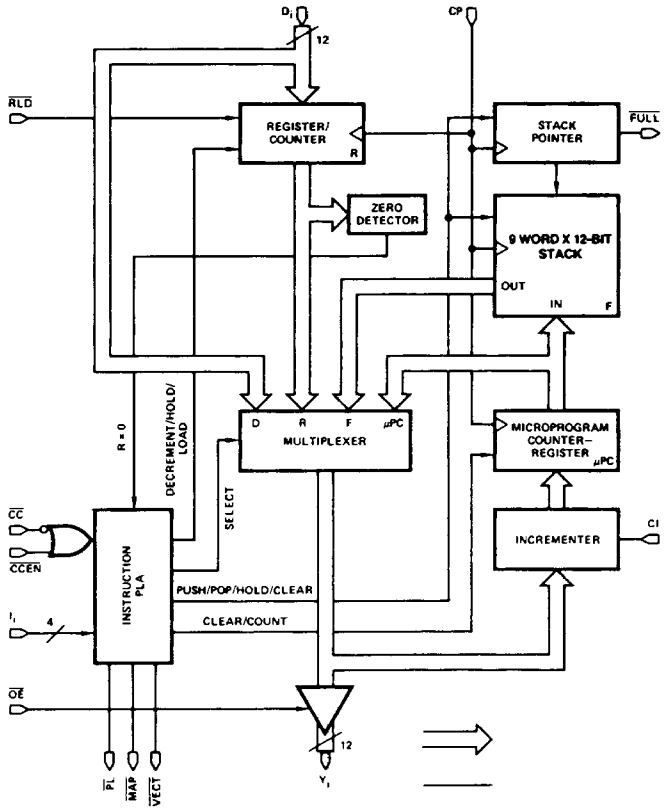
Am2910A

Microprogram Controller

DISTINCTIVE CHARACTERISTICS

- **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 for Three Branch Address Sources**
  Built-in decoder function to enable external devices onto branch address bus. Eliminates external decoder.
- **Fast**
  The Am2910A supports 100ns cycle times and is 25 – 30% faster than the Am2910.

BLOCK DIAGRAM



BDR02320

Publication #	Rev.	Amendment
04522	C	/0

Issue Date: **January 1989**

GENERAL DESCRIPTION

The Am2910A 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 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 (F).

The Am2910A is a speed improved plug-in replacement of the Am2910 featuring AMD's ion-implanted micro-oxide (IMOX) processing and offering 25–30% speed improvement. The Am2910A also features a nine-word deep stack versus the five-deep stack of the Am2910.

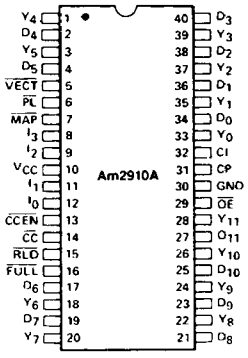
RELATED AMD PRODUCTS

Part No.	Description
Am25LS377	Status Register
Am27S35	Registered PROM
Am2901C	4-Bit Microprocessor Slice
Am2904	Status and Shift Control Unit
Am29C10A	CMOS Microprogram Controller
Am2914	Vectored Interrupt Controller
Am2918	Pipeline Register
Am2922	Condition Code MUX
Am2940	DMA Address Generator
Am2952A	Bidirectional I/O Port
Am29800A	High-Performance Bus Interface Family
Am29C800	High-Performance CMOS BUS Interface Family
Am29818A	SSR™ Diagnostics/Pipeline Register
Am29CPL141	CMOS 64-Word EPROM Field Programmable Controller
Am29CPL142	CMOS 128-Word EPROM Field Programmable Controller
Am29CPL144	CMOS 512-Word EPROM Field Programmable Controller
Am29CPL151	CMOS 64-Word EPROM Field Programmable Controller
Am29CPL152	CMOS 128-Word EPROM Field Programmable Controller
Am29CPL154	CMOS 512-Word EPROM Field Programmable Controller
Am29C332	CMOS 32-Bit Integer ALU
Am29C334	CMOS 64-Word x 18-Bit Register File
Am29C323	CMOS 32 x 32 Integer Multiplier
Am29C325	CMOS 32-Bit Single Precision Floating-Point Processor
Am29C327	CMOS 64-Bit Double Precision Floating-Point Processor

SSR is a trademark of Advanced Micro Devices, Inc.

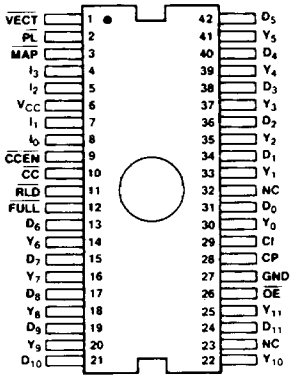
CONNECTION DIAGRAMS
Top View

DIPs



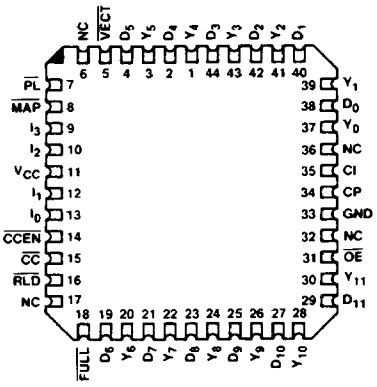
CD004670

Flatpack



CD004680

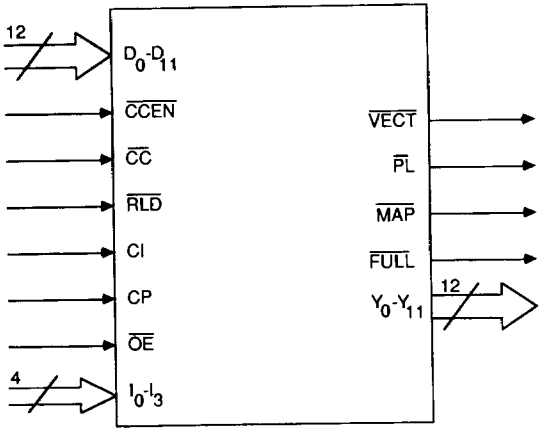
LCC



CD004690

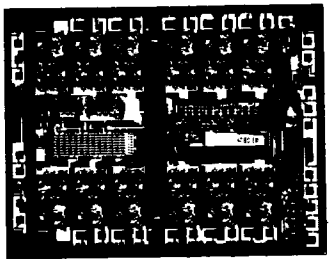
Note: Pin 1 is marked for orientation.

LOGIC SYMBOL



LS002940

METALLIZATION AND PAD LAYOUT



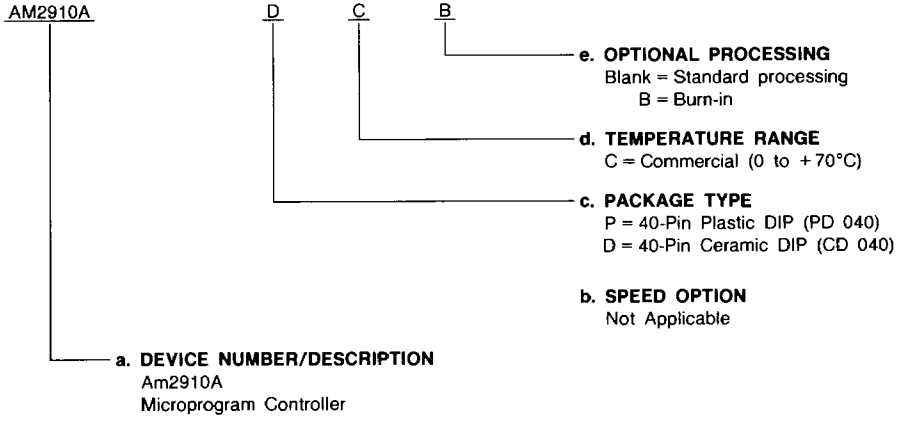
Component Count: 2,345
Die Size: 0.170" x 0.194"

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	
AM2910A	DC, DCB, PC, PCB

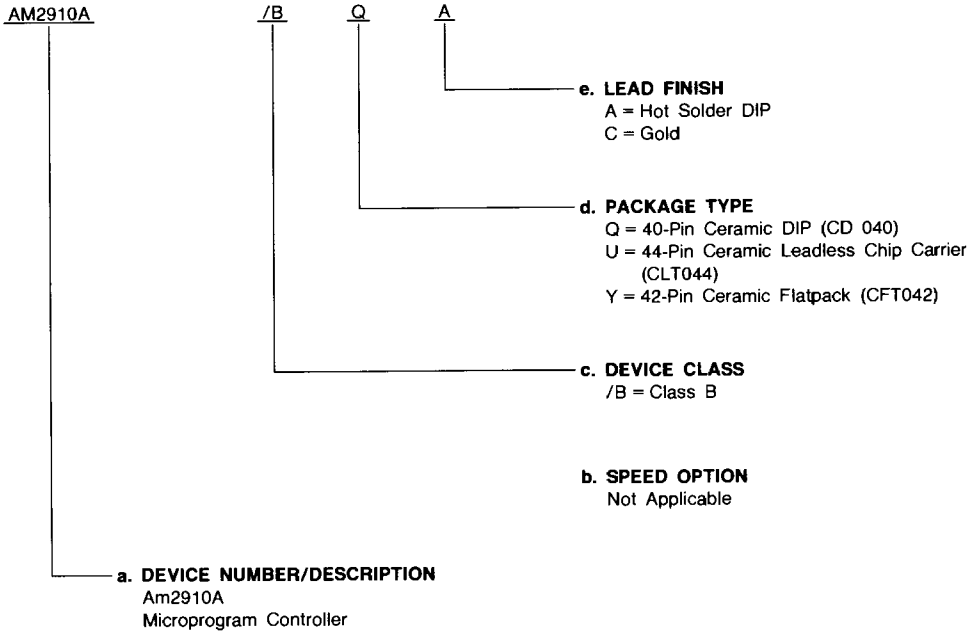
Valid Combinations

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.

MILITARY ORDERING INFORMATION

APL Products

AMD products for Aerospace and Defense applications are available in several packages and operating ranges. APL (Approved Products List) products are fully compliant with MIL-STD-883C requirements. The order number (Valid Combination) 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	
AM2910A	/BYC, /BUA, /BQA

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 Am2910A.

$\overline{MAP}$ 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.

PRODUCT OVERVIEW

The Am2910A is a bipolar microprogram controller intended for use in high-speed 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, direct input, microprogram counter, or stack as the source of the next microinstruction address.

The register/counter consists of 12 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 register/counter is available to the multiplexer as a source for the next microinstruction address. The direct input furnishes a source of data for loading the register/counter.

The Am2910A contains a microprogram counter (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 direct (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 stack pointer operates as an up/down counter. During microinstructions 1,4, and 5, the PUSH operation may occur. This causes the stack pointer to increment and the file to be written with the required return linkage. On the cycle following the PUSH, the return data is at the new location pointed to by the stack pointer.

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

The stack pointer linkage is such that any sequence of pushes, pops, or stack references can be achieved. At RESET (Instruction 0), the depth of nesting becomes zero. For each PUSH, the nesting depth increases by one; for each POP, the depth 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 stack pointer unchanged. This operation will usually destroy useful information and is normally avoided. A POP from an empty stack may place non-meaningful data on the Y outputs, but is otherwise safe. The stack pointer remains at zero whenever a POP is attempted from a stack already empty.

The register/counter is operated during three microinstructions (8,9,15) as a 12-bit down counter, with result = zero available as a microinstruction branch test criterion. This provides efficient iteration of microinstructions. The register/counter 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

The Table of Instructions 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 and the stack after

the next positive-going clock edge is also shown. The multiplexer determines which internal source drives the Y outputs. The value loaded into 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 Functional Pin Description), can modify instruction execution. The combination $\overline{CC}$ HIGH and $\overline{CCEN}$ LOW is used as a test in 9 of the 16 instructions. $\overline{RLD}$, when LOW, causes the D input to be loaded into the register/counter, overriding any HOLD or DEC operation specified in the instruction. $\overline{OE}$, normally LOW, may be forced HIGH to remove the Am2910A Y outputs from a three-state bus.

The stack, 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 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 stack pointer 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 OF INSTRUCTIONS

			REG/ CNTR CON-	FAIL $\overline{CCEN}$ = L and $\overline{CC}$ = H		PASS $\overline{CCEN}$ = H or $\overline{CC}$ = L		REG/	
I_3-I_0	MNEMONIC	NAME	TENTS	Y	STACK	Y	STACK	CNTR	ENABLE
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 Am2910A INSTRUCTION SET

The Am2910A provides 16 instructions which 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 which is partially controlled by external, data-dependent condition. Three of the instructions have an effect which is partially controlled by the contents of the internal register/counter. 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 Am2910A 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 register/counter. 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 register/counter.

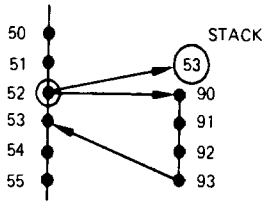
Perhaps the best technique for understanding the Am2910A is to simply take each instruction and review its operation. In order to provide some feel for the actual execution of these instructions, examples of all 16 instructions are included.

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 number 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.

It might be appropriate at this time to mention that AMD has a microprogram assembler called AMDASM, which has the capability of using the Am2910A instructions in symbolic representation. AMDASM's Am2910A instruction symbolics (or mnemonics) are given in the examples, and again in the Table of Instructions.

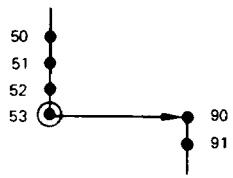
1 COND JSB PL (CJS)



PFR00950

Instruction 1 is a CONDITIONAL JUMP-TO-SUBROUTINE via the address provided in the pipeline register. As shown in Figure 4, the machine might have executed words at address 50, 51, and 52. When the contents of address 52 is 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.

2 JUMP MAP (JMAP)



PFR00960

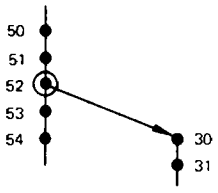
0 JUMP ZERO (JZ)

PFR00830

Instruction 0, JZ (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 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 of Figure 4, 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.

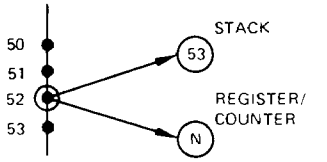
3 COND JUMP PL (CJP)



PFR00820

Instruction 3, CONDITIONAL JUMP PIPELINE, derives its branch address from the pipeline register branch address value (BR_0 - BR_{11}). This instruction provides a technique for branching to various microprogram sequences depending upon the test condition inputs. Quite often, state machines are designed which 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 microprogram counter which, in this example, is 53.

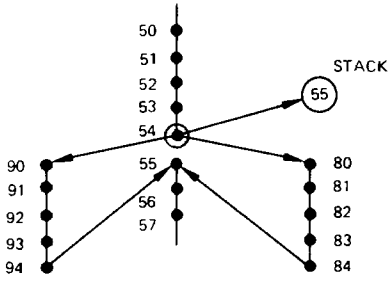
4 PUSH/COND LD CNTR (PUSH)



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 instruction 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 register/counter for looping.

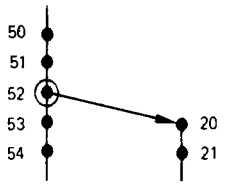
5 COND JSB R/PL (JSRP)



PFR00940

Instruction 5 is a CONDITIONAL JUMP-TO-SUBROUTINE via the register/counter 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 number 10) returns the microprogram flow to address 55. In order for this microinstruction control sequence to operate correctly, both the next address fields of instruction 53 and the next address fields of instruction 54 would have to contain the proper value. Let's assume that the branch address fields of instruction 53 contain the value 90 so that it will be in the Am2910A register/counter when the contents of address 54 are in the pipeline register. This requires that the instruction at address 53 load the register/counter. Now, during the execution of instruction 5 (at address 54), if the test failed, 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.

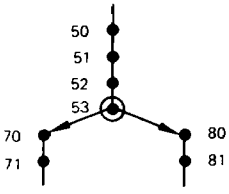
6 COND JUMP VECTOR (CJV)



PFR00920

Instruction 6 is a CONDITIONAL JUMP VECTOR instruction which provides the capability to take the branch address from a third source heretofore not discussed. In order for this instruction to be useful, the Am2910A 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. 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.

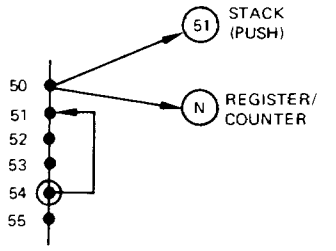
7 COND JUMP R/PL (JRP)



PFR00930

Instruction 7 is a CONDITIONAL JUMP via the contents of the Am2910A 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 PL. 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 is clocked into the pipeline register, the value 70 is loaded into the register/counter in the Am2910A. The value 80 is available when the contents of address 53 is in the pipeline register. Thus, control is transferred to either address 70 or address 80 depending on the test condition.

8 REPEAT LOOP, CNTR ≠ 0 (RFCT)



PFR00910

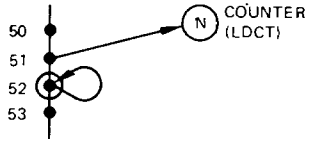
Instruction 8 is the REPEAT LOOP, COUNTER ≠ ZERO instruction. This microinstruction makes use of the decrementing capability of the register/counter. To be useful, some previous instruction, such as 4, must have loaded a count value into the register/counter. This instruction checks to see whether the register/counter contains a non-zero value. If so, the register/counter is decremented, and the address of the next microinstruction is taken from the top of the stack. If the register/counter contains zero, the loop exit condition is occurring; control falls through to the next sequential microinstruction by selecting μPC; the stack is POP'd by decrementing the stack pointer, 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 (microaddress 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.

9 REPEAT PL, CNTR ≠ 0 (RPCT)

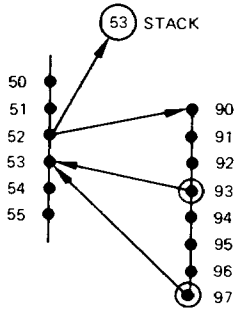


PFR00890

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; and, 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, COUNTER ≠ ZERO instruction is instruction 52 and is shown as a single microinstruction loop. The address in the pipeline register would be 52. Instruction 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.

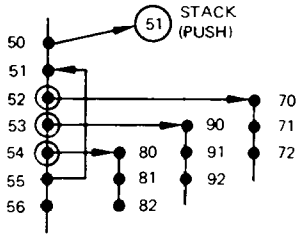
10 COND RETURN (CRTN)



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 instruction location 52 where control is transferred to location 90. At location 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; the microinstruction at address 97 is programmed to force $\overline{CCEN}$ HIGH, disabling the test and the forced PASS causes an unconditional return.

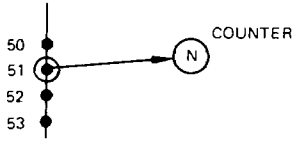
11 COND JUMP PL & POP (CJPP)



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 and POP instructions. At address 52, if the $\overline{CC}$ 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.

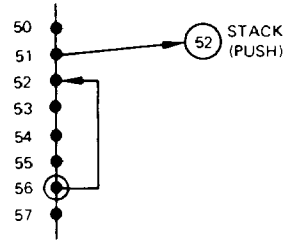
12 LD CNTR & CONTINUE (LDCT)



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 – the explicit load by this instruction 12; the conditional load included as part of instruction 4; and the use of $\overline{RLD}$ 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. This 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 in those implementations which do not provide microprogrammed control for $\overline{RLD}$.

13 TEST END LOOP (LOOP)



PFR00860

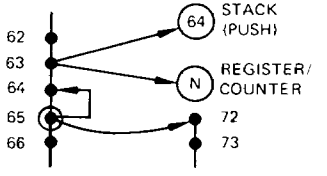
Instruction 13 is the TEST END-OF-LOOP instruction, which provides the capability of conditionally exiting a loop at the bottom; that is, this is a conditional instruction that will cause the microprogram to loop, via the file, if the test is failed else to continue to the next sequential instruction. 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 instruction 56, the loop is terminated and the next sequential microinstruction at address 57 is executed, which also causes the stack to be POP'd; thus, accomplishing the required stack maintenance.

14 CONTINUE (CONT)

PFR00880

Instruction 14 is the CONTINUE instruction, which simply causes the microprogram counter to increment so that the next sequential microinstruction is executed. This is the simplest microinstruction of all and should be the default instruction which the firmware requests whenever there is nothing better to do.

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 provides 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 register/counter 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; then the next address 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 microprogram counter 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 POP'd by decrementing the stack pointer, since interest in the value contained at the top of the stack is then complete.

The application of instruction 15 can enhance performance of a variety of machine-level instructions. For instance, (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 disc controller processing variable length records; (4) normalization of a floating point number.

As one example, consider the case of a memory search instruction. As shown, 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 which fetches the next operand from the memory area to be searched and compares it with the search key. Location 65 contains a microinstruction which tests the result of the comparison and also is 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 POP'd once, removing the value 64 from the top of the stack.

ARCHITECTURES USING THE Am2910A

(Shading shows path(s) which usually limit speed)

One Level Pipeline Based (Recommended)

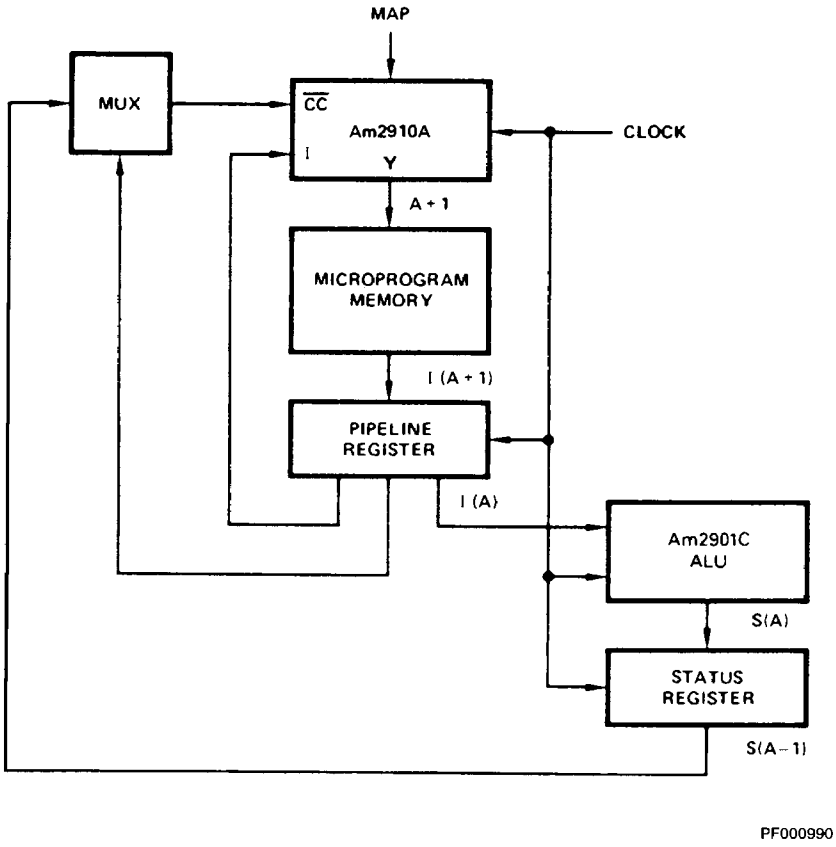
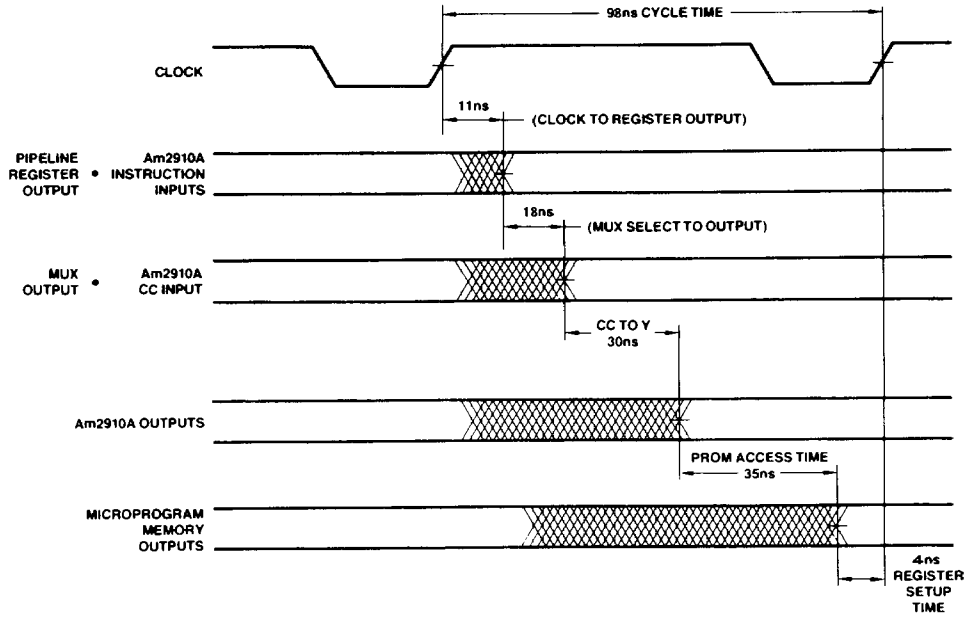


Figure 1.

One level pipeline provides better speed than most other architectures. The μProgram Memory and the Am2901 array are in parallel speed paths instead of in series. This is the recommended architecture for Am2900 designs.

Typical CCU Cycle Timing Waveforms



WF003051

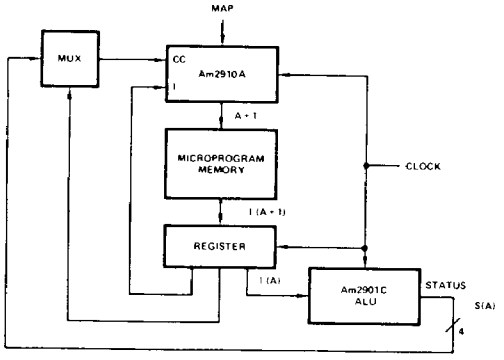
This drawing shows the timing relationships in the CCU illustrated above.

OTHER ARCHITECTURES USING THE Am2910A

(Shading shows path(s) which usually limit speed)

Figure 2.

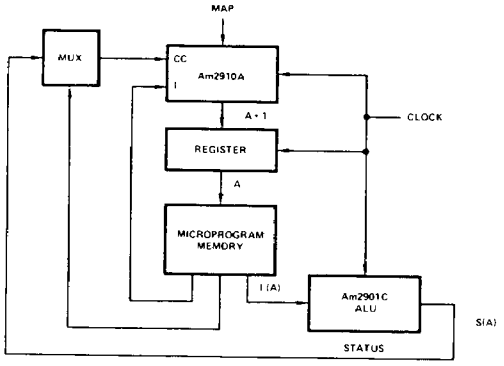
a. Instruction Based



PF001001

A Register at the Microprogram Memory output contains the microinstruction being executed. The microprogram memory and Am2901C delay are in series. Conditional branches are executed on same cycle as the ALU operation generating the condition.

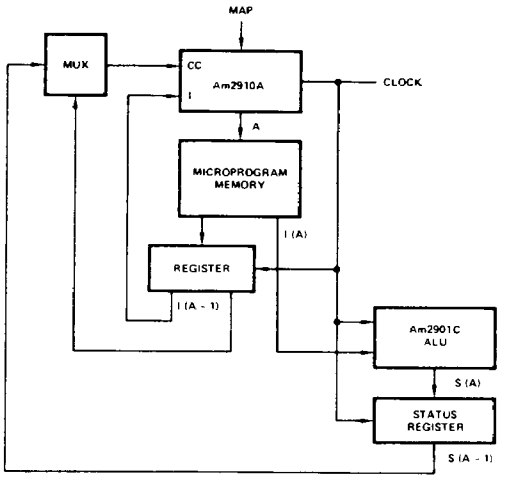
b. Addressed Based



PF001011

The Register at the Am2910A output contains the address of the microinstruction being executed. The Microprogram Memory and Am2901C 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).

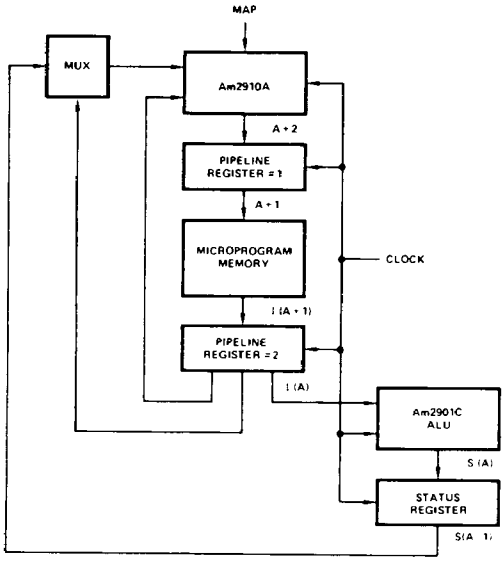
c. Data Based



PF001021

The Status Register provides conditional Branch control based on results of previous ALU cycle. The Microprogram Memory and Am2901C are in series in the critical paths.

d. Two Level Pipeline Based



PF001051

Two level pipeline provides highest possible speed. It is more difficult to program because the selection of a microinstruction occurs two instructions ahead of its execution.

Am2910A HIGH-SPEED APPLICATION

Optimal Am2910A configurations can support high-speed bit slice designs. When used with high-speed registers and PROMs, the Am2910A can execute simple instructions in 100ns.

The following figure illustrates the usual critical path in the sequencer.

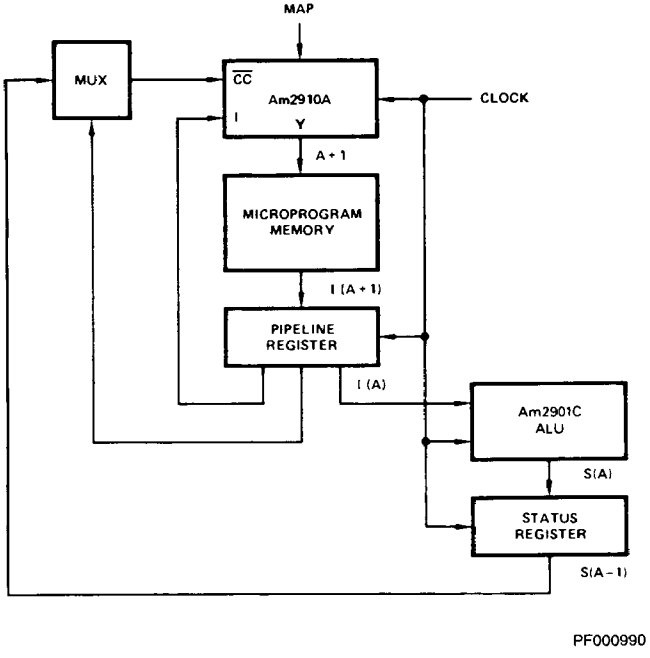
Timing on the critical paths becomes:

Device	Path	Delay
Status Register	Clock → Output	8ns
Fast MUX	Select → Output	18ns
Am2910A	$\overline{CC}$ → Y	30ns
Fast PROM	Addr → Output	35ns
Pipeline Register	Setup	4ns
		96ns

The following gives one suggested parts configuration to meet this design criterion.

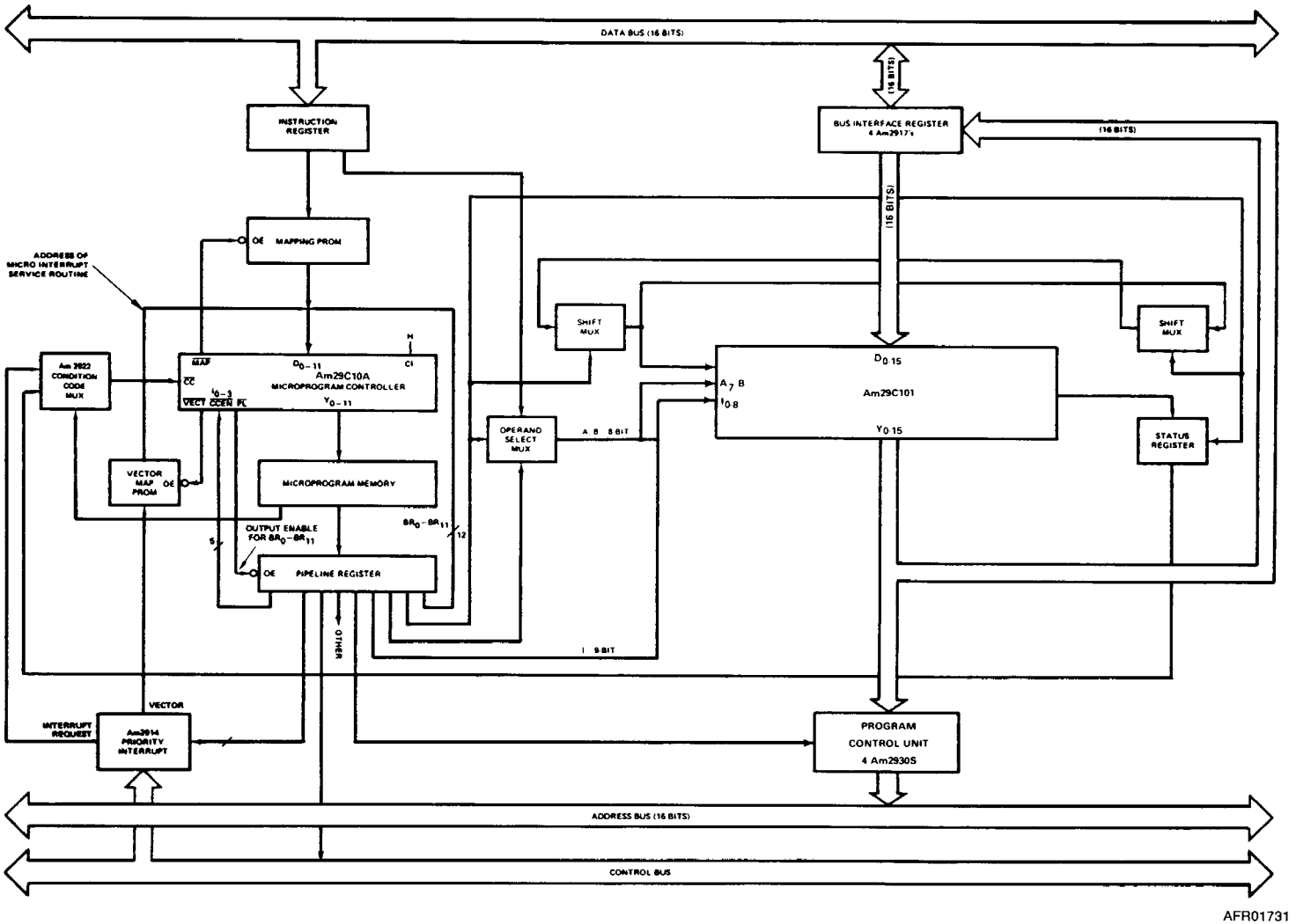
Status Register	Am29825A
MUX (8 to 1)	74S151
PROM	Am27S35
Pipeline Register	Am2918

One Level Pipeline Based (Recommended)



PF000990

Figure 3.



AFR01731

Figure 4. Typical Bipolar Microcomputer Using Am2910A

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 ..... −0.5 V to +V_{CC} max
DC Input Voltage ..... −0.5 V to +5.5 V
DC Output Current, Into Outputs (Except Bus) .....30 mA
DC Input Current ..... −30 to +5.0 mA

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

OPERATING RANGES

Commercial (C) Devices
Ambient Temperature (T_A) ..... 0°C to +70°C
Supply Voltage(V_{CC}) to ..... +4.75 V to +5.25 V

Military (M) Devices
Case Temperature (T_C) ..... −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.

Thermal Resistance (Typical)

Symbol	CD 040	PD 040	CLT044	CFT042	Unit
θ_{JA}	38	52	57	55	°C/W
θ_{JC}	3	N/A	4	4	°C/W

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

Parameter Symbol	Parameter Description	Test Conditions (Note 1)			Min	Typ (Note 2)	Max	Unit
V_{OH}	Output HIGH Voltage	V_{CC} = MIN, I_{OH} = −1.6mA, V_{IN} = V_{IH} or V_{IL}			2.4			
V_{OL}	Output LOW Voltage	V_{CC} = MIN, V_{IN} = V_{IH} or V_{IL}	Y_{0-11}, I_{OL} = 12mA				0.5	
			$\overline{PL}$, $\overline{VECT}$, $\overline{MAP}$, $\overline{FULL}$, I_{OL} = 8mA					
V_{IH}	Input HIGH Level (Note 4)	Guaranteed input logical HIGH voltage for all inputs			2.0			V
V_{IL}	Input LOW Level (Note 4)	Guaranteed input logical LOW voltage for all inputs					0.8	
V_{CLMP}	Input Clamp Voltage	V_{CC} = MIN, I_{IN} = −18 mA					−1.5	
I_{IL}	Input LOW Current	V_{CC} = MAX, V_{IN} = 0.5 V		D_{0-11}			−0.87	mA
				C1, $\overline{CCEN}$			−0.54	
				I_{0-3}, $\overline{OE}$,$\overline{RLD}$			−0.72	
				$\overline{CC}$			−1.31	
				CP			−2.14	
I_{IH}	Input HIGH Current	V_{CC} = MAX, V_{IN} = 2.7 V		D_{0-11}			80	µA
				C1, $\overline{CCEN}$			30	
				I_{0-3}, $\overline{OE}$, $\overline{RLD}$			40	
				$\overline{CC}$			50	
				CP			100	
I_I	Input HIGH Current	V_{CC} = MAX, V_{IN} = 5.5 V					1.0	mA
I_{SC}	Output Short-Circuit Current (Note 3)	V_{CC} = MAX+0.5 V, I_O = 0.5 V			−30		−85	mA
I_{OZL}	Output OFF Current	V_{CC} = MAX, $\overline{OE}$ = 2.4V		V_{OUT} = 0.5V			−50	µA
I_{OZH}				V_{OUT} = 2.4V			50	
I_{CC}	Power Supply Current	V_{CC} = MAX	COM'L	T_A = 0 to +70°C			344	mA
				T_A = +70°C			280	
			MIL	T_C = −55 to +125°C			340	
				T_C = +125°C			280	

Notes: 1. For conditions shown as MIN or MAX, use the appropriate value specified under Operating Ranges for the applicable device type.
2. Typical limits are at V_{CC} = 5.0 V, 25°C ambient and a maximum loading.
3. Not more than one output should be shorted at a time. Duration of the short-circuit test should not exceed one second.
4. These input levels provide no guaranteed noise immunity and should only be static tested in a noise-free environment (not functionally tested).

SWITCHING CHARACTERISTICS over COMMERCIAL operating range

The tables below define the Am2910A switching characteristics. Tables A are set up and hold times relative to the clock LOW-to-HIGH transition. Tables B are combinational delays. Tables C are clock requirements. All measurements are made at 1.5 V with input levels at 0 V or 3 V. All values are in ns. All outputs have maximum DC loading.

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

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

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	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

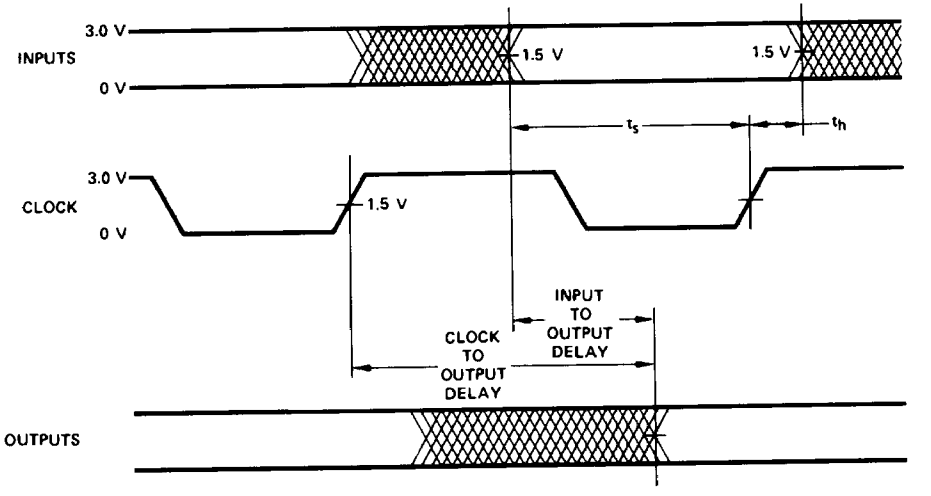
Note 1. Enable/Disable. Disable times measured to 0.5V change on output voltage level with C_L = 5.0pF.

SWITCHING WAVEFORMS

Key to Switching Waveforms

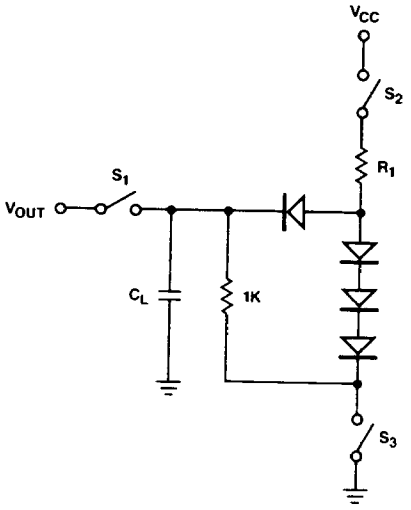
WAVEFORM	INPUTS	OUTPUTS
———	MUST BE STEADY	WILL BE STEADY
———		
⫽⫽⫽	MAY CHANGE FROM H TO L	WILL BE CHANGING FROM H TO L
⫽⫽⫽	MAY CHANGE FROM L TO H	WILL BE CHANGING FROM L TO H
⫽⫽⫽	DON'T CARE; ANY CHANGE PERMITTED	CHANGING; STATE UNKNOWN
⫽⫽⫽	DOES NOT APPLY	CENTER LINE IS HIGH IMPEDANCE "OFF" STATE

KS000010



WFR02990

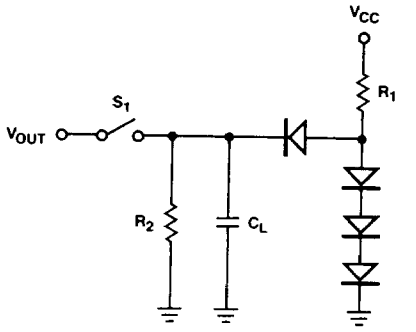
SWITCHING TEST CIRCUITS



TCR01350

A. Three-State Outputs

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



TCR01370

B. Normal Outputs

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

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

Notes: 1. C_L = 50pF 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 = 5.0pF for output disable tests.

TEST OUTPUT LOADS FOR Am2910A

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

TEST PHILOSOPHY AND METHODS

The following nine points describe AMD's philosophy for high volume, high speed automatic testing.

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

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

3. Do not attempt to perform threshold tests at high speed. Following an output transition, ground current may change by as much as 600 mA in 5-8 ns. Inductance in the ground cable may allow the ground pin at the device to rise by hundreds of millivolts momentarily.

4. Use extreme care in defining point input levels for AC tests. Many inputs may be changed at once, so there will be significant noise at the device pins and they may not actually reach V_{IL} or V_{IH} until the noise has settled. AMD recommends using $V_{IL} \leqslant 0$ V and $V_{IH} \geqslant 3.0$ V for AC tests.

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

6. Capacitive Loading for AC Testing

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

   Similarly, a product may be specified at more than one capacitive load. Since the typical automatic tester is not capable of switching loads in mid-test, it is impossible to make measurements at both capacitances even though they may both be greater than the stray capacitance. In these cases, a measurement is made at one of the two capacitances. The result at the other capacitance is predicted from engineering correlations based on data taken with a bench setup and the knowledge that certain DC measurements (I_{OH}, I_{OL} for example) have already been taken and are within spec. In some cases, special DC tests are performed in order to facilitate this correlation.

7. Threshold Testing

   The noise associated with automatic testing (due to the long, inductive cables) and the high gain of the tested device when in the vicinity of the actual device threshold, frequently give rise to oscillations when testing high speed circuits. These oscillations are not indicative of a reject device, but instead of an overtaxed test system. To minimize this problem, thresholds are tested at least once for each input pin. Thereafter, "hard" high and low levels are used for other tests. Generally this means that function and AC testing are performed at "hard" input levels rather than at V_{IL} Max. and V_{IH} Min.

8. AC Testing

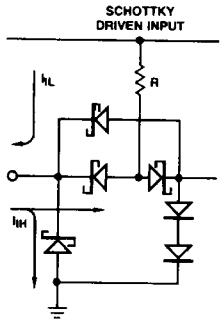
   Occasionally, parameters are specified that cannot be measured directly on automatic testers because of tester limitations. Data input hold times often fall into this category. In these cases, the parameter in question is guaranteed by correlating these tests with other AC tests that have been performed.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

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

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} = Max. case.

INPUT/OUTPUT CURRENT INTERFACES



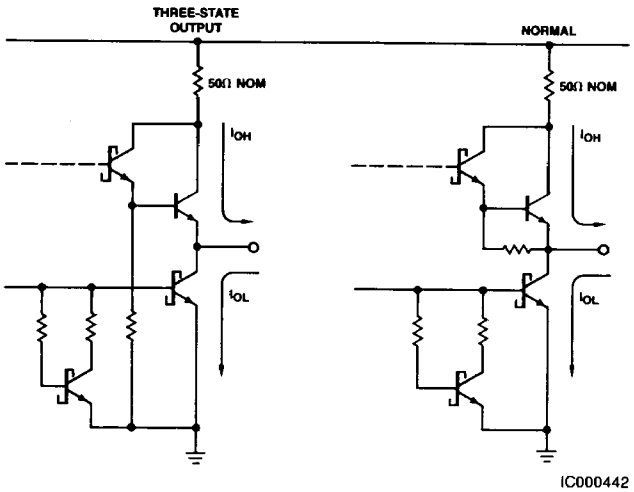
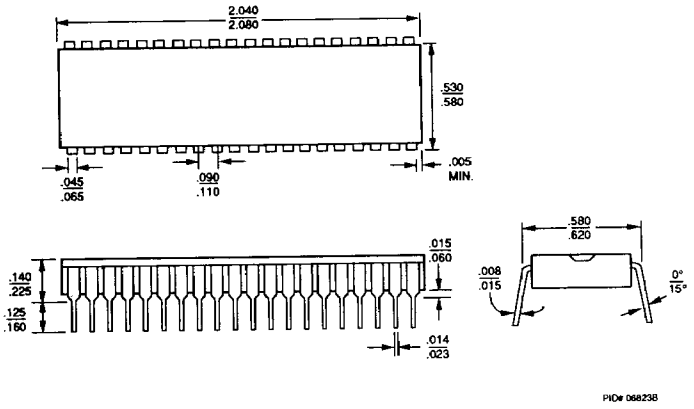


Figure 11.

PHYSICAL DIMENSIONS*

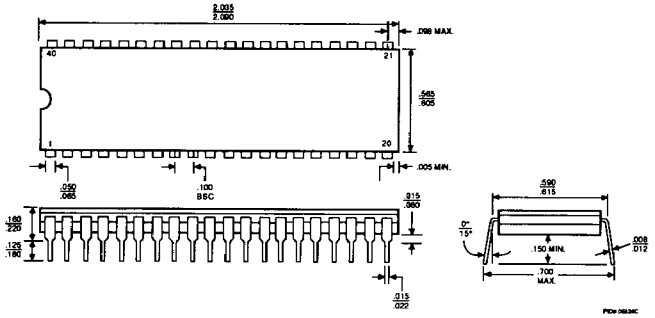
PD 040



*For reference only. All dimensions are measured in inches. BSC is an ANSI standard for Basic Space Centering.

PHYSICAL DIMENSIONS (Cont'd.)

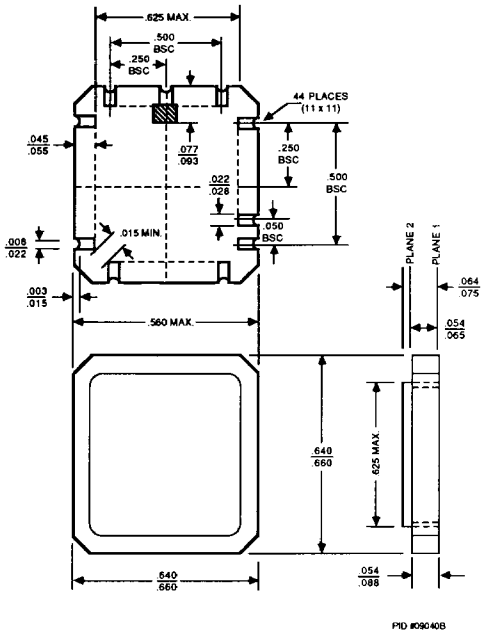
CD 040



*For reference only. All dimensions are measured in inches. BSC is an ANSI standard for Basic Space Centering.

PHYSICAL DIMENSIONS (Cont'd.)

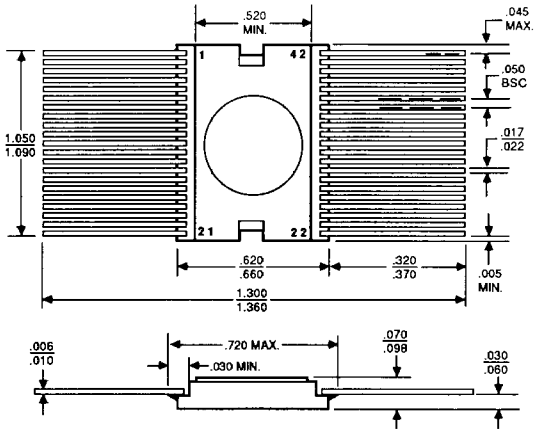
CLT044



*For reference only. All dimensions are measured in inches. BSC is an ANSI standard for Basic Space Centering.

PHYSICAL DIMENSIONS (Cont'd.)

CFT042



PID #07415D

*For reference only. All dimensions are measured in inches. BSC is an ANSI standard for Basic Space Centering.

Sales Offices

North American

ALABAMA	(205) 882-9122
ARIZONA	(602) 242-4400
CALIFORNIA,	
Culver City	(213) 645-1524
Newport Beach	(714) 752-6262
Roseville	(916) 786-6700
San Diego	(619) 560-7030
San Jose	(408) 452-0500
Woodland Hills	(818) 992-4155
CANADA, Ontario,	
Kanata	(613) 592-0060
Willowdale	(416) 224-5193
COLORADO	(303) 741-2900
CONNECTICUT	(203) 264-7800
FLORIDA,	
Clearwater	(813) 530-9971
Ft. Lauderdale	(305) 776-2001
Orlando	(407) 830-8100
GEORGIA	(404) 449-7920
ILLINOIS,	
Chicago	(312) 773-4422
Naperville	(312) 505-9517
KANSAS	(913) 451-3115
MARYLAND	(301) 796-9310
MASSACHUSETTS	(617) 273-3970
MINNESOTA	(612) 938-0001
MISSOURI	(913) 451-3115
NEW JERSEY,	
Cherry Hill	(609) 662-2900
Parsippany	(201) 299-0002
NEW YORK,	
Liverpool	(315) 457-5400
Poughkeepsie	(914) 471-8180
NORTH CAROLINA	(919) 878-8111
OHIO,	
Columbus	(614) 891-6455
Dayton	(513) 439-0470
OREGON	(503) 245-0080
PENNSYLVANIA	(215) 398-8006
SOUTH CAROLINA	(803) 772-6760
TEXAS,	
Austin	(512) 346-7830
Dallas	(214) 934-9099
Houston	(713) 785-9001

International

BELGIUM, Bruxelles	TEL	(02) 771-91-42
	FAX	(02) 762-37-12
	TLX	61028
FRANCE, Paris	TEL	(1) 49-75-10-10
	FAX	(1) 49-75-10-13
	TLX	263282
WEST GERMANY,		
Hannover area	TEL	(0511) 736085
	FAX	(0511) 721254
	TLX	922850
München	TEL	(089) 4114-0
	FAX	(089) 406490
	TLX	523883
Stuttgart	TEL	(0711) 62 33 77
	FAX	(0711) 625187
	TLX	721882
HONG KONG	TEL	852-5-8654525
	FAX	852-5-8654335
	TLX	67955AMDAPHX
ITALY, Milan	TEL	(02) 3390541
		(02) 3533241
	FAX	(02) 3498000
	TLX	315286
JAPAN,		
Kanagawa	TEL	462-47-2911
	FAX	462-47-1729
Tokyo	TEL	(03) 345-8241
	FAX	(03) 342-5196
	TLX	J24064AMDTKOJ
Osaka	TEL	06-243-3250
	FAX	06-243-3253

International (Continued)

KOREA, Seoul	TEL	822-784-0030
	FAX	822-784-8014
LATIN AMERICA,		
Ft. Lauderdale	TEL	(305) 484-8600
	FAX	(305) 485-9736
	TEL	5109554261 AMDFTL
NORWAY, Hovik	TEL	(02) 537810
	FAX	(02) 591959
	TLX	79079
SINGAPORE	TEL	65-3481188
	FAX	65-3480161
	TLX	55650 AMDMMI
SWEDEN,		
Stockholm	TEL	(08) 733 03 50
	FAX	(08) 733 22 85
	TLX	11602
TAIWAN	TEL	886-2-7213393
	TLX	886-2-7122066
	FAX	886-2-7723422
UNITED KINGDOM,		
Manchester area	TEL	(0925) 828008
	FAX	(0925) 827693
	TLX	628524
London area	TEL	(0483) 740440
	FAX	(0483) 756196
	TLX	859103

North American Representatives

CANADA	
Burnaby, B.C.	
DAVETEK MARKETING	(604) 430-3680
Calgary, Alberta	
DAVETEK MARKETING	(403) 291-4984
Kanata, Ontario	
VITEL ELECTRONICS	(613) 592-0060
Mississauga, Ontario	
VITEL ELECTRONICS	(416) 676-9720
Lachine, Quebec	
VITEL ELECTRONICS	(514) 636-5951
IDAHO	
INTERMOUNTAIN TECH MKTG	(208) 888-6071
ILLINOIS	
HEARTLAND TECHNICAL MARKETING	(312) 577-9222
INDIANA	
ELECTRONIC MARKETING CONSULTANTS, INC	(317) 921-3452
IOWA	
LORENZ SALES	(319) 377-4666
KANSAS	
Merriam – LORENZ SALES	(913) 384-6556
Wichita – LORENZ SALES	(316) 721-0500
KENTUCKY	
ELECTRONIC MARKETING CONSULTANTS, INC	(317) 921-3452
MICHIGAN	
Holland – COM-TEK SALES, INC	(616) 399-7273
Novi – COM-TEK SALES, INC	(313) 344-1409
MISSOURI	
LORENZ SALES	(314) 997-4558
NEBRASKA	
LORENZ SALES	(402) 475-4660
NEW MEXICO	
THORSON DESERT STATES	(505) 293-8555
NEW YORK	
East Syracuse – NYCOM, INC	(315) 437-8343
Woodbury – COMPONENT CONSULTANTS, INC	(516) 364-8020
OHIO	
Centerville – DOLFUSS ROOT & CO	(513) 433-6776
Columbus – DOLFUSS ROOT & CO	(614) 885-4844
Strongsville – DOLFUSS ROOT & CO	(216) 238-0300
PENNSYLVANIA	
DOLFUSS ROOT & CO	(412) 221-4420
PUERTO RICO	
COMP. REP. ASSOCIATES	(809) 746-6550
UTAH	
R^2 MARKETING	(801) 595-0631
WASHINGTON	
ELECTRA TECHNICAL SALE	
WISCONSIN	
HEARTLAND TECHNICAL M	

026709 ✓ _ _

Advanced Micro Devices, Inc. 901 Thompson Place, P.O. Box 3453, Sunnyvale, CA 94088, USA
Tel: (408) 732-2400 • TWX: 910-339-9280 • TELEX: 34-6306 • TOLL FREE: (800) 538-8450
APPLICATIONS HOTLINE TOLL FREE: (800) 222-9323 • (408) 749-5703

2/16/89
AIS-WCP-2.5M-3/89-0 Printed in USA