Programmable Peripheral
SAM448
User-Configurable Microsequencer

Features

- First Generation Programmable System Device
- User-Programmable Microsequencer for Implementing High-Performance State Machines
- On-Chip Reprogrammable EPROM Microcode Memory Up to 448 Words Deep
- 15 x 8 Stack
- Loop Countet
- Prioritized, Multi-Way Control Branching
- 8 General-Purpose Branch Control Inputs
- 16 General-Purpose Control Outputs
- Cascadable to Expand Outputs or States
- Low-Power CMOS Technology
- Footprint Efficient 28 Pin 300 Mil DIP or 28 Lead CLDCC/PLDCC Package
- 25 MHz Maximum Clock Frequency
- High Level PC-XT/AT, PS2 or Compatible Design Support Software (SAM+PLUS):
  - WSI PSD Integrated Software Environment
  - State Machine Input Language
  - Microcode Assembler
  - Functional Simulator
- DESC SMD No. 5962-91747

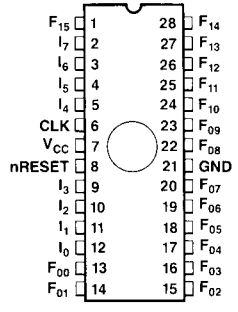
General Description

The SAM448 is WSI's first user programmable microsequencer. On-chip EPROM (up to 448 words) is integrated with Branch Control Logic, Pipeline Register, Stack, and Loop Counter. This generic microcoded architecture provides an efficient vehicle for implementing a broad range of high performance controllers spanning the spectrum from basic state machines to traditional bit-slice controller applications.

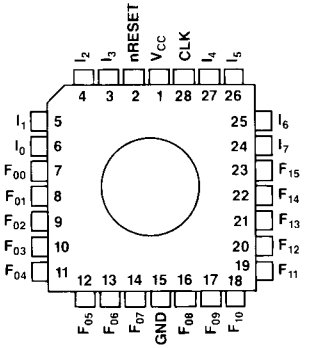
The SAM448 has eight general purpose input pins, a clock pin and reset pin. It has 16 user-definable outputs packaged in a 28-pin 300 Mil DIP or 28 Lead CLDCC/PLDCC package. One-Time-Programmable plastic versions are available to minimize volume production costs.

Pin Configuration (Top View)

Dual-In-Line



Leaded Chip Carrier



Description (Cont.)

Programming the SAM448 device is accomplished on a standard WSI PSD WISPER development system installed with the optional SAM+PLUS software package and device adapters. New users can purchase a separate WISPER-SAM development system with programming hardware included. SAM+PLUS allows designs to be entered in either state machine or microcoded formats. SAM+PLUS automatically performs logic minimization and design fitting for the device. The design may then be simulated or programmed directly to achieve customized working silicon within minutes.

Using WSI's proprietary high performance CMOS EPROM technology allows SAM448 to operate at a 25-MHz typical clock frequency while still enjoying the benefits of low CMOS power consumption. This technology also facilitates 100% generic testability which eliminates the need for post-programming testing.

Ideal application areas for SAM448 include programmable sequence generators (state machines), bus and memory control functions, graphics and DSP algorithm controllers, and other complex, high performance machines. The devices may be cascaded easily to obtain greater output requirements (horizontal cascade) or greater microcode memory depth (vertical cascade) or both.

SAM as a State Machine

The SAM448 architecture allows easy implementation of synchronous state machines. SAM's internal EPROM memory together with its Pipeline Register allows storage of up to 448 unique states. SAM's Branch Control Logic allows single clock, multi-way branching in response to the eight inputs, current device state, and user-defined transition conditions. Design entry is simplified with WSI's State Machine Input Language (ASMILE) supported by the SAM+PLUS development system. This high level language uses IF-THEN-ELSE statements to define state transitions and a truth table to define or tri-state the outputs on a state-by-state basis.

SAM as a Microcoded Sequencer

SAM's architecture has several advanced features that enable it to be used as a sophisticated microcoded sequencer. SAM's on-chip EPROM (448 words) is integrated with a microcoded sequencer consisting of Branch Control Logic, Stack, and Loop Counter. The eight general-purpose inputs, the Counter, the Stack, and the Pipeline Register feed the Branch Control Logic. The Branch Control Logic gives flexible multi-way microcode branch capability in a single clock, enhancing throughput beyond that of conventional controllers or sequencers.

SAM+PLUS development software offers high level microcode entry featuring a compact assortment of powerful instructions (OP-codes) allowing easy implementation of conditional branches, subroutine calls, multiple level for-next loops, and dispatch functions (branching to an externally specified address).

Functional Description

The SAM architecture is shown in Figure 1. The primary elements are the Microcode EPROM, 36-bit Pipeline Register, Branch Control Logic, 15 × 8-bit Stack, and 8-bit Loop Counter.

The Branch Control Logic generates the address of the next state and applies this address to the Microcode Memory. The outputs of the Microcode Memory represent the user-defined outputs and internal control values associated with the next state. On the leading edge of the clock these new values are clocked into the Pipeline Register and become the current state. The new values in the Pipeline Register—along with the Counter, Stack and Inputs—are used by the Branch Control Logic to generate the new next-state address.

Microcode EPROM and Pipeline Register

The Microcode EPROM is organized into 448, 36-bit words or locations, each of which can be viewed as a single state. 16 of these bits (the F-field) are available at device pins as user-defined outputs.

The other 20 bits are internal control signals that are divided into 4 fields: the 8-bit Q-field normally provides the next-state address; the 8-bit D-field is a general purpose field used either as a constant or as an alternative next-state address; the OP-field contains the instruction; and, the

Functional Description (Cont.)

E-field contains a single bit which enables or tri-states the device outputs.

As shown in Figure 2, the Microcode Memory is organized as 256 rows or addresses. Addresses 0 through 191 contain a single 36-bit word which is associated with the desired next-state. This state information will be clocked into the Pipeline Register on the next rising edge of the clock and the outputs will become valid one T_{CO} (clock to output delay) later.

Addresses 192–255, on the other hand, access four unique 36-bit words which correspond to four possible next states. (The extension .0, .1, .2, and .3 are used to distinguish those four states.) These 64 addresses are known as Multi-Way Branch locations and are used to perform single clock 4-way branches. Whenever the next-state address falls within the Multi-Way Branch locations, the Branch Control Logic will make the necessary 1-of-4 selection based on the next-state address and user-defined input conditions.

Figure 1. SAM448 Block Diagram

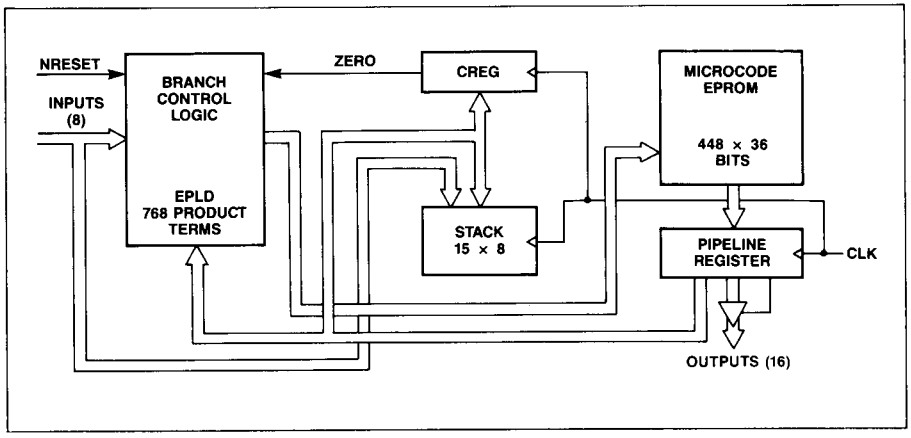


Figure 2. SAM Microcode Memory

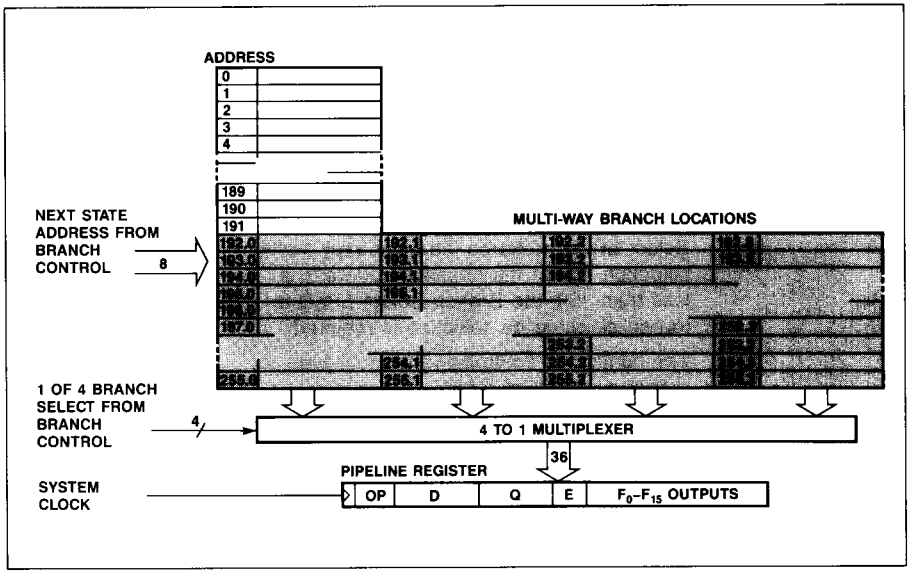
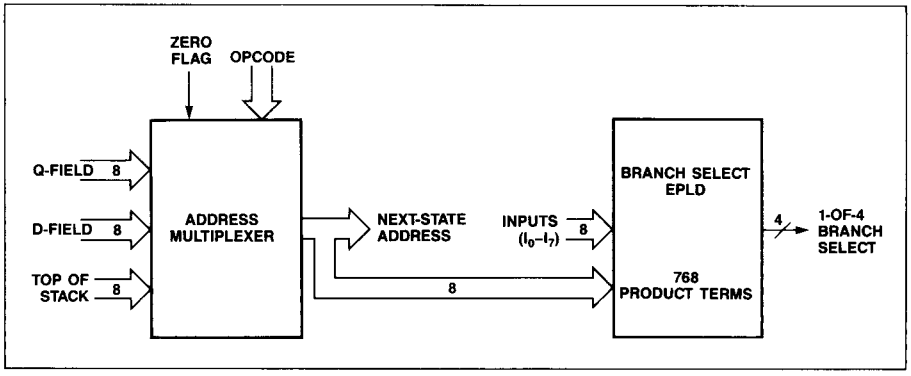


Figure 3. SAM Branch Control Logic



Branch Control Logic Block

At the heart of the high-performance sequencing ability of the SAM family is the Branch Control Logic. This block determines the next-state to be clocked into the Pipeline Register based on the current status of the Pipeline Register, the Counter, the Stack, and the eight input pins.

The Branch Control Logic is divided into two segments: the Address Multiplexer and the Branch Select EPLD.

The Address Multiplexer provides the next-state address to the Microcoded Memory. The next-state address can come from the Q-field, the D-field, or the Top-of-Stack. The selection between these three resources is based on the instruction in the Pipeline Register and the condition of the Zero Flag from the Counter.

The Branch Select EPLD is used to perform up to a 4-way branch based on user-defined input conditions. This block is a 768 product-term programmable logic device with 16 inputs and four outputs. When the next-state address falls within the multi-way branch block of memory (any address greater than 191) the Branch Select EPLD performs the necessary 1-of-4 selection. When the next-state address is less than 192, the Branch Select EPLD is turned off since no selection is required.

The conditions controlling the multi-way branch are defined by the user with a simple IF, THEN, ELSE format like the following:

```
      IF (cond3) THEN select 201.3
ELSEIF (cond2) THEN select 201.2
ELSEIF (cond1) THEN select 201.1
ELSE                select 201.0
```

The conditions are prioritized so that if the first condition is met (cond3), then microword 201.3 will be selected and clocked into the Pipeline Register regardless of the results of cond2 and cond1. If none of the three conditions are met, then the microword 201.0 will be clocked into the Pipeline Register.

The three conditional expressions are user defined and may contain any logical equation based on the inputs that can be reduced to four product-terms. For example, the expression

```
  I1 * /I2 * /I4
+I3 * /I4 * /I5 * /I6 * /I7
+I0
+I2 * /I4 * /I5
```

contains four product-terms and is a valid condition. There is a unique set of 12 product-terms for each of the 64 multi-way branch locations for a total of 768 product-terms. (See Figure 4.)

The SAM448 has been designed so that the number of available product-terms should never be the limiting factor on a design. Prioritization provides an effective product-term count of more than 12 per location. A trade-off between number of product-terms and number of possible branches can be made by simply placing identical state information in two locations as shown in Figure 5.

Figure 4. SAM Branch Logic for Address 192 Through 255

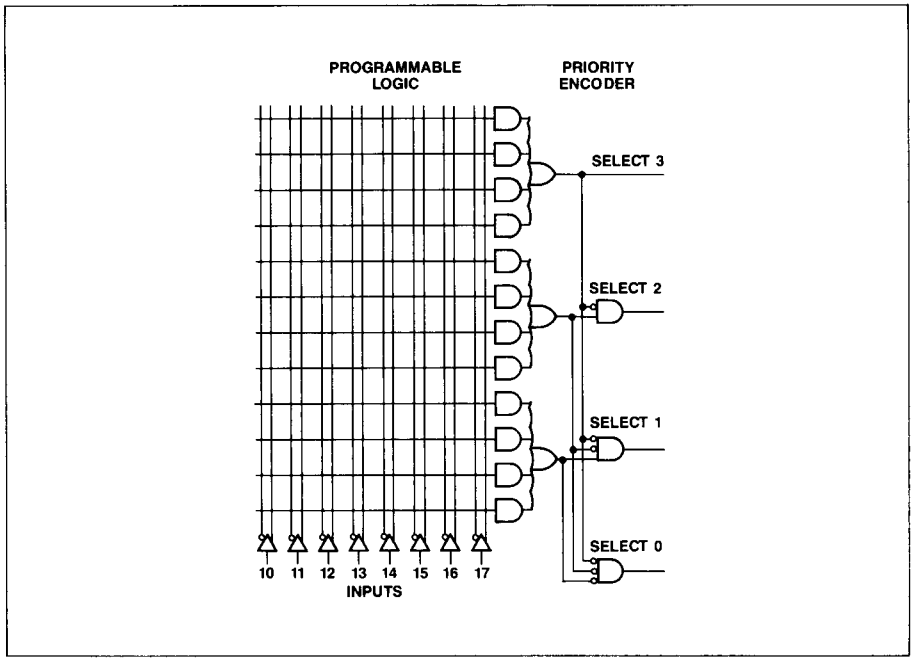
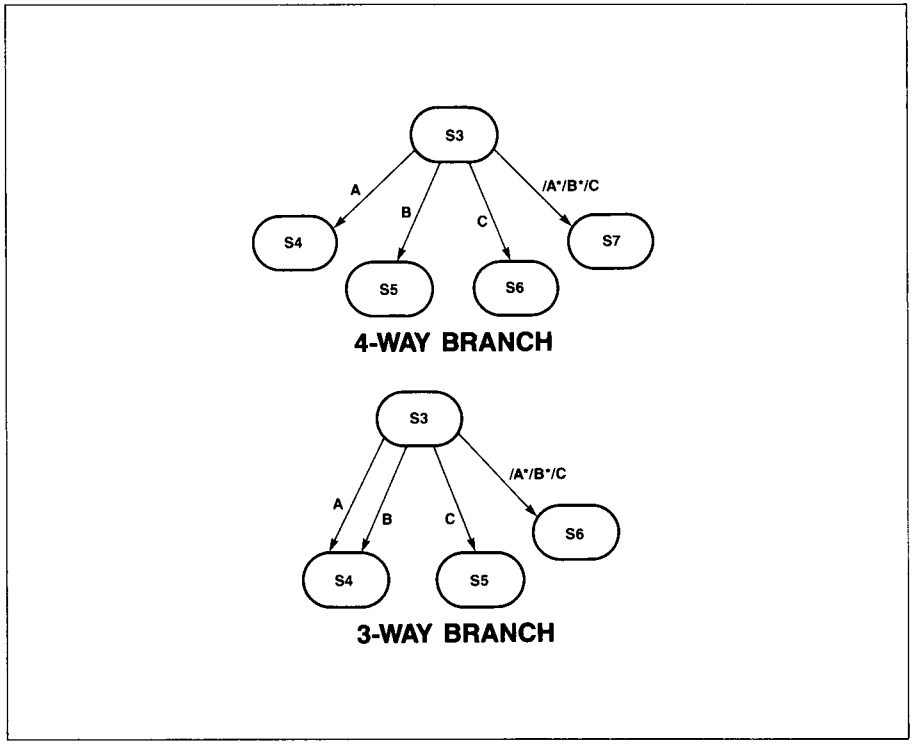


Figure 5. Multi-Way Branching



Functional Description (Cont.)

Stack

The Stack of the SAM448 is a Last In First Out (LIFO) arrangement consisting of 15 8-bit words. The Top-of-Stack may be used as the next-state address or popped into the Counter. Values may be pushed onto the stack either from the D-field in the Pipeline Register or from the Counter enabling efficient implementation of subroutines, nested loops, and other iterative structures. The eight input lines may also be pushed onto the stack to allow external address specification in a dispatch function or to externally load the counter.

The PUSHing or POPing of the stack occurs on the leading edge of the clock. The stack is "zero filled" so that a POP from an empty stack will return all eight bits set to zero. On the other hand, a push to an already full stack will write over the Top-of-Stack leaving the other 14 values unchanged.

Loop Counter

The SAM448 contains an 8-bit Loop Counter, referred to as the Count Register (CREG), which is useful for controlling timing loops and affecting a variety of branch operations. The CREG is a down counter and may be loaded directly from the D-field of the Pipeline Register or from the Top-of-Stack. The value of the CREG may be saved and restored by pushing and popping it to and from the Stack.

The CREG is loaded or decremented on the leading edge of the clock. It is designed so that it will not decrement once it reaches zero to prevent roll-over. A Zero Flag indicates when the counter has reached zero and is used with the LOOPNZ command to control program flow (see Instruction Set Description). Single instruction delay loops are easily constructed and, in combination with the Stack, nested loops or delays of arbitrary length may be generated.

Instruction Set

The instruction set of the SAM448 consists of a compact assortment of powerful commands. Assembly language constructs allow efficient implementation of multi-way branching, subroutines, nested for-next loops, and dispatch functions. The complete instruction set is described at the end of this data sheet. These instructions are only used with assembly language design entry and are automatically supplied when using the WSI State Machine Input Language (ASMILE).

Output Enable Control

Each microcode word contains an OE bit (the E-field) which enables the outputs when E = 1 and causes a high-impedance when E = 0. These bits are accessible through high-level constructs in the WSI Development Software. This capability allows the vertical cascading of SAM448 devices to increase the number of states.

nRESET Pin

The nRESET pin acts as a master reset for the SAM448 causing it to empty the Stack, clear the Counter, and load the microword found at address 0 into the Pipeline Register. The nRESET signal is useful for system reset or for synchronizing several SAMs that are cascaded vertically or horizontally.

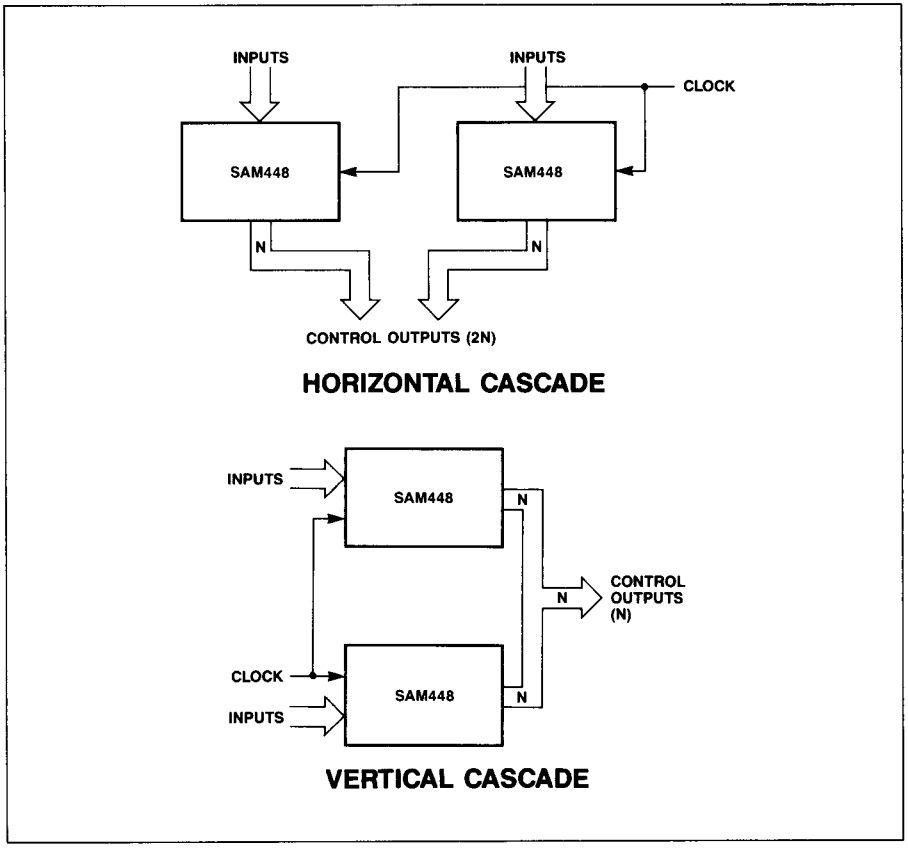
The nRESET signal must be held low for at least three clock rising edges to perform a valid clear. A nRESET of one clock rising edge causes the SAM448 to enter into a supervisor mode and a nRESET of two clock edges results in an undefined state.

The outputs of the boot address (00 Hex) will appear at the pins from the fourth clock edge after nRESET goes low, until the third clock edge after nRESET returns to high.

Horizontal and Vertical Cascading

Just as with memory and bit slice devices, the SAM devices can be cascaded to provide greater functionality. If an application requires more output lines, two or more SAMs can be cascaded horizontally. Likewise, if an application requires more states, two or more SAMs can be cascaded vertically. In either case, no speed penalty is incurred. Designs utilizing horizontal cascading are fully supported by the SAM+PLUS development software. Vertical cascading requires the designer to make certain tradeoffs to split the design.

Figure 6. SAM448 Cascading



Functional Testing

The SAM448 is fully functionally tested and guaranteed through complete testing of each programmable EPROM bit and all internal logic elements thus ensuring 100% programming yield.

The erasable nature of the SAM448 allows test programs to be used and then erased during early stages of production flow. This facility to use application-independent, general purpose tests is called generic testing and is unique among user-defined LSI logic devices. The devices also contain on board test circuitry to allow verification of function and AC specification once encapsulated in non-windowed packages.

Recommended Operating Conditions

Symbol	Parameter	Conditions	Min	Max	Unit
V_{CC}	Supply Voltage	Note 6	4.75 (4.5)	5.25 (5.5)	V
V_I	Input Voltage		0	V_{CC}	V
V_O	Output Voltage		0	V_{CC}	V
T_R	Input Rise Time (Note 6)			500 (100)	ns
T_F	Input Fall Time (Note 6)			500 (100)	ns

DC Operating Characteristics

V_{CC} = 5V ± 5%, 0°C to +70°C for Commercial
V_{CC} = 5V ± 10%, −40°C to +85°C for Industrial
V_{CC} = 5V ± 10%, −55°C to +125°C for Military

Symbol	Parameter	Conditions	Min	Typ	Max	Unit
V_{IH}	High Level Input Voltage		2.0		V_{CC} +0.3	V
V_{IL}	Low Level Input Voltage		−0.3		0.8	V
V_{OH}	High Level TTL Output Voltage	I_{OH} = −8 mA DC	2.4			V
V_{OH}	High Level CMOS Output Voltage	I_{OH} = −4 mA DC	3.84			V
V_{OL}	Low Level TTL Output Voltage	I_{OL} = 8 mA (4 mA) DC			0.45	V
I_I	Input Leakage Current	V_I = V_{CC} or GND			±10	µA
I_{OZ}	3-State Output Off-State Current	V_O = V_{CC} or GND			±10	µA
I_{CC1}	V_{CC} Supply Curent (Standby) (Note 6)	V_1 = V_{CC} or GND I_0 = 0 CLK = V_{CC}		30	65 (90)	mA
I_{CC2}	V_{CC} Supply Current (Active) (Note 6)	No Load 50% CLK f = 20 MHz		55	120 (170)	mA

Absolute Maximum Ratings

(See Design Recommendations)

Symbol	Parameter	Conditions	Min	Max	Unit
V_{CC}	Supply Voltage	With Respect to GND (Note 2)	−2.0	7.0	V
V_{PP}	Programming Supply Voltage		−2.0	14.0	V
V_I	DC Input Voltage		−2.0	7.0	V
I_{CCMAX}	DC V_{CC} or GND Current		−250	250	mA
I_{OUT}	DC Output Current, per Pin		−25	25	mA
P_D	Power Dissipation			1200	mW
T_{STG}	Storage Temperature	No Bias	−65	150	°C
T_{AMB}	Ambient Temperature	Under Bias	−10	85	°C

Capacitance

(Note 3)

Symbol	Parameter	Conditions	Typ	Unit
C_{IN}	Input Capacitance	V_{IN} = 0V f = 1.0 MHz	10	pF
C_{OUT}	Output Capacitance	V_{OUT} = 0V f = 1.0 MHz	15	pF
C_{CLK}	Clock Pin Capacitance	V_{IN} = 0V f = 1.0 MHz	10	pF
C_{RST}	nRESET Pin Capacitance		75	pF

AC Characteristics

V_{CC} = 5V ± 5%, 0°C to +70°C for Commercial
V_{CC} = 5V ± 10%, −40°C to +85°C for Industrial
V_{CC} = 5V ± 10%, −55°C to +125°C for Military (Note 7)

Symbol	Parameter	Conditions	SAM448-20		SAM448-25		Unit
			Min	Max	Min	Max	
f_{CYC}	Maximum Frequency	C_1 = 35 pF	20		25		MHz
t_{CYC}	Minimum Clock Cycle			50		40	ns
t_{SU}	Input Setup Time		22		20		ns
t_H	Input Hold Time		0		0	20	
t_{CO}	Clock to Output Delay	C_1 = 35 pF		22		20	ns
t_{CZ}	Clock to Output Disable or Enable			22			ns
t_{CL}	Minimum Clock Low Time		15		12		ns
t_{CH}	Minimum Clock High Time Data Valid		15		12		ns
t_{SUR}	nRESET Setup Time		18		18		ns
t_{HR}	nRESET Hold Time		5		5		ns

NOTES:
1. Typical values are for T_A = 25°C, V_{CC} = 5 V.
2. Minimum DC input is −0.3 V. During transitions, the inputs may undershoot to −2.0 V for periods less than 20 ns.
3. Capacitance measured at 25°C. Sample tested only.
4. If the nRESET is held low for more than 3 clock edges, then the outputs associated with the boot address (00 Hex) will remain at the pins until the third clock edge after rRESET goes high.
5. For 1.0 < V_1 < 3.8, the nRESET pin will source up to 200 µA.
6. Figures in ( ) pertain to military and industrial temperature versions.
7. The specifications noted above apply to military operating range devices. MIL-STD-883 compliant product specifications are provided in military product drawings available on request from WSI marketing at Tel. (510) 656-5400. These military product drawings should be used for the preparation of source control drawings.

Figure 7. Timing Waveforms

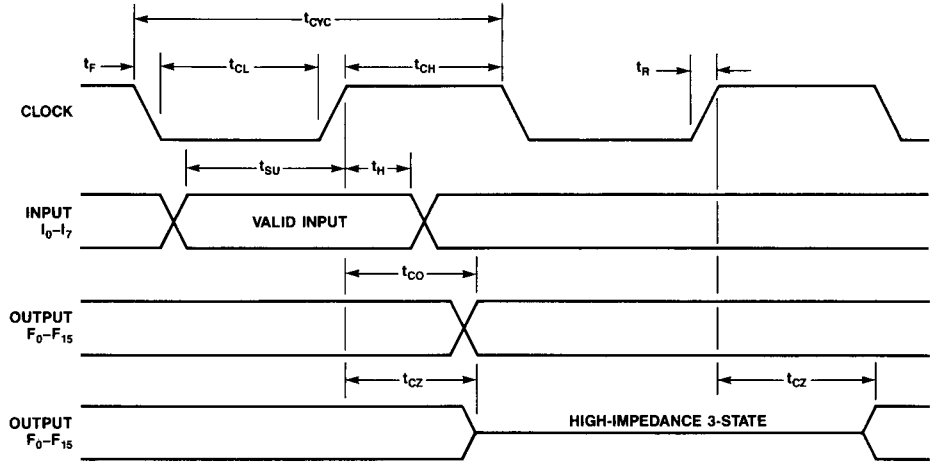
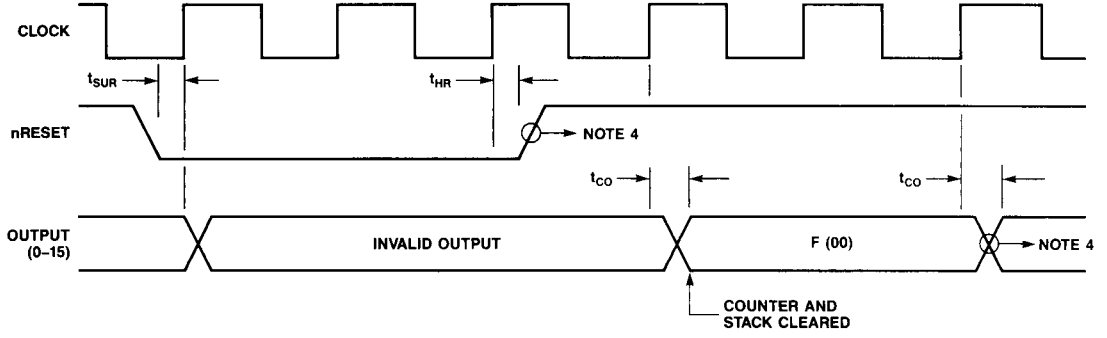


Figure 8. Reset Timing Waveforms



Design Security

The SAM448 contains a programmable design security feature that controls the access to the data programmed into the device. If this programmable feature is used, a proprietary design implemented in the device cannot be copied nor retrieved. This enables a high level of design control to be obtained since programmed data within EPROM cells in invisible. The bit that controls this function, along with all other program data, may be reset simply by erasing the device.

Design Recommendations

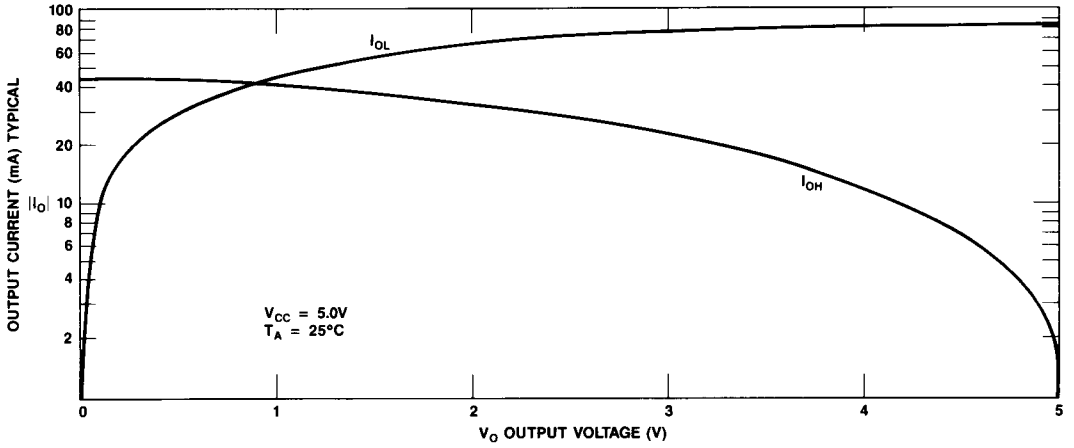
Operation of the SAM448 with conditions above those listed under "Absolute Maximum Ratings" may cause permanent damage to the device. This is a stress rating only, and functional operation of the device at these or any other conditions above those indicated in the operational sections of this data sheet is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability. These devices contain circuitry to protect the input against damage to high static voltages or electric fields; however, it is advised that normal precautions be taken to avoid application of any voltage higher than maximum rated voltages to this high-impedance circuit.

For proper operation, it is recommended that opaque labels be placed over the device window. Input and output pins must be constrained to the range GND $\leq$ (V_{IN} or V_{OUT}) $\leq$ V_{CC}. Unused inputs must always be tied to an appropriate logic level (e.g., either V_{CC} or GND). A power supply decoupling capacitor of at least 0.1 µF must be connected directly between the V_{CC} pin and GND.

When operating in noisy environments it is possible that a glitch on the nRESET pin one T_{SUR} before the clock edge could initiate a supervisor mode. To prevent this possibility, it is recommended to connect a capacitor of at least 0.1 µF from the nRESET input to ground.

All general purpose inputs to the SAM448 should be synchronized to be guaranteed to meet the setup time. Input transitions which occur less than one T_{SU} before the leading clock edge can cause the SAM448 to enter an undefined state.

Figure 9. Output Drive Current



Instruction Set Description

Following is a description of the instruction set available with the SAM448. These instructions can be used in conjunction with the Assembly Language entry to access the various features of the SAM448. They are automatically supplied when using the WSI State Machine Input Language (ASMILE).

In the following description label1 and label2 indicate arbitrary labels located in the assembly (.ASM) file. These labels will be converted by the software into the 8-bit address of that label. The parameter constant is any 8-bit number (0–255 Decimal, 0–FF Hex) representing an address, a mask, or a constant.

The instructions influence the control of the Stack, the Counter, and the Address Multiplexer. These effects are summarized in the Instruction Table. Throughout the examples it is assumed for simplicity that the destination labels do not lie within the Multi-Way Branch Block of memory so that branching based on inputs is not performed. It is valid, however, for any of these labels to lie within the Multi-Way Branch Block so that 4-way branching based on the inputs can be performed. See the MULTI-WAY BRANCH section at the end of this data sheet for more details.

The SAM+PLUS development system allows the designer to use the high level Assembly Language without worrying about the actual values that are placed in the various fields.

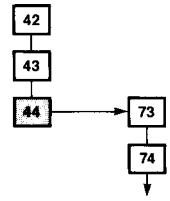
CONTINUE simply causes execution to continue with the next sequential instruction found in the Assembly Language file (.ASM).

CONTINUE

Instruction Set Description (Cont.)

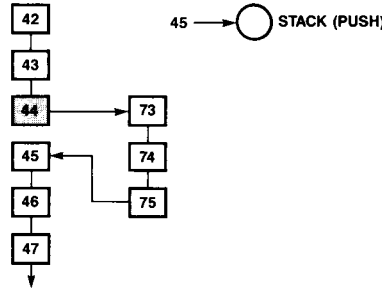
The JUMP instruction causes execution to branch to the indicated location. If address 44 contains the instruction 'JUMP label1,' then the next state will come from label1 which in this case is located at address 73.

JUMP label1



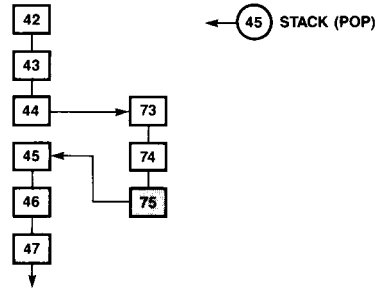
The CALL/RETURNTO instruction is typically used to call a subroutine. In general it will push the address of label2 onto the Stack and cause label1 to be the next-state address. Leaving the RETURNTO designation off will cause label2 to default to the next instruction in the .ASM file. In the example, address 44 contains the command 'CALL label1' where label1 is located at address 73. This causes the address of the following instruction, in this case 45, to be pushed onto the Stack, and the next state to come from address 73. The RETURN command at address 75 returns the execution to address 45.

CALL label1 RETURNTO label2



The RETURN command is used to return from a subroutine call or in general to cause the next-state address to come from the top of the Stack. In the example, the command at address 44 CALLed the subroutine at address 73 and PUSHed the value 45 onto the Stack. The RETURN command at address 75 will transfer execution to address 45 and POP that value off the Stack.

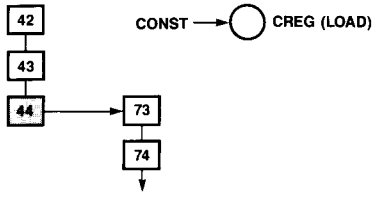
RETURN



Instruction Set Description (Cont.)

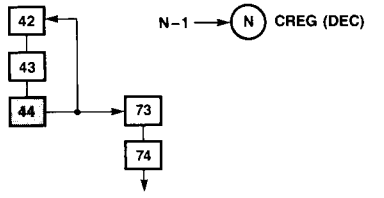
The LOAD Counter command loads the Counter with the value specified and transfers execution to label1. The LOADC command is typically used to initialize the Counter for a repetitive loop. In the example, address 44 has the command 'LOADC 73D GOTO label1' which causes the decimal value 73 to be loaded into the Counter and the next state to come from label1. In this case label1 is located at address 73. If the GOTO designation is left off label1 will default to the next instruction in the .ASM file.

LOADC constant GOTO label1



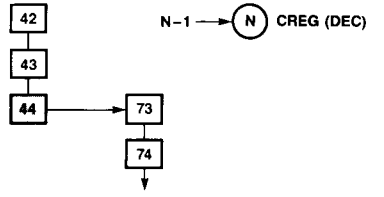
The LOOP on Non-Zero/ON ZERO goto command jumps to one of two addresses based on the value of the Zero Flag and decrements the Counter if not zero. This instruction is typically used to implement for-next loops. In the example, address 44 has the command 'LOOPNZ label1 ONZERO label2' where label1 is located at address 42 and label2 is located at address 73. If the Counter is not at zero then the next state will come from address 42 and the Counter will be decremented. If the Counter is already at zero then the instruction at address 73 will be executed and the Counter will stay at zero. If the ONZERO designation is left off, the default for label2 will be the next instruction in the .ASM file.

LOOPNZ label1 ONZERO label2



The DEcrement Counter on Non-Zero GOTO command will decrement the Counter if it is non-zero and jump to label1. In the example, address 44 has the command 'DECNZ GOTO label1' where label1 is located at address 73. The Counter is decremented and the next instruction comes from address 73. The default for label1 is the next instruction in the .ASM file.

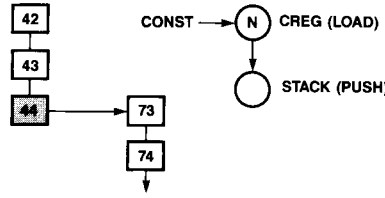
DECNZ GOTO label1



Instruction Set Description (Cont.)

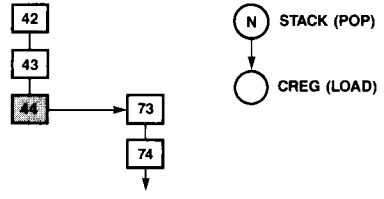
The PUSH Counter LOAD Counter command will push the current value of the Counter onto the Stack, load a constant into the Counter, and jump to label1. This instruction is useful for implementing nested for-next loops. In the example, the instruction at address 44 is 'PUSHLOADC 153D GOTO label1' where label1 is located at address 73. The value in the Counter will be pushed onto the Stack, the decimal value 153 will be loaded into the Counter, and the next instruction will come from address 73. The default for label1 is the next instruction in the .ASM file.

PUSHLOADC constant GOTO label1



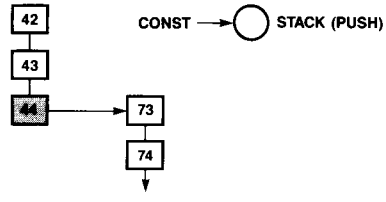
The POP Stack to Counter GOTO command will pop the top of Stack into the Counter and jump to label1. This command is typically used in conjunction with the PUSHLOADC to implement nested for-next loops. In the example, address 44 has the command 'POPC GOTO label1' where label1 is located at address 73. The default for label1 is the next instruction in the .ASM file.

POPC GOTO label1



The PUSH constant to Stack GOTO command will push the value constant onto the Stack and jump to label1. In the example, address 44 has the command 'PUSH 34D GOTO label1' where label1 is located at address 73. The decimal value 34 is pushed onto the Stack and the next state comes from address 73. The default for label1 is the next instruction in the .ASM file.

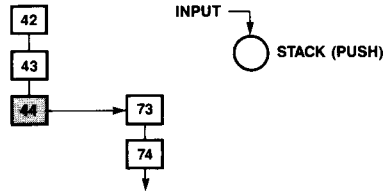
PUSH constant GOTO label1



Instruction Set Description (Cont.)

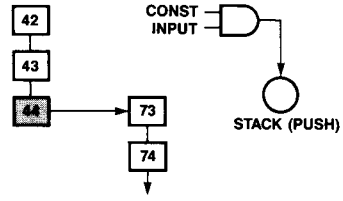
The PUSH Input GOTO command will push the eight inputs (I7–I0) onto the Stack. In the example address 44 has the instruction 'PUSHI GOTO label1' where label1 is located at address 73. At the leading edge of the clock the eight inputs are pushed onto the Stack. In a typical example, address 73 would have a RETURN instruction which would cause execution to jump to the address represented by the recently PUSHed input pins. This implements a dispatch function. The default for label1 will be the next instruction in the .ASM file. This instruction can also be used to load the Counter with an externally specified variable. In this case address 73 would have a POPC instruction.

PUSHI GOTO label1



The AND PUSH Input GOTO command is identical to the PUSHI command except the inputs are first bit-wise ANDed with a constant. This allows the masking of irrelevant inputs before PUSHing an address for a dispatch routine.

ANDPUSHI constant GOTO label1



The POP and XOR Stack to Counter GOTO command will pop the top of Stack, bitwise XOR it with a constant, load the result into the Counter, and jump to label1. In the example, address 44 has the command 'POPXORC 25D GOTO label1' where label1 is located at address 73. The top of Stack is POPed off the Stack, XORed with the decimal number 25, and loaded into the Counter. The next state comes from address 73. Since a XOR function does a comparison, this command can be used to compare the input to a constant and then branch based on the result with a LOOPNZ command. If the GOTO designation is left off the default for label1 will be the next instruction in the .ASM file.

POPXORC constant GOTO label1

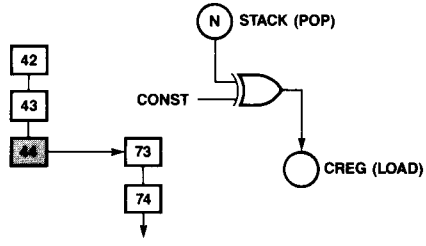


Figure 10. Instruction Set Summary

Instruction	Definition	Next-State Address	Stack	Counter
CONTINUE	Continue with Next Instruction	label1	None	HOLD
JUMP	Jump to a Label	label1	None	HOLD
CALL	Call Subroutine	label1	label2	HOLD
RETURN	Return From Subroutine	STACK	POP	HOLD
LOADC	Load CREG	label1	None	Constant
LOOPNZ	Loop/Dec. on Non-Zero	label 1 or 2	None	DECREMENT
DECNZ	Decrement CREG on Non-Zero	label1	None	DECREMENT
PUSHLOADC	Push CREG to Stack and Load CREG	label1	CREG	Constant
POPC	Pop Stack to CREG	label1	POP	STACK
PUSH	Push Constant to Stack	label1	Constant	HOLD
PUSHI	Push Inputs to Stack	label1	INPUTS	HOLD
ANDPUSHI	Push Masked Inputs to Stack	label1	INP * const	HOLD
POPXORC	XOR Stack with Constant and Send Result to CREG	label1	POP	STACK ⊕ Constant

NOTE: The value label1 is placed in the Q-field. The values label2 and constant are placed in the D-field.

Multi-Way Branching

The multi-way branching capability can be super imposed upon the instruction set providing another dimension of capability. Figure 11 shows how this translates into the flow diagrams. If location 44 had the instruction 'JUMP label1' where label1 is located at address 201, then the next-state would come from address 201. But address 201 is within the Multi-Way Branch Block so the Branch Select EPLD must decide which of the four words to send to the pipeline register. This selection is based on user-defined functions of the inputs.

Similarly, location 44 could contain any of the 13 available commands so that the multi-way branch capability can enhance each instruction. If location 44 was a CALL to a subroutine, then address 201 could contain the starting instruction for 4 unique subroutines. The actual routine executed would depend on the condition of the inputs as defined by the user.

The actual Assembly Language code required to implement this example is as follows:

44D: [Output Spec] CALL label1;

201D: IF cond1 THEN [out 1] JUMP 102D;
ELSEIF cond2 THEN [out 2] JUMP 73D;
ELSEIF cond3 THEN [out 3] JUMP 53D;
ELSE [out 4] JUMP 34D;

Figure 11. Jump to a Multi-Way Branch Address

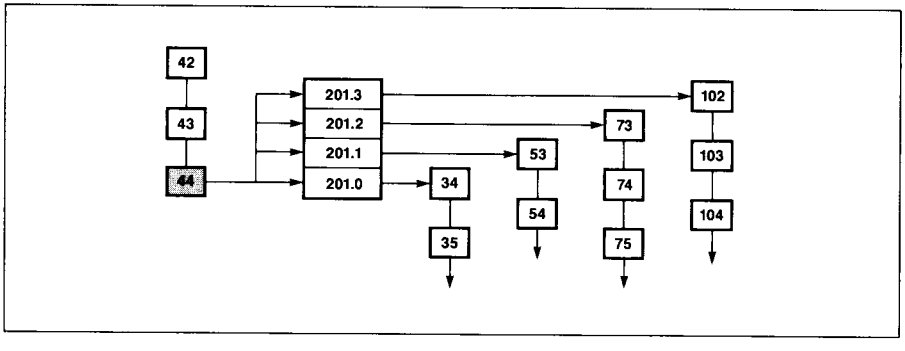
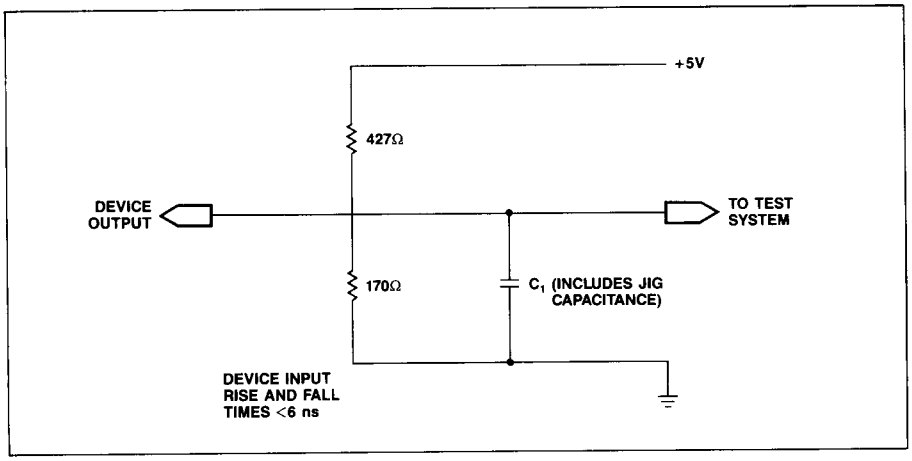


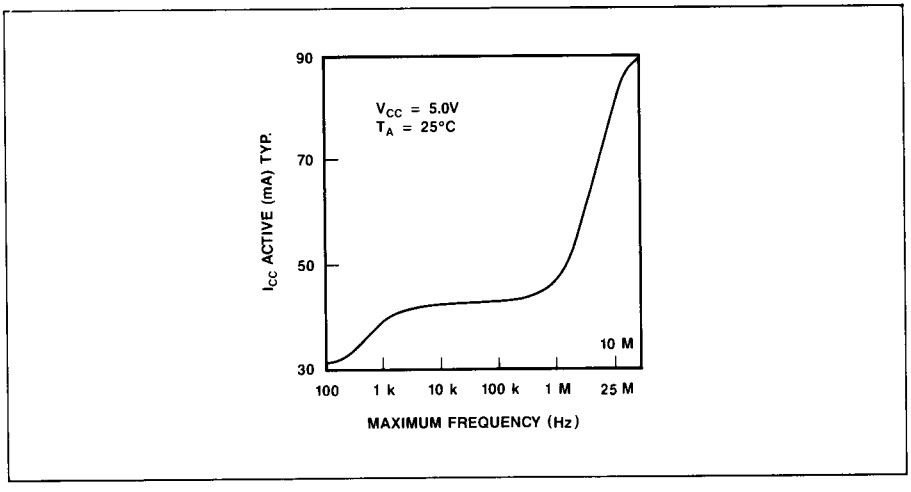
Figure 12. AC Test Conditions



Power supply transients can affect AC measurements; simultaneous transitions of multiple outputs should be avoided for accurate measurement. Do not attempt to perform threshold tests under AC conditions. Large amplitude, fast ground current transients normally occur as the device outputs discharge the load capacitances. These transients flowing through the parasitic inductance between the device ground pin and the test system ground can create significant reductions in observable input noise immunity.

2

Figure 13. I_{CC} vs. F_{MAX}



Product Grades

Application	Temperature Range	Marking Designator
Commercial	0°C to +70°C	
Industrial	−40°C to +85°C	I
Military	−55°C to +125°C	M
MIL-STD-883C, Class B	−55°C to +125°C	MB