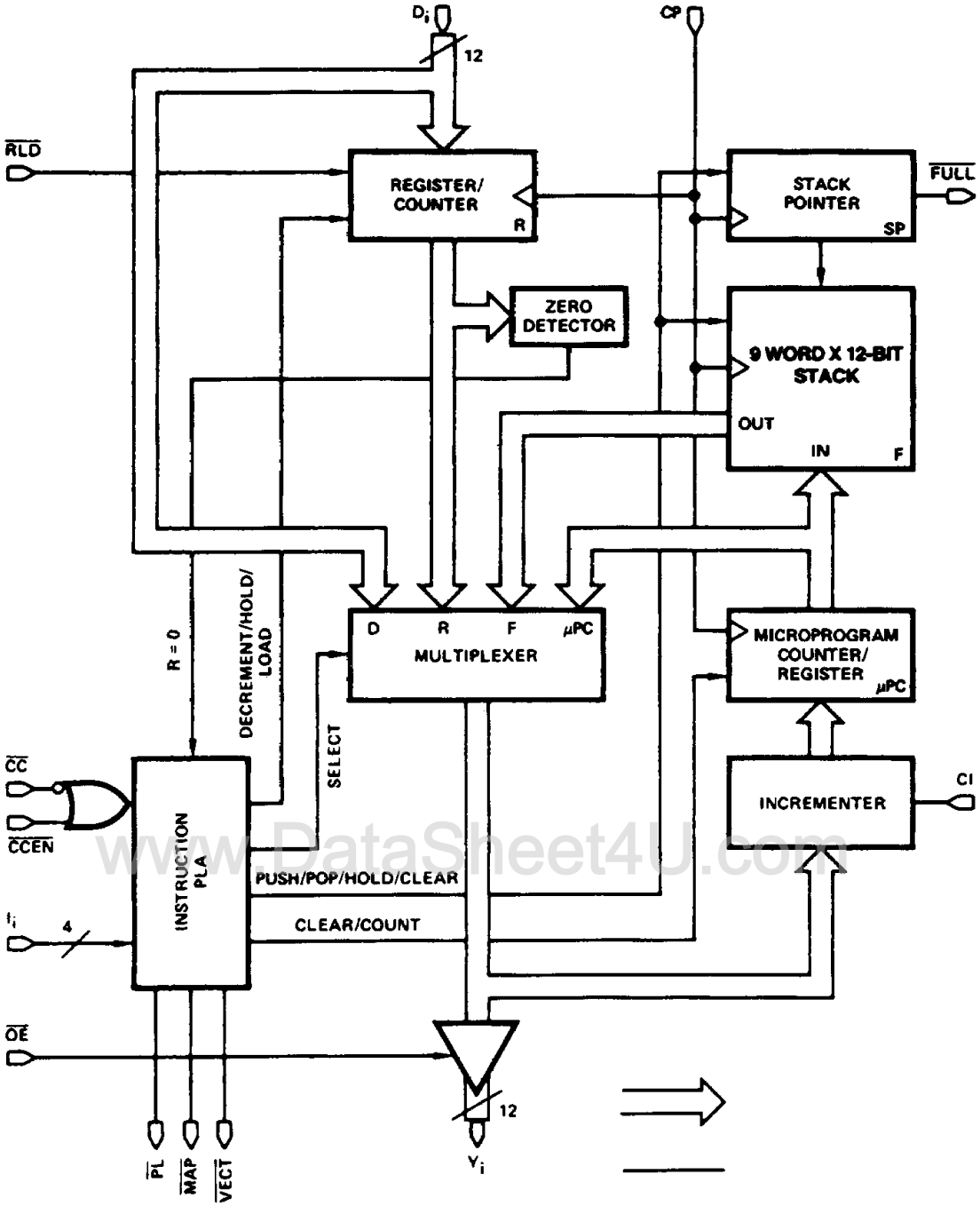
RELATED AMD PRODUCTS

Part No.	Description
Am27S35A	1024 x 8 Registered PROM
Am29C01	CMOS 4-Bit Microprocessor Slice
Am2902A	High-Speed Carry Look-Ahead Circuit
Am29C03	CMOS 4-Bit Super Slice
Am2904	Status and Shift Controller
Am29C101	16-Bit CMOS Microprocessor Slice
Am29C111	CMOS 16-Bit Microsequencer
Am29C116	CMOS 16-Bit Microprocessor
Am2914	Vectored Interrupt Controller
Am2918	Pipeline Register
Am2922	Condition Code MUX
Am2925	Clock Generator
Am29C331	CMOS 16-Bit Microprogram Sequencer
Am2940	DMA Address Generator
Am2952A	8-Bit Bidirectional I/O Port
Am29800A	High-Performance Bus Interface Family
Am29C800	High-Performance CMOS Bus Interface Family
Am29818A	SSR™ Diagnostics/Pipeline Register

SSR is a trademark of Advanced Micro Devices, Inc.

Publication # 06402 | Rev. C | Amendment /0
Issue Date: April 1987

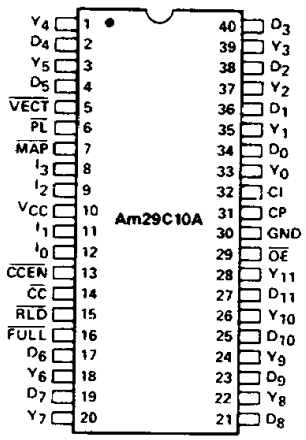
BLOCK DIAGRAM



BDR02321

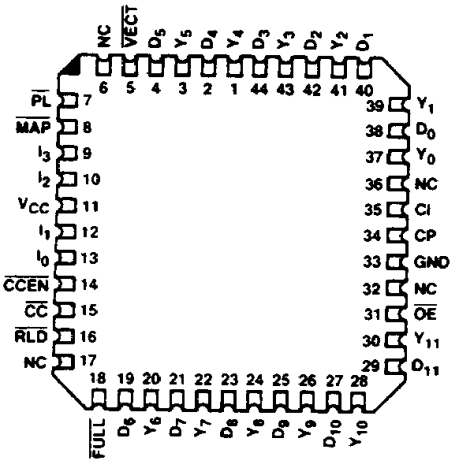
CONNECTION DIAGRAMS
Top View

DIPs



CD004671

LCC*

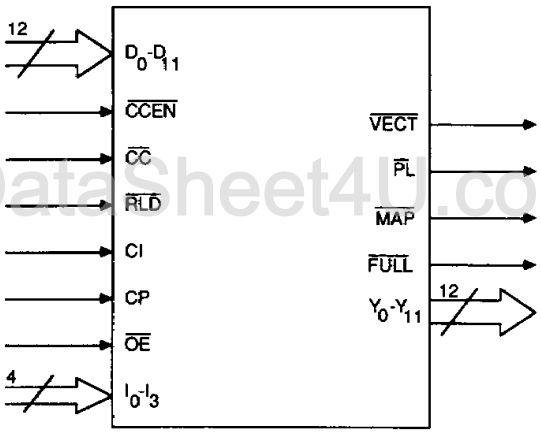


CD004690

*Also available in 44-Pin PLCC.
Pinouts are identical to LCC.

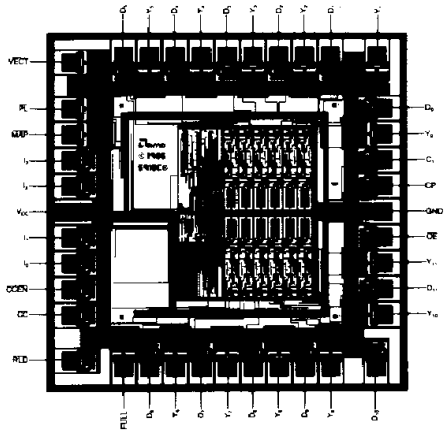
Note: Pin 1 is marked for orientation.

LOGIC SYMBOL



LS002940

METALLIZATION AND PAD LAYOUT

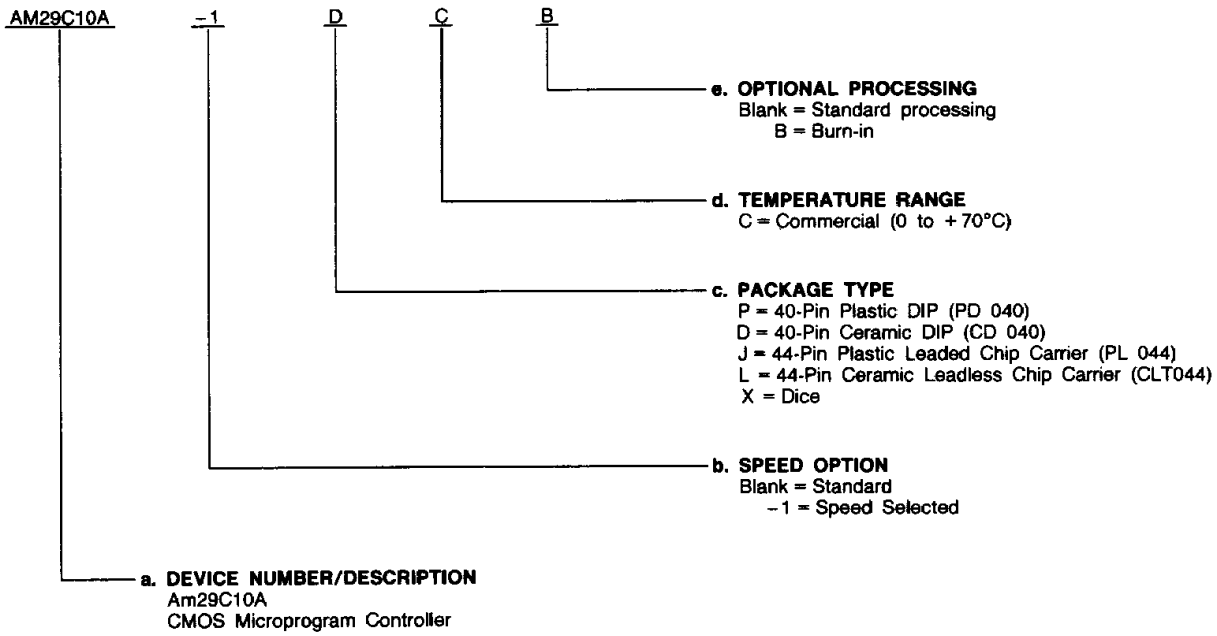


Die Size: 0.123" x 0.128"
Equivalent Gate Count: ≈850 Gates

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	
AM29C10A	PC, PCB, DC, DCB, JC, JCB, LC, XC
AM29C10A-1	PC, PCB, DC, DCB, JC, JCB

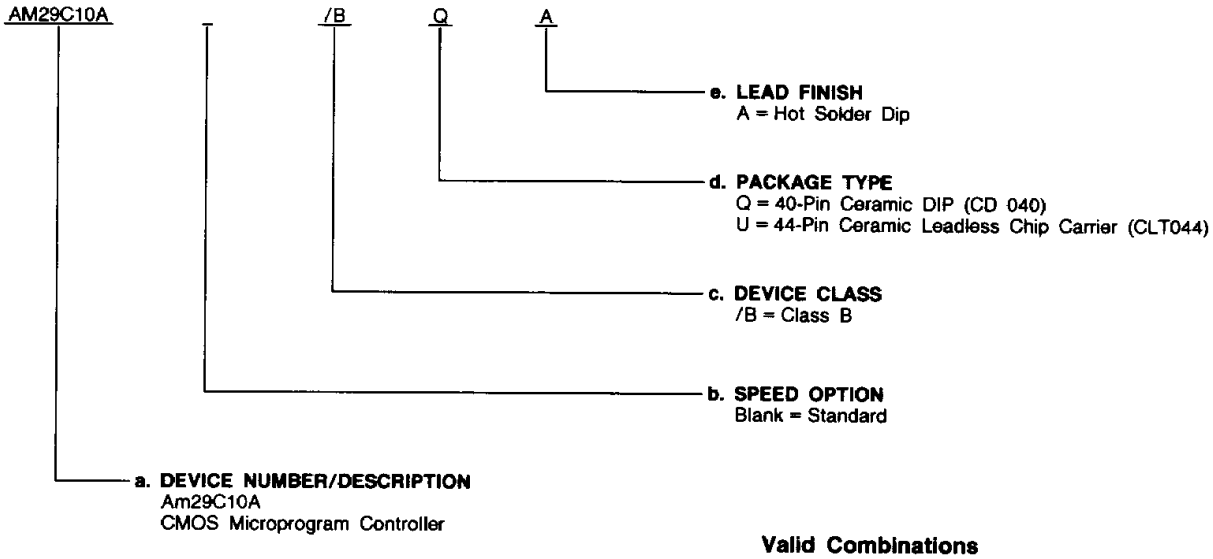
Valid Combinations

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

ORDERING INFORMATION (Cont'd.)

APL Products

AMD products for Aerospace and Defense applications are available in several packages and operating ranges. APL (Approved Products List) products are fully compliant with MIL-STD-883C requirements. The order number (Valid Combination) for APL products is formed by a combination of: **a. Device Number**
b. Speed Option (if applicable)
c. Device Class
d. Package Type
e. Lead Finish



Valid Combinations	
AM29C10A	/BQA, /BUA

Valid Combinations

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

Group A Tests

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

PIN DESCRIPTION

$\overline{CC}$ Condition Code (Input; Active LOW)
Used as test criterion. Pass test is a LOW on $\overline{CC}$.

$\overline{CCEN}$ Condition Code Enable (Input; Active LOW)
Whenever the signal is HIGH, $\overline{CC}$ is ignored and the part operates as though $\overline{CC}$ were true (LOW).

CI Carry In (Input)
Low-order carry input to incrementer for microprogram counter.

CP Clock Pulse (Input)
Triggers all internal state changes at LOW-to-HIGH edge.

D_I Direct Input (Input)
Direct input to register/counter and multiplexer. D_0 is LSB.

$\overline{FULL}$ Stack Full Flag (Output; Active LOW)
Indicates that nine items are on the stack.

I_i Instruction (Input)
Selects one of sixteen instructions for the Am29C10A.

$\overline{MAP}$ Mapping PROM Output Enable (Output; Active LOW)
Can select #2 source (usually Mapping PROM or PLA) as direct input source.

$\overline{OE}$ Output Enable (Input; Active LOW)
Three-state control of Y_i outputs.

$\overline{PL}$ Pipeline Register Output Enable (Output; Active LOW)
Can select #1 source (usually Pipeline Register) as direct input source.

$\overline{RLD}$ Register/Counter Load Enable (Input; Active LOW)
When LOW forces loading of register/counter regardless of instruction or condition.

$\overline{VECT}$ Vector Output Enable (Output; Active LOW)
Can select #3 source (for example, Interrupt Starting Address) as direct input source.

Y_i Address (Output)
Address to microprogram memory. Y_0 is LSB, Y_{11} is MSB.

FUNCTIONAL DESCRIPTION

The Am29C10A is a CMOS Microprogram Controller intended for use in power-sensitive microprocessor applications. It allows addressing of up to 4K words of microprogram.

The controller contains a four-input multiplexer that is used to select either the Register/counter (R), Direct input (D), Microprogram Address Counter/Register (μPC), or top of the stack/File (F) as the source of the next microinstruction address.

The "R" consists of twelve D-type, edge-triggered flip-flops, with a common clock enable. When its load control, $\overline{RLD}$, is LOW, new data is loaded on a positive clock transition. A few instructions include load. In most systems, these instructions will be sufficient, simplifying the microcode. The output of the "R" is available to the multiplexer as a source for the next microinstruction address. The D input furnishes a source of data for loading the "R."

The Am29C10A contains a μPC that is composed of a 12-bit incrementer followed by a 12-bit register. The μPC can be used in either of two ways: when the carry-in to the incrementer is HIGH, the microprogram register is loaded on the next clock cycle with the current Y output word plus one (Y + 1 → μPC). Sequential microinstructions are thus executed. When the carry-in is LOW, the incrementer passes the Y output word unmodified so that μPC is reloaded with the same Y word on the next clock cycle (Y → μPC). The same microinstruction is thus executed any number of times.

The third source for the multiplexer is the D input. This source is used for branching.

The fourth source available at the multiplexer input is a 9-word by 12-bit stack (file). The stack is used to provide return address linkage when executing microsubroutines or loops. The stack contains a built-in Stack Pointer (SP) which always points to the last file word written. This allows stack reference operations (looping) to be performed without a POP.

The "SP" operates as an up/down counter. During microinstructions 1, 4, and 5, the PUSH operation may occur. This causes the "SP" to increment and the file to be written with the required return linkage. On the cycle following the PUSH, the data is at the new location pointed to by the "SP" and is available to be read. However, it cannot be POPed during this cycle.

During five microinstructions, a POP operation may occur. The "SP" decrements at the next rising clock edge following a POP, effectively removing old information from the top of the stack.

At RESET (Instruction 0), the depth of stack nesting becomes zero. For each PUSH, the nesting depth increases by one; for each POP, the depth stack decreases by one. The depth can grow to nine. After a depth of nine is reached, $\overline{FULL}$ goes LOW. Any further PUSHes onto a full stack overwrite information at the top of the stack, but leave the "SP" unchanged. This operation will usually destroy useful information and is normally avoided. A POP from an empty stack may place nonmeaningful data on the Y outputs, but is otherwise safe. The "SP" remains at zero whenever a POP is attempted from a stack already empty.

The Register/counter is used during three microinstructions (8, 9, and 15) as a 12-bit down counter, with result = zero available as a microinstruction branch test criterion. This provides efficient iteration of microinstructions. The "R" is arranged such that if it is preloaded with a number N and then used as a loop termination counter, the sequence will be executed exactly N + 1 times. During instruction 15, a three-way branch under combined control of the loop counter and the condition code is available.

The device provides three-state Y outputs. These can be particularly useful in designs requiring automatic checkout of the processor. The Microprogram Controller outputs can be forced into the high-impedance state, and pre-programmed sequences of microinstructions can be executed via external access to the address lines.

Operation

Table 1 shows the result of each instruction in controlling the multiplexer which determines the Y outputs, and in controlling the three enable signals $\overline{PL}$, $\overline{MAP}$, and $\overline{VECT}$. The effect on the Register/counter (R) and the stack/File (F) after the next positive-going clock edge is also shown. The multiplexer determines which internal source drives the Y outputs. The value loaded into the Microprogram Address Counter/Register (μPC) is either identical to the Y output, or else one greater, as determined by CI. For each instruction, one and only one of the three outputs $\overline{PL}$, $\overline{MAP}$, and $\overline{VECT}$ is LOW. If these outputs control three-state enables for the primary source of microprogram jumps (usually part of a pipeline register), a PROM which maps the instruction to a microinstruction starting location, and an optional third source (often a vector from a DMA or interrupt source), respectively, the three-state sources can drive the D inputs without further logic.

Several inputs (see Pin Description), can modify instruction execution. The combination $\overline{CC}$ HIGH and $\overline{CCEN}$ LOW is used as a test in nine of the sixteen instructions. $\overline{RLD}$, when LOW, causes the D input to be loaded into the "R," overriding any HOLD or DEC operation specified in the instruction. $\overline{OE}$, normally LOW, may be forced HIGH to remove the Am29C10A Y outputs from a three-state bus.

The "F," a nine-word last-in, first-out 12-bit memory, has a pointer which addresses the value presently on the top of the stack. Explicit control of the Stack Pointer (SP) occurs during Instruction 0 (RESET), which makes the stack empty by resetting the "SP" to zero. After a RESET, and whenever else the stack is empty, the contents of the top of stack are undefined until a PUSH occurs. Any POPs performed while the stack is empty put undefined data on the "F" outputs and leave the "SP" at zero.

Any time the stack is full (nine more PUSHes than POPs have occurred since the stack was last empty), the $\overline{FULL}$ warning output occurs. This signal first appears on the microcycle after a ninth PUSH. No additional PUSH should be attempted onto a full stack; if tried, information within the stack will be overwritten and lost.

TABLE 1. INSTRUCTION SET

I_3-I_0	MNEMONIC	NAME	REG/ CNTR CONTENTS	FAIL $\overline{CCEN}$ = L and $\overline{CC}$ = H		PASS $\overline{CCEN}$ = H or $\overline{CC}$ = L		REG/ CNTR	ENABLE
				Y	STACK	Y	STACK		
0	JZ	JUMP ZERO	X	0	CLEAR	0	CLEAR	HOLD	PL
1	CJS	COND JSB PL	X	PC	HOLD	D	PUSH	HOLD	PL
2	JMAP	JUMP MAP	X	D	HOLD	D	HOLD	HOLD	MAP
3	CJP	COND JUMP PL	X	PC	HOLD	D	HOLD	HOLD	PL
4	PUSH	PUSH/COND LD CNTR	X	PC	PUSH	PC	PUSH	Note 1	PL
5	JSRP	COND JSB R/PL	X	R	PUSH	D	PUSH	HOLD	PL
6	CJV	COND JUMP VECTOR	X	PC	HOLD	D	HOLD	HOLD	VECT
7	JRP	COND JUMP R/PL	X	R	HOLD	D	HOLD	HOLD	PL
8	RFCT	REPEAT LOOP, CNTR ≠ 0	≠ 0	F	HOLD	F	HOLD	DEC	PL
			= 0	PC	POP	PC	POP	HOLD	PL
9	RPCT	REPEAT PL, CNTR ≠ 0	≠ 0	D	HOLD	D	HOLD	DEC	PL
			= 0	PC	HOLD	PC	HOLD	HOLD	PL
10	CRTN	COND RTN	X	PC	HOLD	F	POP	HOLD	PL
11	CJPP	COND JUMP PL & POP	X	PC	HOLD	D	POP	HOLD	PL
12	LDCT	LD CNTR & CONTINUE	X	PC	HOLD	PC	HOLD	LOAD	PL
13	LOOP	TEST END LOOP	X	F	HOLD	PC	POP	HOLD	PL
14	CONT	CONTINUE	X	PC	HOLD	PC	HOLD	HOLD	PL
15	TWB	THREE-WAY BRANCH	≠ 0	F	HOLD	PC	POP	DEC	PL
			= 0	D	POP	PC	POP	HOLD	PL

Note 1: If $\overline{CCEN}$ = L and $\overline{CC}$ = H, hold; else load. H=HIGH L=LOW X = Don't Care

The Am29C10A Instruction Set

The Am29C10A provides sixteen instructions that select the address of the next microinstruction to be executed. Four of the instructions are unconditional – their effect depends only on the instruction. Ten of the instructions have an effect that is partially controlled by external, data-dependent conditions. Three of the instructions have an effect that is partially controlled by the contents of the internal Register/counter (R). In this discussion it is assumed the C_i is tied HIGH.

In the ten conditional instructions, the result of the data-dependent test is applied to $\overline{CC}$. If the $\overline{CC}$ input is LOW, the test is considered to have been passed, and the action specified in the name occurs; otherwise, the test has failed and an alternate (often simply the execution of the next sequential microinstruction) occurs. Testing of $\overline{CC}$ may be disabled for a specific microinstruction by setting $\overline{CCEN}$ HIGH, which unconditionally forces the action specified in the name; that is, it forces a pass. Other ways of using $\overline{CCEN}$ include: 1) tying it HIGH, which is useful if no microinstruction is data-dependent; 2) tying it LOW if data-dependent instructions are never forced unconditionally; or 3) tying to the source of Am29C10A instruction bit I_0, which leaves Instructions 4, 6, and 10 as data-dependent, but makes others unconditional. All of these tricks save one bit of microcode width.

The effect of three instructions depends on the contents of the "R." Unless the counter holds a value of zero, it is decremented; if it does hold zero, it is held and a different microprogram next address is selected. These instructions are useful for executing a microinstruction loop a known number of times. Instruction 15 is affected both by the external condition code and the internal "R."

Perhaps the best technique for understanding the Am29C10A is to simply take each instruction and review its operation. In order to provide some feel for the actual execution of these instructions, examples of all sixteen instructions are included.

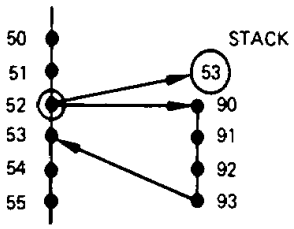
The examples given should be interpreted in the following manner: the intent is to show microprogram flow as various microprogram memory words are executed. For example, the CONTINUE instruction, Instruction 14, simply means that the contents of microprogram memory word 50 are executed, then the contents of word 51 are executed. This is followed by the contents of microprogram memory word 52 and the contents of microprogram memory word 53. The microprogram addresses used in the examples were arbitrarily chosen and have no meaning other than to show instruction flow. The exception to this is the first example, JUMP ZERO, which forces the microprogram location counter to address ZERO. Each dot refers to the time that the contents of the microprogram memory word is in the pipeline register. While no special symbology is used for the conditional instructions, the text to follow will explain what the conditional choices are in each example.

Instruction 0 – JUMP ZERO (JZ)

PFR00830

Instruction 0, JUMP-to-ZERO, or RESET, unconditionally specifies that the address of the next microinstruction is zero. Many designs use this feature for power-up sequences and provide the power-up firmware beginning at microprogram memory word location 0.

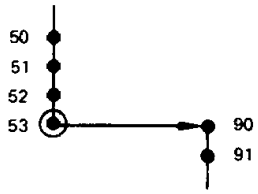
Instruction 1 – COND JSB PL (CJS)



PFR00950

Instruction 1 is a CONDITIONAL JUMP-TO-SUBROUTINE via the address provided in the PIPELINE register. As shown above, the machine might have executed words at address 50, 51, and 52. When the contents of address 52 are in the PIPELINE register, the next address control function is the CONDITIONAL JUMP-TO-SUBROUTINE. Here, if the test is passed, the next instruction executed will be the contents of microprogram memory location 90. If the test has failed, the JUMP-TO-SUBROUTINE will not be executed; the contents of microprogram memory location 53 will be executed instead. Thus, the CONDITIONAL JUMP-TO-SUBROUTINE instruction at location 52 will cause the instruction either in location 90 or in location 53 to be executed next. If the TEST input is such that location 90 is selected, value 53 will be PUSHed onto the internal stack. This provides the return linkage for the machine when the subroutine beginning at location 90 is completed. In this example, the subroutine was completed at location 93 and a RETURN-FROM-SUBROUTINE would be found at location 93. Location 90 cannot be a CRTN (Instruction 10).

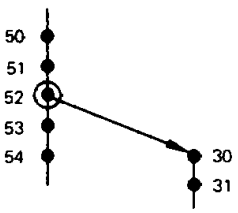
Instruction 2 – JUMP MAP (JMAP)



PFR00960

Instruction 2 is the JUMP MAP instruction. This is an unconditional instruction which causes the $\overline{MAP}$ output to be enabled so that the next microinstruction location is determined by the address supplied via the mapping PROMs. Normally, the JUMP MAP instruction is used at the end of the instruction fetch sequence for the machine. In the example above, microinstructions at locations 50, 51, 52 and 53 might have been the fetch sequence and at its completion at location 53, the JUMP MAP function would be contained in the PIPELINE register. This example shows the mapping PROM outputs to be 90; therefore, an unconditional jump to microprogram memory address 90 is performed.

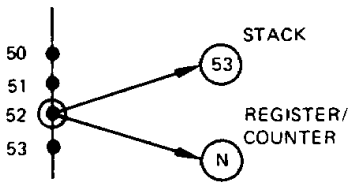
Instruction 3 – COND JUMP PL (CJP)



PFR00820

Instruction 3, CONDITIONAL JUMP PIPELINE, derives its branch address from the PIPELINE register branch address value ($D_{11} - D_0$). This instruction provides a technique for branching to various microprogram sequences depending upon the test condition inputs. Quite often, state machines are designed that simply execute tests on various inputs waiting for the condition to come true. When the true condition is reached, the machine then branches and executes a set of microinstructions to perform some function. This usually has the effect of resetting the input being tested until some point in the future. The example shows the CONDITIONAL JUMP via the PIPELINE register address at location 52. When the contents of microprogram memory word 52 are in the PIPELINE register, the next address will be either location 53 or location 30 in this example. If the test is passed, the value currently in the PIPELINE register (30) will be selected. If the test fails, the next address selected will be contained in the μPC, which in this example is 53.

Instruction 4 – PUSH/COND LD CNTR (PUSH)

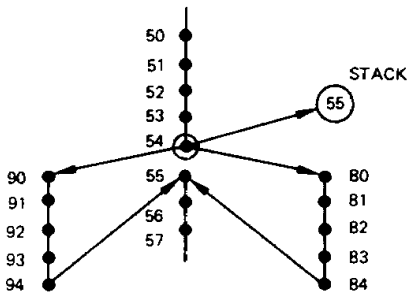


PFR00840

Instruction 4 is the PUSH/CONDITIONAL LOAD COUNTER instruction and is used primarily for setting up loops in microprogram firmware. In this example, when address 52 is in

the PIPELINE register, a PUSH will be made onto the stack and the counter will be loaded based on the condition. When a PUSH occurs, the value PUSHed is always the next sequential instruction address. In this case, the address is 53. If the test fails, the counter is not loaded; if it is passed, the counter is loaded with the value contained in the PIPELINE register branch address field. Thus, a single microinstruction can be used to set up a loop to be executed a specific number of times. Instruction 8 will describe how to use the PUSHed value and the "R" for looping.

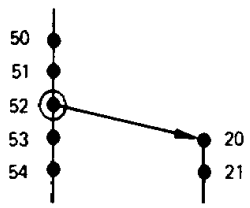
Instruction 5 – COND JSB R/PL (JSRP)



PFR00940

Instruction 5 is a CONDITIONAL JUMP-TO-SUBROUTINE via the REGISTER/counter (R) of the contents of the PIPELINE register. A PUSH is always performed and one of two subroutines executed. In this example, either the subroutine beginning at address 80 or the subroutine beginning at address 90 will be performed. A RETURN-FROM-SUBROUTINE (Instruction 10) returns the microprogram flow to address 55. In order for this microinstruction control sequence to operate correctly, both the next address fields of location 53 and the next address fields of location 54 would have to contain the proper value. For this example, assume that the branch address fields of location 53 contain the value 90, so that it will be in the Am29C10A "R" when the contents of address 54 are in the PIPELINE register. This requires that the instruction at address 53 load the "R." If the test failed during the execution of Instruction 5 (at address 54), the contents of the register (value = 90) will select the address of the next microinstruction. If the test input passes, the PIPELINE register contents (value = 80) will determine the address of the next microinstruction. Therefore, this instruction provides the ability to select one of two subroutines to be executed based on a test condition. Neither the instruction at address 80 nor address 90 can be CRTN (Instruction 10).

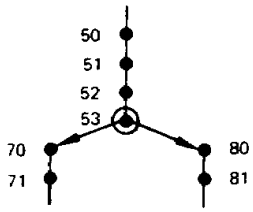
Instruction 6 – COND JUMP VECTOR (CJV)



PFR00920

Instruction 6 is a CONDITIONAL JUMP VECTOR instruction that provides the capability to take the branch address from a third source previously not discussed. In order for this instruction to be useful, the Am29C10A output $\overline{VECT}$ is used to control a three-state control input of a register, buffer, or PROM containing the next microprogram address. This instruction provides one technique for performing interrupt type branching at the microprogram level. Since this instruction is conditional, a pass causes the next address to be taken from the vector source, while failure causes the next address to be taken from the Microprogram Counter (μPC). In the example, if the CONDITIONAL JUMP VECTOR instruction is contained at location 52, execution will continue at vector address 20 if the $\overline{CC}$ input is LOW, and the microinstruction at address 53 will be executed if the $\overline{CC}$ input is HIGH.

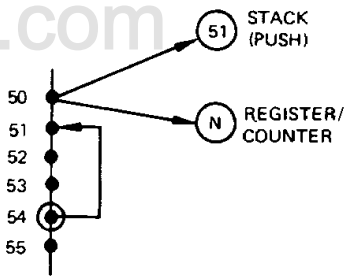
Instruction 7 – COND JUMP R/PL (JRP)



PFR00930

Instruction 7 is a CONDITIONAL JUMP via the contents of the Am29C10A REGISTER/counter or the contents of the PIPELINE register. This instruction is very similar to Instruction 5, the CONDITIONAL JUMP-TO-SUBROUTINE via "R" or PIPELINE. The major difference between Instruction 5 and Instruction 7 is that no PUSH onto the stack is performed with 7. The example depicts this instruction as a branch to one of two locations depending on the test condition. The example assumes the PIPELINE register contains the value 70 when the contents of address 52 is being executed. As the contents of address 53 are clocked into the PIPELINE register, the value 70 is loaded into the "R" in the Am29C10A. The value 80 is available when the contents of address 53 are in the PIPELINE register. Thus, control is transferred to either address 70 or address 80, depending on the test condition.

Instruction 8 – REPEAT LOOP, CNTR ≠ 0 (RFCT)



PFR00910

Instruction 8 is the REPEAT LOOP, COUNTER ≠ ZERO instruction. This microinstruction makes use of the decrementing capability of the "R." To be useful, some previous instruction, such as 4, must have loaded a count value into the "R." This instruction checks to see whether the "R" contains a non-zero value. If so, the "R" is decremented, and the address of the next microinstruction is taken from the top of the stack. If the "R" contains zero, the loop exit condition is occurring; control falls through to the next sequential microinstruction by selecting μPC; the stack/File (F) is POPed by decrementing the Stack Pointer (SP), but the contents of the top of the stack are thrown away.

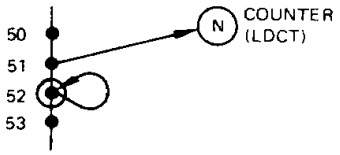
In this example, location 50 most likely would contain a PUSH/CONDITIONAL LOAD COUNTER instruction which would have caused address 51 to be PUSHed on the stack and the counter to be loaded with the proper value for looping the desired number of times.

In this example, since the loop test is made at the end of the instructions to be repeated (address 54), the proper value to

be loaded by the instructions at address 50 is one less than the desired number of passes through the loop. This method allows a loop to be executed 1 to 4096 times. If it is desired to execute the loop from 0 to 4095 times, the firmware should be written to make the loop exit test immediately after loop entry.

Single-microinstruction loops provide a highly efficient capability for executing a specific microinstruction a fixed number of times. Examples include fixed rotates, byte swap, fixed point multiply, and fixed point divide.

Instruction 9 – REPEAT PL, CNTR ≠ 0 (RPCT)

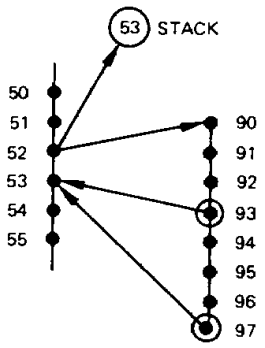


PFR00890

Instruction 9 is the REPEAT PIPELINE register, COUNTER ≠ ZERO instruction. This instruction is similar to Instruction 8 except that the branch address now comes from the PIPELINE register rather than the file. In some cases, this instruction may be thought of as a one-word file extension. That is, by using this instruction, a loop with the counter can still be performed when subroutines are nested nine deep. This instruction's operation is very similar to that of Instruction 8. The differences are that on this instruction, a failed test condition causes the source of the next microinstruction address to be the D inputs. When the test condition is passed, this instruction does not perform a POP because the stack is not being used.

In this example, the REPEAT PIPELINE register, COUNTER ≠ ZERO instruction is location 52, and is shown as a single microinstruction loop. The address in the PIPELINE register would be 52. Location 51 in this example could be the LOAD COUNTER AND CONTINUE instruction (number 12). While the example shows a single microinstruction loop by simply changing the address in a PIPELINE register, multi-instruction loops can be performed in this manner for a fixed number of times as determined by the counter.

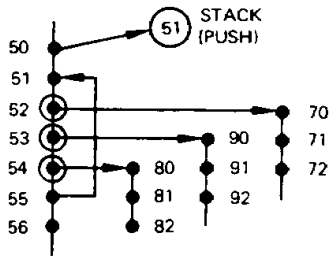
Instruction 10 – COND RETURN (CRTN)



PFR00900

Instruction 10 is the CONDITIONAL RETURN-FROM-SUBROUTINE instruction. As the name implies, this instruction is used to branch from the subroutine back to the next microinstruction address following the subroutine call. Since this instruction is conditional, the return is performed only if the test is passed. If the test is failed, the next sequential microinstruction is performed. This example depicts the use of the CONDITIONAL RETURN-FROM-SUBROUTINE instruction in both the conditional and the unconditional modes. This example first shows a JUMP-TO-SUBROUTINE at address 52 where control is transferred to address 90. At address 93, a CONDITIONAL RETURN-FROM-SUBROUTINE instruction is performed. If the test is passed, the stack is accessed and the program will transfer to the next instruction at address 53. If the test is failed, the next microinstruction at address 94 will be executed. The program will continue to address 97 where the subroutine is complete. To perform an unconditional RETURN-FROM-SUBROUTINE, the CONDITIONAL RETURN-FROM-SUBROUTINE instruction is executed unconditionally by programming the microinstruction at address 97 to force $\overline{\text{CCEN}}$ HIGH, disabling the test and the forced PASS to cause an unconditional return.

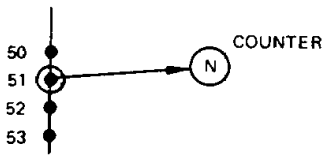
Instruction 11 – COND JUMP PL & POP (CJPP)



PFR00850

Instruction 11 is the CONDITIONAL JUMP PIPELINE register address and POP stack instruction. This instruction provides another technique for loop termination and stack maintenance. The example shows a loop being performed from address 55 back to address 51. The instructions at locations 52, 53, and 54 are all CONDITIONAL JUMP PIPELINE and POP instructions. At address 52, if the $\overline{\text{CC}}$ input is LOW, a branch will be made to address 70 and the stack will be properly maintained via a POP. Should the test fail, the instruction at location 53 (the next sequential instruction) will be executed. Likewise, at address 53, either the instruction at 90 or 54 will be subsequently executed, respective to the test being passed or failed. The instruction at 54 follows the same rules, going to either 80 or 55. An instruction sequence as described here (using the CONDITIONAL JUMP PIPELINE and POP instruction) is very useful when several inputs are being tested and the microprogram is looping, waiting for any of the inputs being tested to occur, before proceeding to another sequence of instructions. This provides the powerful jump-table programming technique at the firmware level.

Instruction 12 – LD CNTR & CONTINUE (LDCT)

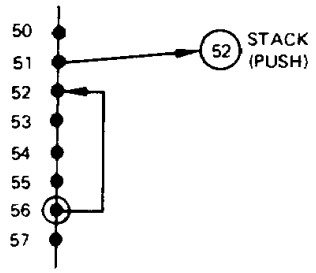


PFR00870

Instruction 12 is the LOAD COUNTER and CONTINUE instruction, which simply enables the counter to be loaded with the value at its parallel inputs. These inputs are normally connected to the PIPELINE branch address field which (in the architecture being described here) serves to supply either a branch address or a counter value depending upon the microinstruction being executed. Altogether there are three ways of loading the counter: 1) the explicit load by Instruction 12; 2) the conditional load included as part of Instruction 4; and 3) the use of $\overline{\text{RLD}}$ input along with any instruction. The use

of $\overline{RLD}$ with any instruction overrides any counting or decrementation specified in the instruction, calling for a load instead. Its use provides additional microinstruction power, at the expense of one bit of microinstruction width. Instruction 12 is exactly equivalent to the combination of Instruction 14 and $\overline{RLD}$ LOW. Its purpose is to provide a simple capability to load the Register/counter (R) in those implementations that do not provide microprogrammed control for $\overline{RLD}$.

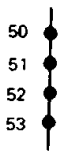
Instruction 13 – TEST END LOOP (LOOP)



PFR00860

Instruction 13 is the TEST END-OF-LOOP instruction which provides the capability of conditionally exiting a loop at the bottom. This is a conditional instruction that will cause the microprogram to loop, via the file if the test is failed, or to continue to the next sequential instruction. The example shows the TEST END-OF-LOOP microinstruction at address 56. If the test fails, the microprogram will branch to address 52. Address 52 is on the stack because a PUSH instruction had been executed at address 51. If the test is passed at address 56, the loop is terminated and the next sequential microinstruction at address 57 is executed, which also causes the stack to be POPed and thus accomplishes the required stack maintenance.

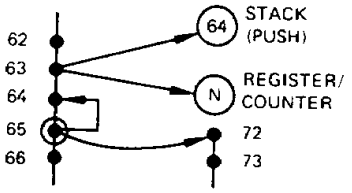
Instruction 14 – CONTINUE (CONT)



PFR00880

Instruction 14 is the CONTINUE instruction, which simply causes the Microprogram Address Counter/Register (μPC) to increment so that the next sequential microinstruction is executed. This is the simplest microinstruction of all, and it should be the default instruction the firmware requests whenever there is nothing else to do.

Instruction 15 – THREE-WAY BRANCH (TWB)



PFR00810

Instruction 15, THREE-WAY BRANCH, is the most complex. It provides for testing of both a data-dependent condition and the counter during one microinstruction, and for selecting among one of three microinstruction addresses as the next microinstruction to be performed. Like Instruction 8, a previous instruction will have loaded a count into the "R" while PUSHing a microbranch address onto the stack. Instruction 15 performs a decrement-and-branch-until-zero function similar to Instruction 8. The next address is taken from the top of the stack until the count reaches zero; the next address then comes from the PIPELINE register. The above action continues as long as the test condition fails. If at any execution of Instruction 15 the test condition is passed, no branch is taken. The μPC register furnishes the next address. When the loop is ended, either by the count becoming zero, or by passing the conditional test, the stack is POPed by decrementing the "SP," since interest in the value contained at the top of the stack is then complete.

The application of Instruction 15 can enhance the performance of a variety of machine-level instructions. Examples are: 1) a memory search instruction to be terminated either by finding a desired memory content or by reaching the search limit; 2) variable-field-length arithmetic terminated early upon finding that the content of the portion of the field still unprocessed is all zeroes; 3) key search in a disk controller processing variable-length records; and 4) normalization of a floating point number.

In the illustration above, a memory search, the instruction at microprogram address 63 can be Instruction 4 (PUSH), which will PUSH the value 64 onto the microprogram stack and load the number N, which is one less than the number of memory locations to be searched before giving up. Location 64 contains a microinstruction that fetches the next operand from the memory area to be searched and compares it with the search key. Location 65 contains a microinstruction that tests the result of the comparison, and is also a THREE-WAY BRANCH for microprogram control. If no match is found, the test fails and the microprogram goes back to location 64 for the next operand address. When the count becomes zero, the microprogram branches to location 72, which does whatever is necessary if no match is found. If a match occurs on any execution of the THREE-WAY BRANCH at location 65, control falls through to location 66 which handles this case. Whether the instruction ends by finding a match or not, the stack will have been POPed once, removing the value 64 from the top of the stack.

APPLICATIONS

Architectures Using The Am29C10A

A One-Level Pipeline provides better speed than most other architectures. The Microprogram Memory and the Am29C101 array are in parallel speed paths instead of in series. This is the recommended architecture for Am2900 designs. See Figure 1 (shading shows path(s) that usually limit(s) speed).

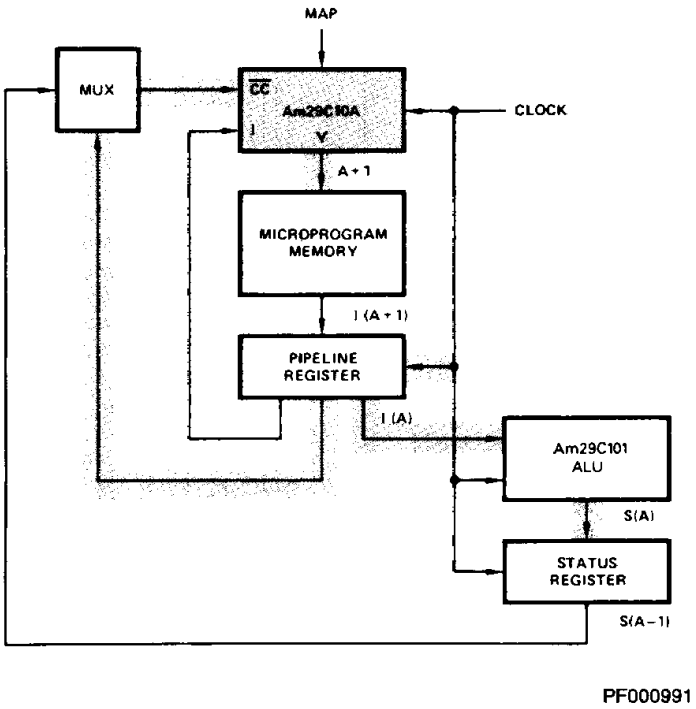


Figure 1. One-Level Pipeline-Based Architecture

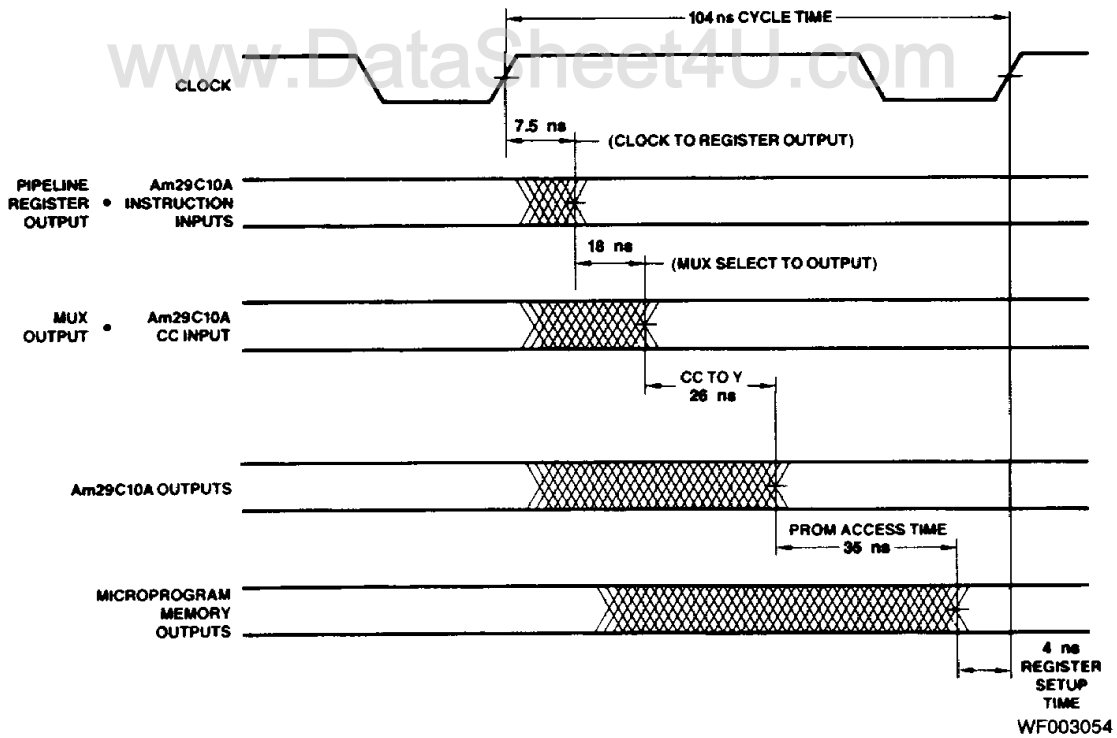
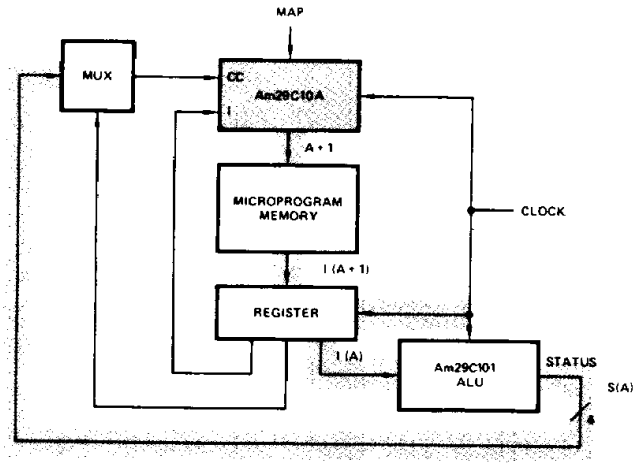


Figure 2. Typical CCU Cycle Timing Waveforms for One-Level Pipeline Architecture

This drawing shows the timing relationships in the CCU illustrated above.

Other Architectures Using The Am29C10A

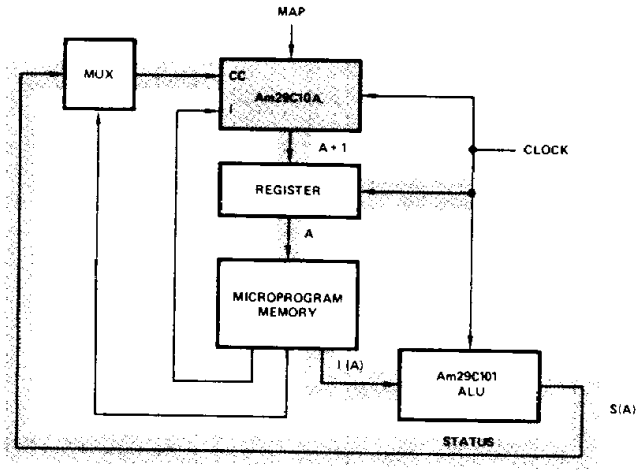
A Register at the Microprogram Memory output contains the microinstruction being executed. The Microprogram Memory and Am29C101 delay are in series. Conditional branches are executed on the same cycle as the ALU operation generating the condition (Figure 3*).



PF001002

Figure 3. Instruction-Based Architecture

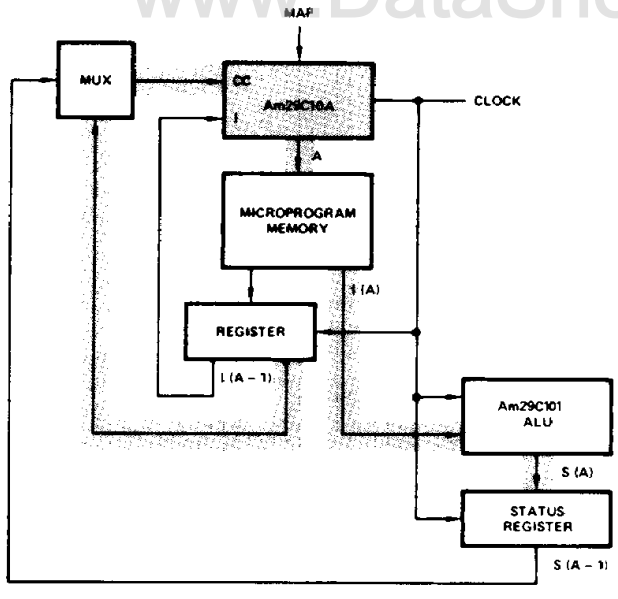
The Register at the Am29C10A output contains the address of the microinstruction being executed. The Microprogram Memory and Am29C101 are in series in the critical path. This architecture provides about the same speed as the Instruction-based architecture, but requires fewer register bits since only the address (typically 10 – 12 bits) is stored instead of the instruction (typically 40 – 60 bits) (Figure 4*).



PF001012

Figure 4. Address-Based Architecture

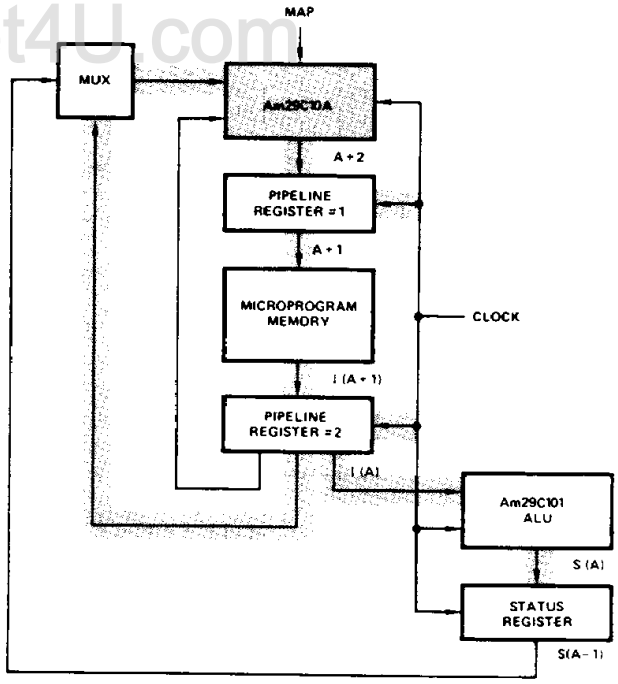
The Status Register provides the conditional branch control based on results of previous ALU cycle. The Microprogram Memory and Am29C101 are in series in the critical paths (Figure 5*).



PF001022

Figure 5. Data-Based Architecture

A Two-Level Pipeline provides the highest possible speed. It is more difficult to program because the selection of a microinstruction occurs two instructions ahead of its execution (Figure 6*).



PF001052

Figure 6. Two-Level Pipeline-Based Architecture

*Shading shows path(s) that usually limit(s) speed.

Am29C10A High-Speed Application

Optimal Am29C10A configurations can support high-speed bit-slice designs. When used with high-speed registers and PROMs, the Am29C10A can execute simple instructions in sub-100 ns.

Figure 7 illustrates the usual critical path in the sequencer.

Timing on the critical paths becomes:

Device	Path	Delay
Status Register	Clock →Output	7.5 ns
Fast MUX	Select →Output	18 ns
Am29C10A-1	$\overline{CC}$ →Y	26 ns
Fast PROM*	Addr →Output	35 ns
Pipeline Register*	Setup	4 ns
	Total	90.5 ns

*A registered PROM, such as the Am27S35A, can be used, which further reduces the delay in the critical path as well as reducing board space.

The following gives a suggested parts configuration to meet this design criterion:

Am29825A	Status Register
74S151	MUX (8 to 1)
Am27S33A	PROM
Am2918	Pipeline Register

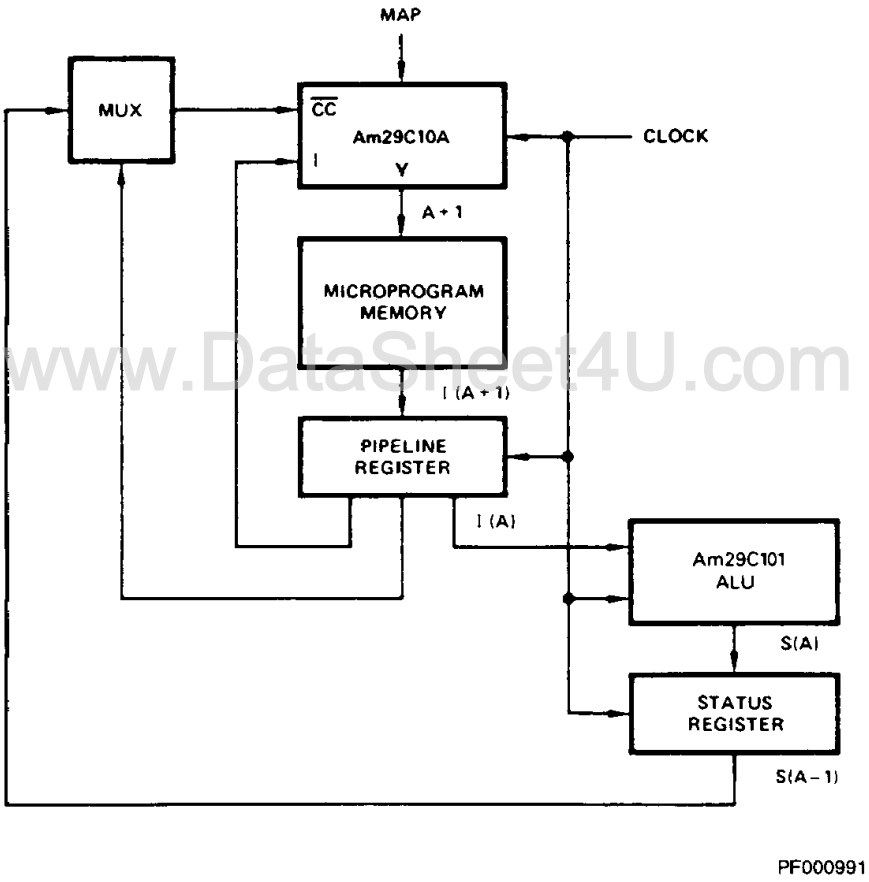


Figure 7. One-Level Pipeline-Based Architecture (Recommended)

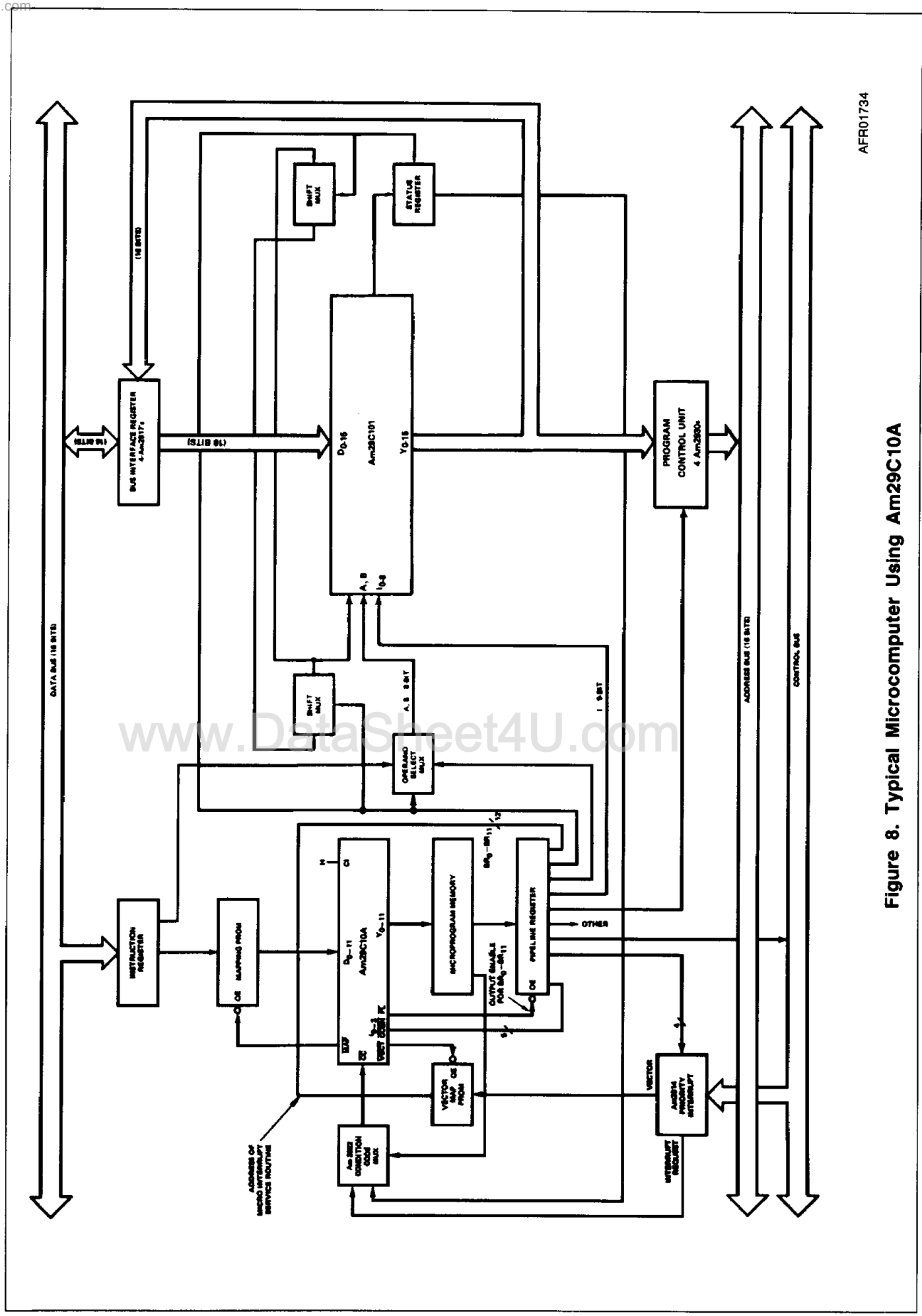


Figure 8. Typical Microcomputer Using Am29C10A

ABSOLUTE MAXIMUM RATINGS

Storage Temperature ..... −65°C to +150°C
Ambient Temperature with
Power Applied ..... −55°C to +125°C
Supply Voltage to Ground Potential
Continuous ..... −0.5 V to +7.0 V
DC Voltage Applied to Outputs For
High Output State ..... −0.3 V to +V_{CC} +0.3 V
DC Input Voltage ..... −0.3 V to +V_{CC} +0.3 V
DC Output Current, Into Low Outputs
(Except Bus) ..... 30 mA
DC Input Current ..... −10 to +10 mA

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

OPERATING RANGES

Commercial (C) Devices
Ambient Temperature (T_A) ..... 0°C to +70°C
Supply Voltage (V_{CC}) ..... +4.5 V to +5.5 V

Military (M) Devices*
Case Temperature (T_A) ..... −55°C to +125°C
Supply Voltage (V_{CC}) ..... +4.5 V to +5.5 V

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

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

DC CHARACTERISTICS over operating ranges unless otherwise specified; Included in Group A, Subgroup 1, 2, 3 tests unless otherwise noted.

Parameter Symbol	Parameter Description	Test Conditions (Note 1)			Min.	Max.	Units
V_{OH}	Output HIGH Voltage	V_{CC} = Min., I_{OH} = −1.6 mA, V_{IN} = V_{IH} or V_{IL}			2.4		V
V_{OL}	Output LOW Voltage	V_{CC} = Min., V_{IN} = V_{IH} or V_{IL}	Y_{0-11}, I_{OL} = 12 mA			0.5	V
			$\overline{PL}$, $\overline{VECT}$, $\overline{MAP}$, $\overline{FULL}$, I_{OL} = 8 mA				
V_{IH}	Input HIGH Level (Note 2)	Guaranteed input logical HIGH voltage for all inputs			2.0		V
V_{IL}	Input LOW Level (Note 2)	Guaranteed input logical LOW voltage for all inputs				0.8	
I_{IL}	Input LOW Current	V_{CC} = Max., V_{IN} = 0.5 V				−10	μA
I_{IH}	Input HIGH Current	V_{CC} = Max., V_{IN} = V_{CC} − 0.5 V				10	μA
I_{OZL}	Output OFF Current	V_{CC} = Max., $\overline{OE}$ = 2.4 V		V_{OUT} = 0.5 V		−10	μA
I_{OZH}				V_{OUT} = 2.4 V		10	
I_{CC} (Note 3)	Power Supply Current	V_{CC} = Max.	COM'L	T_A = 0 to +70°C		42	mA
			MIL	T_A = −55 to +125°C		50	
C_{PD}	Power Dissipation Capacitance (Note 4)	V_{CC} = 5.0 V, T_A = 25°C, No Load			400 pF Typical		

Notes:
1. For conditions shown as Min. or Max., use the appropriate value specified under Operating Ranges for the applicable device type.
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. C_{PD} determines the no-load dynamic current consumption:
I_{CC} (Total) = I_{CC} (Static) + C_{PD} V_{CC} f, where f is the switching frequency of the majority of the internal nodes, normally one-half of the clock frequency.

SWITCHING CHARACTERISTICS over Commercial Operating Range unless otherwise specified.

Am29C10A

The following tables specify the guaranteed performance of the Am29C10A over the commercial operating range of 0°C to +70°C, with V_{CC} from 4.5 to 5.5 V. All data are in ns, with measurements made at 1.5 V. All outputs have maximum DC load. C_L = 50 pF

A. Setup and Hold Times

Input	t_s	t_h
$D_i \rightarrow R$	16	0
$D_i \rightarrow PC$	30	0
I_0-I_3	35	0
$\overline{CC}$	24	0
$\overline{CCEN}$	24	0
CI	18	0
$\overline{RLD}$	19	0

B. Combinational Delays

Input	Y	$\overline{PL}$, $\overline{VECT}$, $\overline{MAP}$	$\overline{FULL}$
D_0-D_{11}	20	–	–
I_0-I_3	35	30	–
$\overline{CC}$	30	–	–
$\overline{CCEN}$	30	–	–
CP	40	–	31
$\overline{OE}$ (Note 1)	25/27	–	–

C. Clock Requirements

Minimum Clock LOW Time	20	ns
Minimum Clock HIGH Time	20	ns
Minimum Clock Period	50	ns

20MHz

Am29C10A-1

The following tables below specify the guaranteed performance of the Am29C10A-1 over the commercial operating range of 0° to +70°C, with V_{CC} from 4.5 to 5.5 V. All data are in ns, with measurements made at 1.5 V. All outputs have maximum DC load. C_L = 50 pF

A. Setup and Hold Times

Input	t_s	t_h
$D_i \rightarrow R$	13	0
$D_i \rightarrow PC$	24	0
I_0-I_3	29	0
$\overline{CC}$	24	0
$\overline{CCEN}$	21	0
CI	15	0
$\overline{RLD}$	15	0

B. Combinational Delays

Input	Y	$\overline{PL}$, $\overline{VECT}$, $\overline{MAP}$	$\overline{FULL}$
D_0-D_{11}	18	–	–
I_0-I_3	33	20	–
$\overline{CC}$	26	–	–
$\overline{CCEN}$	27	–	–
CP	33	–	25
$\overline{OE}$ (Note 1)	23/21	–	–

C. Clock Requirements

Minimum Clock LOW Time	20	ns
Minimum Clock HIGH Time	20	ns
Minimum Clock Period	40	ns

25MHz

Note 1. Enable/Disable. Disable times measured to 0.5 V change on output voltage level with C_L = 5.0 pF.

SWITCHING CHARACTERISTICS over Military Operating Range unless otherwise specified (for APL Products, Group A, Subgroups 9, 10, 11 are tested unless otherwise noted).

Am29C10A

The following tables specify the guaranteed performance of the Am29C10A over the military operating range of −55 to +125°C, with V_{CC} from 4.5 to 5.5 V. All data are in ns, with measurements made at 1.5 V. All outputs have maximum DC load. C_L = 50 pF

A. Setup and Hold Times

Input	t_s	t_h
$D_i \rightarrow R$	16	2
$D_i \rightarrow PC$	30	2
I_0-I_3	38	0
$\overline{CC}$	35	0
$\overline{CCEN}$	35	0
CI	18	0
$\overline{RLD}$	20	0

B. Combinational Delays

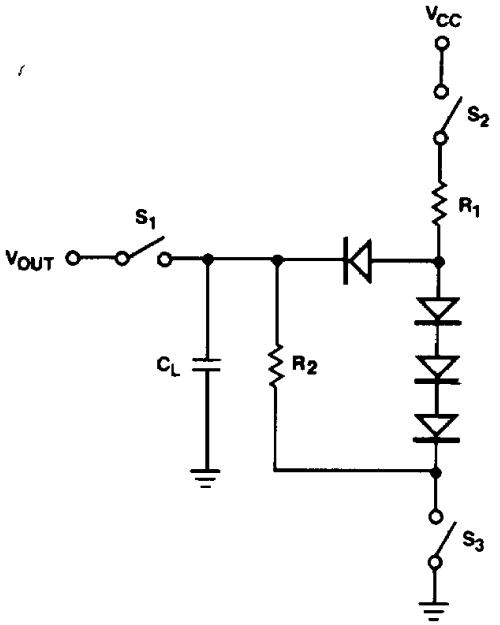
Input	Y	$\overline{PL}$, $\overline{VECT}$, $\overline{MAP}$	$\overline{FULL}$
D_0-D_{11}	25	–	–
I_0-I_3	40	35	–
$\overline{CC}$	36	–	–
$\overline{CCEN}$	36	–	–
CP	46	–	35
$\overline{OE}$ (Note 1)	25/30	–	–

C. Clock Requirements

Minimum Clock LOW Time	25	ns
Minimum Clock HIGH Time	25	ns
Minimum Clock Period	51	ns

19.6

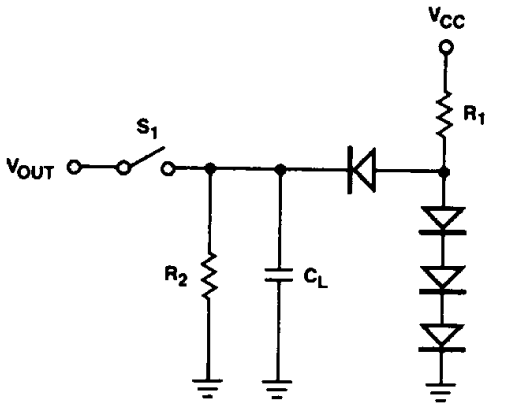
SWITCHING TEST CIRCUITS



TCR01352

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

A. Three-State Outputs



TCR01370

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

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

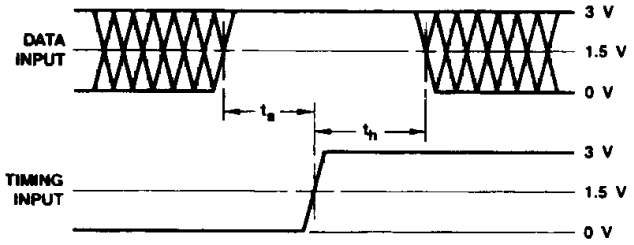
B. Normal Outputs

Notes: 1. C_L = 50 pF includes scope probe, wiring and stray capacities 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.

TEST OUTPUT LOADS FOR Am29C10A

Pin# (DIP)	Pin Label	Test Circuit	R_1	R_2
–	$Y_{0\text{-}11}$	A	300	1K
5	$\overline{VECT}$	B	470	1.5K
6	$\overline{PL}$	B	470	1.5K
7	$\overline{MAP}$	B	470	1.5K
16	$\overline{FULL}$	B	470	1.5K

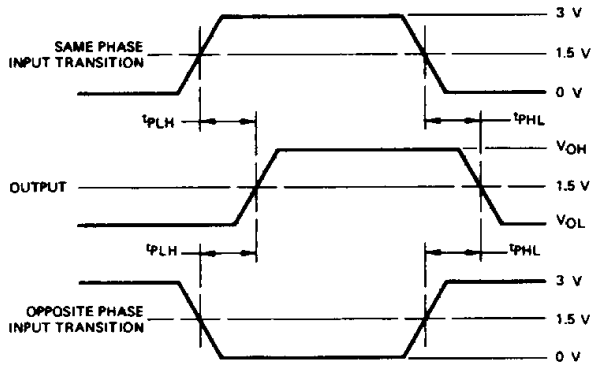
SWITCHING TEST WAVEFORMS



WFR02970

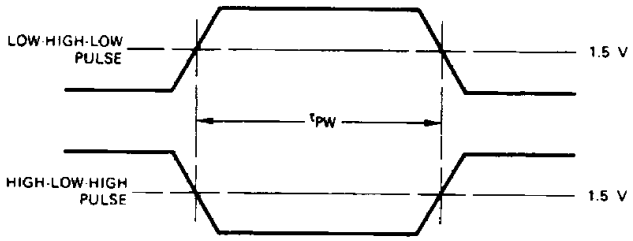
Notes: 1. Diagram shown for HIGH data only. Output transition may be opposite sense.
2. Cross hatched area is don't care condition.

Setup, Hold, and Release Times



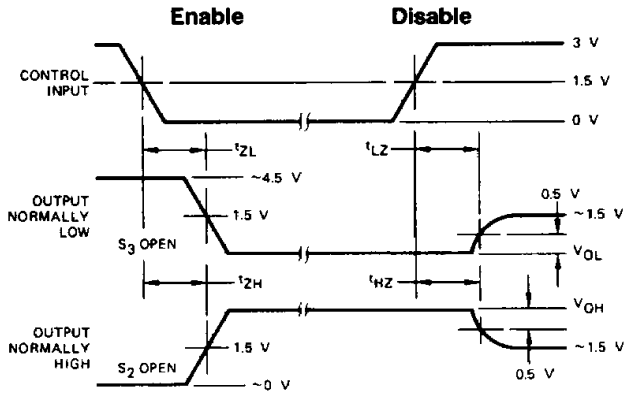
WFR02980

Propagation Delay



WFR02790

Pulse Width

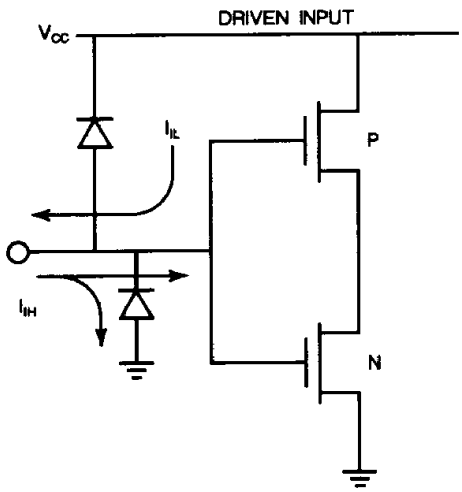


WFR02660

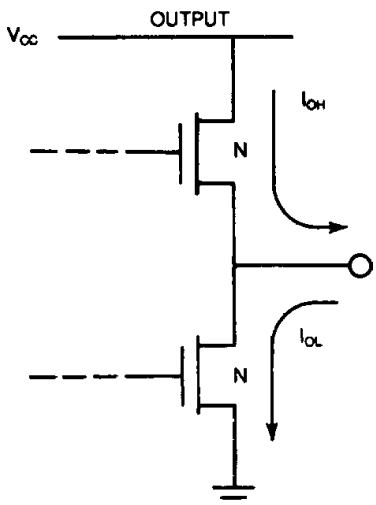
Notes: 1. Diagram shown for Input Control Enable-LOW and Input Control Disable-HIGH.
2. S_1, S_2 and S_3 of Load Circuit are closed except where shown.

Enable and Disable Times

INPUT/OUTPUT CIRCUIT DIAGRAMS*



IC000863



IC000870

*$C_I \approx 5.0$ pF, all inputs (Plastic Pkg.)
$C_I \approx 10.0$ pF, all inputs (Ceramic Pkg.)

Notes on Test Methods

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

1. Ensure the part is adequately decoupled at the test head. Large changes in supply current when the device switches may cause function failures due to V_{CC} changes.

2. Do not leave inputs floating during any tests, as they may oscillate at high frequency.

3. Do not attempt to perform threshold tests at high speed. Following an input transition, ground current may change by as much as 400 mA in 5 – 8 ns. Inductance in the ground cable may allow the ground pin at the device to rise by hundreds of millivolts momentarily.

4. Use extreme care in defining input levels for AC tests. Many inputs may be changed at once, so there will be significant noise at the device pins that may not actually reach V_{IL} or V_{IH} until the noise has settled. AMD recommends using $V_{IL} \leqslant 0$ V and $V_{IH} \geqslant 3$ V for AC tests.

5. To simplify failure analysis, programs should be designed to perform DC, Function, and AC tests as three distinct groups of tests.

6. Capacitive Loading for AC Testing

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

   Similarly, a product may be specified at more than one capacitive load. Since the typical automatic tester is not capable of switching loads in mid-test, it is impossible to make measurements at both capacitances even though they may both be greater than the stray capacitance. In these cases, a measurement is made at one of the two capacitances. The result at the other capacitance is predicted from engineering correlations based on data taken with a bench 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 A.C. testing are performed at "hard" input levels rather than at V_{IL} Max. and V_{IH} Min.

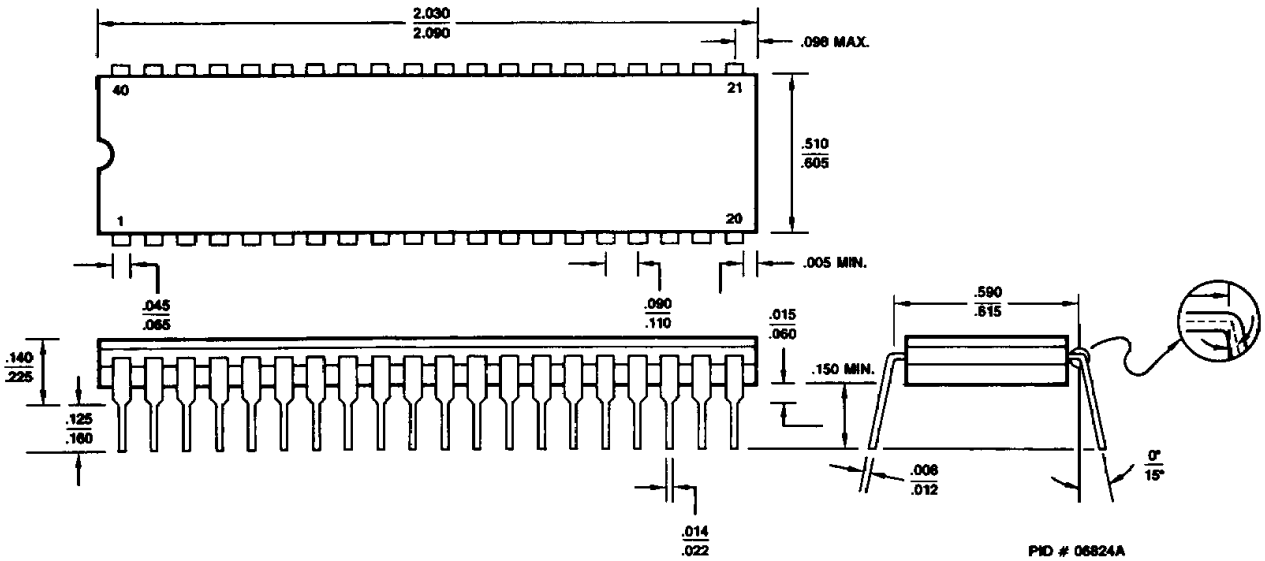
8. A.C. Testing

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

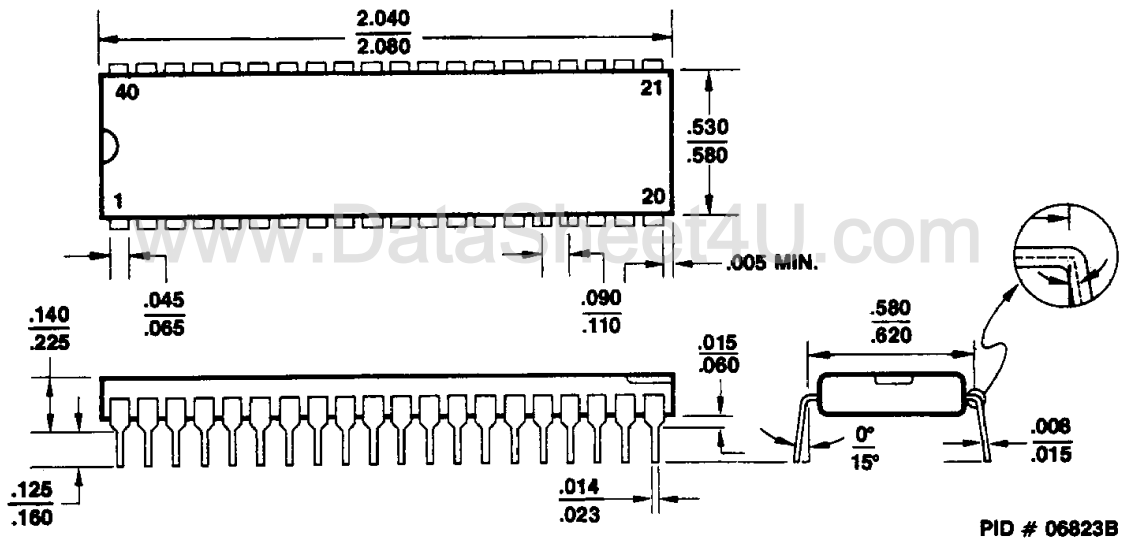
   In some cases, certain AC tests are redundant since they can be shown to be predicted by other tests that have already been performed. In these cases, the redundant tests are not performed.

PHYSICAL DIMENSIONS*

CD 040



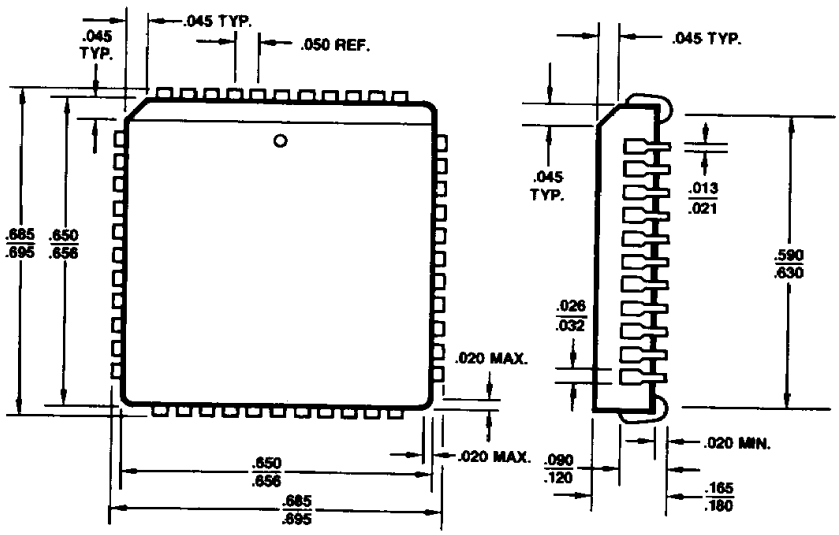
PD 040



*For reference only.

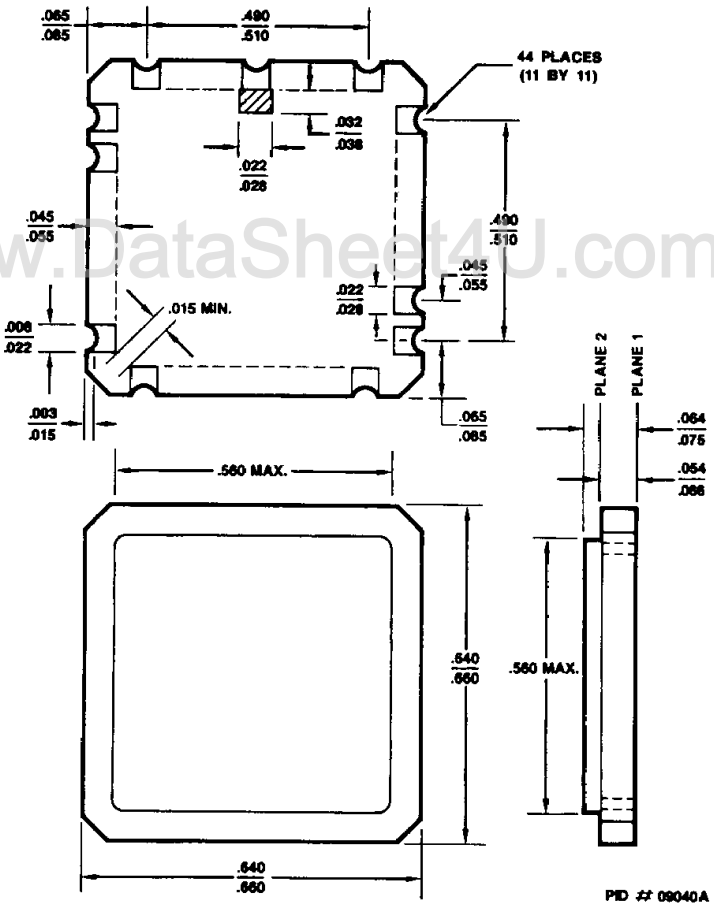
PHYSICAL DIMENSIONS (Cont'd.)

PL 044



PID # 06752B

CLT044



PID # 09040A