TEXAS INSTRUMENTS

9900

SBP9989

Advance 16 Bit I^2L Microprocessor

MICROPROCESSOR SERIES™

Data Manual

Texas Instruments . . . the inventor of the integrated circuit, the microprocessor, and the microcomputer . . . is a leader in the technology that provides electronic sensing, control, computing, and even speech. For additional information on TI's high-technology products contact:

Military Products Marketing Group
Texas Instruments Incorporated
P.O. Box 6448, MS 3030
Midland, Texas 79701
(915) 685-7142

or

Customer Response Center
Texas Instruments Incorporated
P.O. Box 225012, MS 57
Dallas, Texas 75265
(214) 995-6611

IMPORTANT NOTICES

Texas Instruments reserves the right to make changes at any time in order to improve design and to supply the best product possible.

TI cannot assume any responsibility for any circuits shown or represent that they are free from patent infringement

SBP9989 MICROPROCESSOR
TABLE OF CONTENTS

INTRODUCTION

The 9900 Family

The 9900 Family is a compatible set of microprocessors, microcomputers, microcomputer modules, and minicomputers. It is supported with peripheral devices, development systems, and software and provides the designer with a system solution that has built-in protection against technological obsolescence. The family features true software compatibility, I/O bus compatibility and price/performance ratios which encompass a wide range of applications. The family is designed with a unique flexible architecture to allow technological changes to be easily incorporated while minimizing the impact these changes have on an overall system design

The SBP9989 I^{2}L 16-Bit Microprocessor

The SBP9989 is a second-generation, bipolar 16 bit microprocessor offering twice the performance of the original SBP9900A Implemented in TI's Advanced I^{2}L Technology, the SBP9989 combines environmental ruggedness and inherent reliability with a 16-bit word length, an advanced memory-to-memory architecture, and a full minicomputer instruction set to extend the end application reach of the Texas Instruments 9900 microprocessor family into those applications requiring efficient, stable, reliable performance in severe operating conditions

The instruction set of the SBP9989 includes the capabilities offered by full minicomputers and is a superset of the 9900 instruction repertoire The SBP9989 can be used to upgrade existing SBP9900A systems or to implement new system designs requiring the increased through-put and doubled memory capability

KEY FEATURES

The SBP9989 is downward-compatible with all of TI's existing microcomputer and minicomputer products and employs the advanced, memory-to-memory architecture that ensures optimal performance in the structured, I/O-intensive applications of the 80's.

- Memory-to-memory architecture

- 73 basic instructions include all 69 instructions of the SBP9900A plus Signed Multiply, Signed Divide, Load WP, and Load ST The signed multiply and divide instructions allow significant improvements in system through-put in numerical applications, while load workspace register and load status register are essential to the efficient implementation of advanced operating systems.

- User extension to the basic instruction set allows undefined op codes to be assigned meanings during system design. The processor detects undefined op codes in an instruction stream and allows either software interpretation and execution of the code or hardware execution in special, user-designed "attached processors". Coordination between the SBP9989 and its attached processors is achieved via a new input signal to the microprocessor designated $\overline{\text{XIPP}}$ (External Instruction Processor Present).

- Direct access to 128 kilobytes of memory from the SBP9989 is provided by a new output signal designated $\overline{\text{MPEN}}$ (Memory Map Enable) which can be used directly as an additional address bit from the processor. This bit is represented in the processor Status Register as Status Bit 8, which may be manipulated by the user to allow access to two 64-kilobyte pages of memory. $\overline{\text{MPEN}}$ can also be used with the SN54LS610 Memory Mapper to allow access to as much as 16 megabytes of memory. All traps cause Status Bit 8 to be forced to zero during a context switch, ensuring consistent interrupt operation and full software and hardware compatibility with other TI products.

- Multiprocessor system features allow coordination between several processing elements that must share memory and other resources. These features include:

  MPILCK (Multiprocessor Interlock), a new output signal that allows a processor to secure and hold a system resource against possible access contention with other processing elements.

  INTACK (Interrupt Acknowledge), a new output signal which allows the SBP9989 to acknowledge the presence of an interrupt during those times when it may not have control of the system resources.

  $\overline{\text{XIPP}}$ (Extended Instruction Processor Present), a new input signal that establishes a protocol for orderly transfer of bus control between host and slave processors that share memory resources.

  LOAD WP and LOAD ST (Load Workspace Pointer and Load Status Register) instructions that allow the SBP9989 to capture a complete software context from an external source.

- Fully static design allows the SBP9989 to be clocked up to 4 4 MHz, or single stepped The TTL-compatible I/O permits any standard logic and memory devices to be used
- Simplified clock requirements consist of a single-phase clock with a 50% duty cycle
- Improved microcode within the processor enhances the efficiency of the SBP9989 over its predecessors. Micro cycles were removed from more than half of the instructions, resulting in a 15% to 20% improvement in operating efficiency.
- Fully implemented 16-bit status register with Arithmetic Overflow Interrupt trap
- Improved external instruction, utilizing five address lines, provides a total of 160 forms available to the user.
- Improved HOLD and WAIT interfaces.
- Implemented in Advanced I^2L . . a proven bipolar technology for high-reliability applications.

The single most important feature of the SBP9989, and all of TI's microprocessors, is its memory-to-memory concept. The user has access to an unlimited number of effective registers, and may completely change register context (an operation equivalent to sixteen push and sixteen pop stack operations in a conventional registered architecture) in just five memory cycle times The ability to change register content rapidly becomes of prime importance in systems that rely on multiprocessor architectures and high-level languages for efficient software . trends which will gain momentum as embedded computer applications become increasingly more complex

DEVELOPMENT SUPPORT

AMPL System Advanced Microprocessor Prototyping Lab

The AMPL System is a complete set of software and hardware tools that maximize software productivity for the 9900 family It includes a video display terminal, multiuser hard disk or floppy diskette options, and extensive software Programs can be edited, assembled, loaded, and executed with easy self-prompting commands.

The AMPL System supports software development as well as in-circuit emulation for existing 9900 family CPU's. The logic state trace capability features interactive on-line control and analysis to provide fast data reduction, and programmable emulation control based on the result of this analysis. The high-level language has designed-in features to simplify orientation for the new user while providing extensive flexibility and support for the experienced user.

PROM programming implements target-system memory in PROM and EPROM while the AMPL interactive process makes it easy to identify and implement needed design changes.

Transportable Assembler

The SBP9989 is supported by a transportable general assembler (TI part numberTMAM4015). This assembler allows the use of symbolic instructions and assembler directives supporting the full instruction set. The 9900 family instruction set is composed of a base set of 69 instructions plus extensions peculiar to each CPU which provide the input and output manipulation comparison of words and bytes, and ASCII-character data This product can be executed in TI 990, IBM, or DEC environments

I^2L TECHNOLOGY

I^2L is an integrated-injection circuit logic where current steering is used to control gate switching levels A lateral PNP transistor is used as an injector to supply base current to the vertical NPN transistor as shown in the figure. The NPN actually serves as the gate that is controlled by placing either a high or low voltage on the base that steers the injector current into or out of the NPN base

I^2L technology provides inherent advantages to the user when compared to other technologies.

o **–55°C to +125°C Temperature Range**

Circuits are designed to operate over the full military temperature range of –55°C to +125°C rather than just being selected by screening. This provides added design margin that enhances system reliability.

o **Low Power Consumption**

The SBP9989 dissipates less than 0.75 watts, depending upon the state of the I/O's.

o **Radiation Hardness**

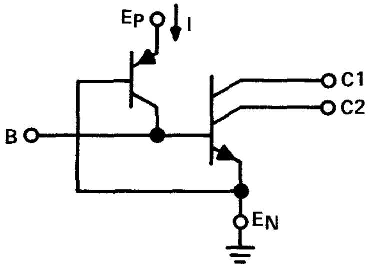
Also inherent with I^2L as used on the SBP9989 is tolerance to radiation Its tolerance to transient upset is among the best observed on LSI parts while the simplicity of the process eliminates latch-up.

o **Low Internal Stress Voltage**

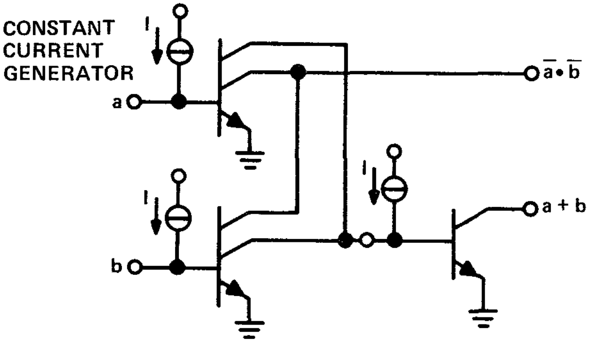
High internal voltages have been eliminated with I^2L, thereby assuring further gains in reliability. The 5,000 internal gates of the SBP9989 are stressed only with the V_{INJ}, which is typically 1.2 volts. Only the inputs and outputs are exposed to high stress voltages.

o **High Reliability**

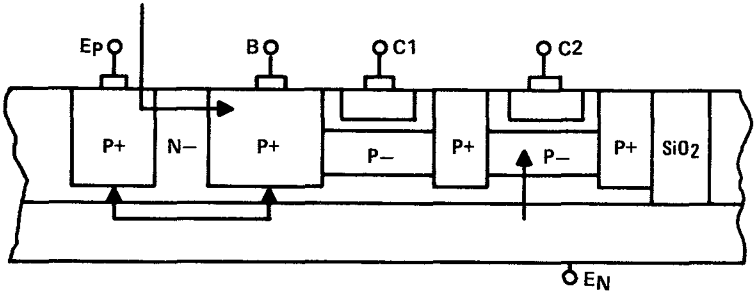
The SBP9989 has been designed, fabricated and 100% screened with processes to assure the highest levels of reliability. Each part is individually identified to assure traceability



I^2L CIRCUIT SCHEMATIC



I²L LOGIC GATES



I²L DEVICE STRUCTURE

COMPATIBILITY WITH SBP9900A SYSTEMS

The SBP9989 is pin for pin compatible with the SBP9900A except for 4 new I/O controls $\overline{\text{(MPEN}}$, INTACK, MPILCK and $\overline{\text{XIPP}}$) assigned to pins that were not used on the original device The modifications required to plug an SBP9989 into an existing SBP9900A socket are listed below

Hardware

- o Deactivating $\overline{\text{XIPP}}$ (tie Pin 58 to V_{CC} through a pull up resistor)
- o Provision for an increase in injector voltage to 1 25 V and a reduction in injector current to 400 mA
- o Verifying that Pin 25 $\overline{\text{(MPEN)}}$, Pin 37 (INTACK), and Pin 39 (MPILCK) are open or grounded
- o Interrupt lines sampled during hold and non hold states
- o Verifying fan out compatibility (I_{OL} = 20 mA for SBP9900A, I_{OL} = 16 mA for SBP9989)
- o The READY input signal (pin 62) needs to be active during CRU transactions to avoid wait states
- o Automatic wait states are typically generated with the SBP9900A by tying the wait output signal back into the READY input. However, unlike the SBP9900A, the SBP9989 will continue to generate wait signals during a HOLD operation To eliminate the unwanted wait signals during HOLD operations, the HOLDA and WAIT output signal lines can be OR-ed to drive the READY input line

Software

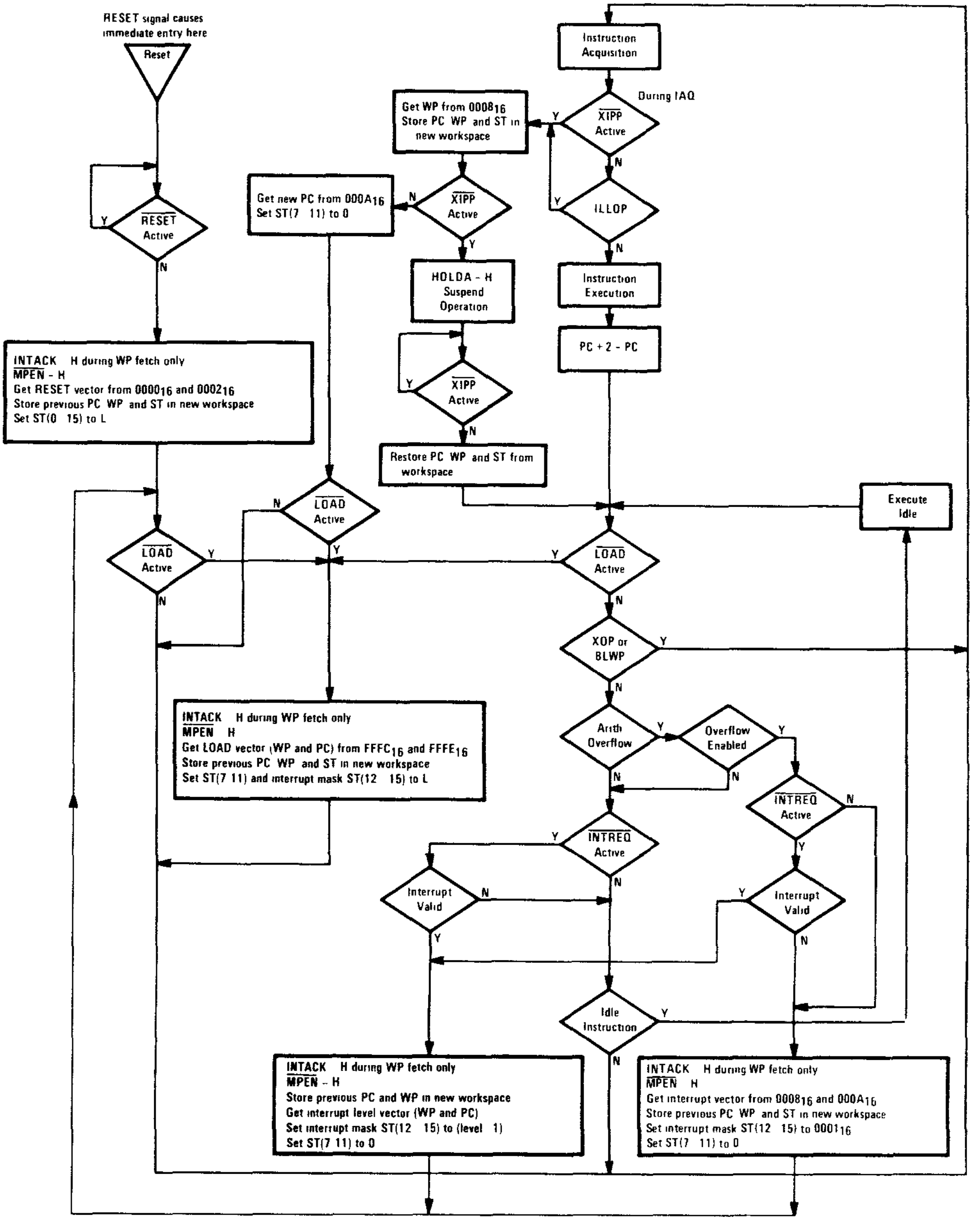
- o Although software written for the SBP9900A can be executed by the SBP9989, a reduction of 15% to 20% in execution time should be anticipated Software timing loops will need new time constants to compensate for the reduced execution times
- o Any unused op codes used as NOP's in the SBP9900A software will cause the SBP9989 to execute a level 2 interrupt trap
- o Status bit 10 set to 1 will enable the arithmetic overflow detection trap

DEVICES FOR MILITARY TEMPERATURE APPLICATIONS

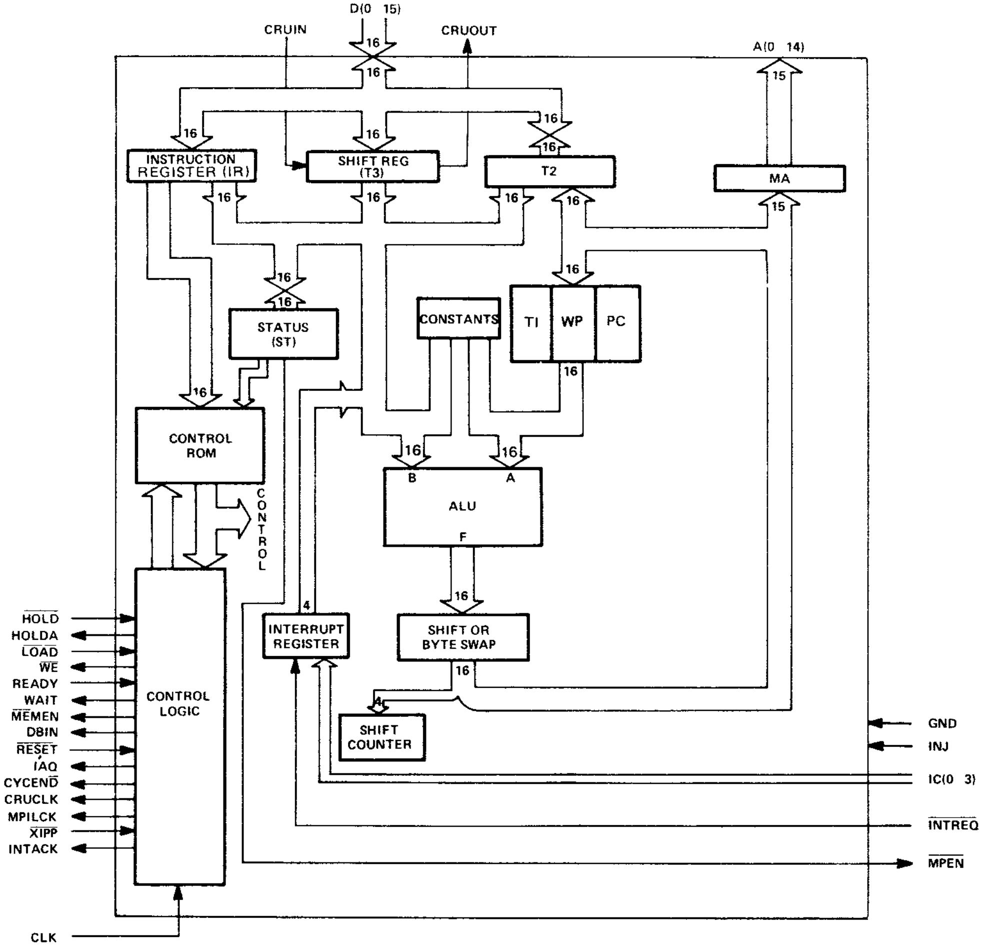
The SBP9989 is a member of the ever growing family of military microprocessor components

SBP9989	16 Bit Advanced Hi Rel Microprocessor
SBP9900A	16-Bit Hi-Rel Microprocessor
SBP9965	Peripheral Interface Adapter
SBP9901	Programmable Systems Interface (in design)
SNJ54LS244	Octal Buffer Line Driver/Line Receiver
SNJ54LS373	Octal D-Type Transparent Latch
SNJ54LS374	Octal D Type Edge-Triggered Flip-Flop
SNJ54LS610	Memory Mapper (future product)
SNJ54LS629	Dual Voltage Controlled Oscillator
SNJ54LS630	16-Bit Parallel Error Detection and Correction Unit
SNJ54LS644	Octal Bus Transceiver
SNJ54LS645	Octal Bus Transceiver
SNJ54LS673	16-Bit Shift Register
SNJ54LS674	16 Bit Shift Register
SNJ54S189	64 Bit (16 x 4) RAM
JBP24S10	1024-Bit (256 x 4) PROM
JBP18S030	256 Bit (32 x 8) PROM
JBP28S46	4096 Bit (512 x 8) PROM
JBP28L86	8192-Bit (1024 x 8) PROM
JBP28S86	8192-Bit (1024 x 8) PROM

FLOW CHART

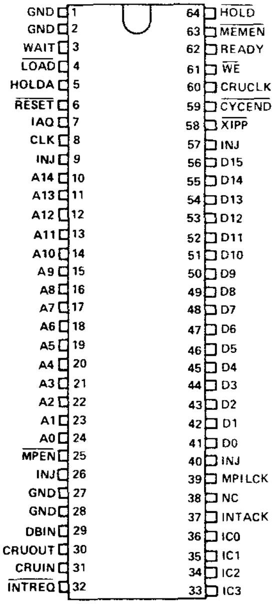

RESET signal causes immediate entry here
Reset
RESET Active
Y
N
INTACK H during WP fetch only
MPEN - H
Get RESET vector from 0008₁₆ and 0002₁₆
Store previous PC WP and ST in new workspace
Set ST(0 15) to L
LOAD Active
Y
N
Instruction Acquisition
During IAQ
XIPP Active
Y
N
Get WP from 0008₁₆
Store PC WP and ST in new workspace
ILLOP
Y
N
XIPP Active
N
Y
Get new PC from 000A₁₆
Set ST(7 11) to 0
HOLDA - H
Suspend Operation
XIPP Active
Y
N
Restore PC WP and ST from workspace
LOAD Active
N
Y
Instruction Execution
PC + 2 - PC
Execute Idle
LOAD Active
Y
N
XOP or BLWP
Y
N
INTACK H during WP fetch only
MPEN H
Get LOAD vector (WP and PC) from FFFC₁₆ and FFFE₁₆
Store previous PC WP and ST in new workspace
Set ST(7 11) and interrupt mask ST(12 15) to L
Arith Overflow
Y
N
Overflow Enabled
Y
N
INTREQ Active
N
Y
INTREQ Active
Y
N
Interrupt Valid
N
Y
Interrupt Valid
Y
N
Idle Instruction
Y
N
INTACK H during WP fetch only
MPEN - H
Store previous PC and WP in new workspace
Get interrupt level vector (WP and PC)
Set interrupt mask ST(12 15) to (level 1)
Set ST(7 11) to 0
INTACK H during WP fetch only
MPEN H
Get interrupt vector from 0008₁₆ and 000A₁₆
Store previous PC WP and ST in new workspace
Set interrupt mask ST(12 15) to 0001₁₆
Set ST(7 11) to 0

BLOCK DIAGRAM

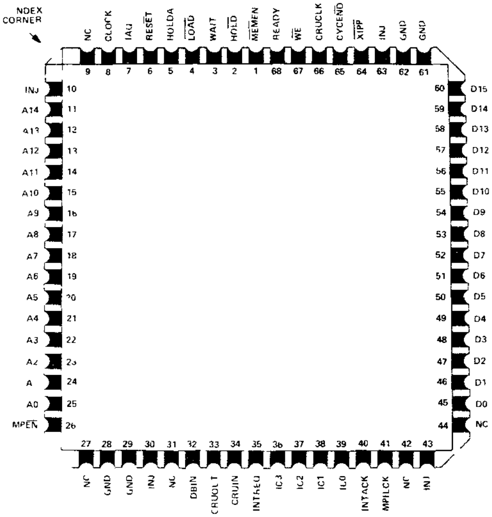


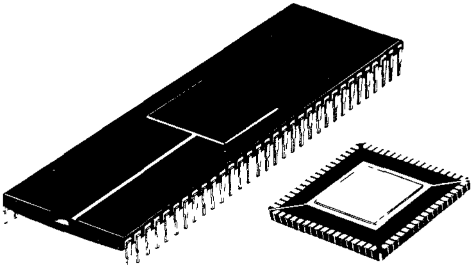
PIN ASSIGNMENTS

SBP9989NJD
64-PIN CERAMIC SIDE BRAZED PACKAGE
(TOP VIEW)



SBP9989NFD
68 TERMINAL CERAMIC CHIP CARRIER
(TOP VIEW)





PIN ASSIGNMENTS AND FUNCTIONS (PIN NUMBERS ARE FOR 64-PIN DIP)

SIGNATURE	PIN #	I/O	DESCRIPTION
			ADDRESS BUS
A0 (MSB) ↓ ↓ A14 (LSB)	24 ↓ 10	OUT*	A0-A14 comprise the address bus This open-collector bus provides the memory address to the memory system when $\overline{\text{MEMEN}}$ is active and CRU I/O bit addresses to the I/O system when $\overline{\text{MEMEN}}$ is inactive and DBIN is active
$\overline{\text{MPEN}}$	25	OUT*	MEMORY MAP ENABLE $\overline{\text{MPEN}}$ represents the inverted value of Status Register Bit 8 (ST8) $\overline{\text{MPEN}}$ can be changed by any instruction (i e , LST, etc) affecting ST8 and will be set to 1 during SBP9989 trap addressing, namely interrupts, $\overline{\text{LOAD}}$, $\overline{\text{RESET}}$, XOP and ILLOP, $\overline{\text{MPEN}}$ may be used to allow memory expansion to 64 kilowords
			DATA BUS
D0 (MSB) ↓ ↓ D15 (LSB)	41 ↓ 56	I/O*	D0-D15 comprise the bidirectional, open-collector data bus This bus transfers memory data to (when writing) and from (when reading) the external memory system when $\overline{\text{MEMEN}}$ is active
			POWER SUPPLY
INJ	9		Injector-Supply Current
INJ	26		Injector-Supply Current
INJ	40		Injector-Supply Current
INJ	57		Injector-Supply Current
GND	1		Ground
GND	2		Ground
GND	27		Ground
GND	28		Ground
			CLOCK
CLK	8	IN	Single-phase clock input
			BUS CONTROL
$\overline{\text{MEMEN}}$	63	OUT*	MEMORY ENABLE When active (low), $\overline{\text{MEMEN}}$ indicates that the address bus contains a valid memory address
DBIN	29	OUT*	DATA BUS IN When activated (high) by the SBP9989 during $\overline{\text{MEMEN}}$, DBIN indicates that the SBP9989 has disabled its output buffers to allow the memory system to place memory read data on the bus The SBP9989 will also activate DBIN during all CRU operations and during the execution of the five external instructions. In all other cases except when HOLDA is active, the SBP9989 will maintain DBIN at a low logic level
$\overline{\text{WE}}$	61	OUT*	WRITE ENABLE When active (low), $\overline{\text{WE}}$ indicates that the SBP9989 data bus is outputting data to be written into memory

*When HOLDA is active, these terminals are high

SIGNATURE	PIN#	I/O	DESCRIPTION
			COMMUNICATION REGISTER UNIT (CRU)
CRUCLK	60	OUT	CRU CLOCK When active (high), CRUCLK indicates to the external logic the presence of output data on CRUOUT or the presence of an encoded external instruction on A0 A2
CRUIN	31	IN	CRU DATA IN CRUIN receives input data from the external interface logic When the SBP9989 executes a STCR or TB instruction, it samples CRUIN for the level of the CRU bit specified by the address bus (A3 A14)
CRUOUT	30	OUT	CRU DATA OUT CRUOUT outputs serial data when the SBP9989 executes a LDCR, SBZ, or SBO instruction The data on CRUOUT should be sampled by the external interface logic when CRUCLK goes active
			INTERRUPT CONTROL
$\overline{\text{INTREQ}}$	32	IN	INTERRUPT REQUEST When active (low), $\overline{\text{INTREQ}}$ indicates that an external interrupt is requesting service If $\overline{\text{INTREQ}}$ is active the SBP9989 loads the data on the interrupt code input lines IC0 IC3 into the internal interrupt-code storage register The code is then compared to the interrupt mask bits of the status register If the interrupt code is equal to or less than Status Register Bits 12 15 (equal or higher priority than the previous enabled interrupt level), the SBP9989 initiates the interrupt sequence If the comparison fails, the SBP9989 ignores the interrupt request In that case, $\overline{\text{INTREQ}}$ should be held active The SBP9989 will continue to sample IC0 IC3 until the program enables a sufficiently low interrupt level to accept the requesting interrupt
ICO (MSB) ↓ ↓ IC3 (LSB)	36 ↓ 33	IN	INTERRUPT CODES ICO (MSB) IC3 (LSB), indicating an interrupt identity code, are sampled by the SBP9989 when $\overline{\text{INTREQ}}$ is active (low) When IC0-IC3 are LLLL, the highest-priority external interrupt is requesting service, when HHHH, the lowest-priority external interrupt is requesting service
INTACK	37	OUT	INTERRUPT ACKNOWLEDGE When active (high) during non-hold states, INTACK indicates the SBP9989 has initiated a trap sequence caused by the receipt of a valid interrupt, $\overline{\text{LOAD}}$ or $\overline{\text{RESET}}$ INTACK shall be activated in the trap sequence while the SBP9989 is obtaining the new WP value from memory An external device may determine which function or interrupt level is being serviced by monitoring the address bus during the INTACK time When the SBP9989 is in a hold state (caused by activation of $\overline{\text{XIPP}}$ or $\overline{\text{HOLD}}$) INTACK indicates SBP9989 has received a valid interrupt (level is less than value of interrupt mask), a $\overline{\text{LOAD}}$ or $\overline{\text{RESET}}$ INTACK will remain valid (high) until the SBP9989 leaves a hold state ($\overline{\text{HOLD}}$ or $\overline{\text{XIPP}}$ released) or until the signal requesting interrupt is released

SIGNATURE	PIN=	I/O	DESCRIPTION
			MEMORY CONTROL
$\overline{\text{HOLD}}$	64	IN	HOLD When active (low), $\overline{\text{HOLD}}$ indicates to the SBP9989 that an external controller (e g , DMA device) desires to use the memory bus for direct memory data transfers In response, the SBP9989 enters the hold state after completion of its present cycle (memory or nonmemory) The SBP9989 then asserts HOLDA and allows its address bus, $\overline{\text{MPEN}}$, data bus, $\overline{\text{MEMEN}}$, $\overline{\text{WE}}$, DBIN, IAQ and $\overline{\text{CYCEND}}$ to be pulled to the high logic state When $\overline{\text{HOLD}}$ is deactivated, the SBP9989 reassumes bus control and continues operation by resuming execution of the suspended instruction
HOLDA	5	OUT	HOLD ACKNOWLEDGE When active (high) HOLDA indicates that the SBP9989 is in a hold state and that its address bus, $\overline{\text{MPEN}}$, data bus, $\overline{\text{MEMEN}}$, $\overline{\text{WE}}$, DBIN, IAQ, and $\overline{\text{CYCEND}}$ are pulled to the high state. The SBP9989 will enter a hold state in response to the activation of $\overline{\text{HOLD}}$ or $\overline{\text{XIPP}}$ (during the execution of an ILLOP or XOP instruction)
READY	62	IN	READY When active (high) READY indicates that the memory (for memory operations) or CRU device (for CRU operations) will be ready to read or write during the next clock cycle. When READY is not active (low), the SBP9989 enters a wait state and suspends internal operations until the memory system or CRU device activates READY.
WAIT	3	OUT	WAIT When active (high), WAIT indicates the SBP9989 has entered a wait state in response to a not READY condition from a memory system or a CRU device
			TIMING AND CONTROL
IAQ	7	OUT*	INSTRUCTION ACQUISITION IAQ is activated (high) by the SBP9989 during any SBP9989 initiated instruction acquisition memory cycle Consequently, IAQ may be used by an external device as an indication of when to sample the memory data bus to obtain instruction operations code data
$\overline{\text{CYCEND}}$	59	OUT*	END OF CYCLE When active (low), $\overline{\text{CYCEND}}$ indicates that the SBP9989 will initiate a new microinstruction cycle on the next low-to-high transition of the clock
MPILCK	39	OUT	MULTIPROCESSOR INTERLOCK When active (high) MPILCK indicates the SBP9989 is performing the operations associated with operand transfer and manipulation for the ABS instruction MPILCK shall be activated by the SBP9989 during any ABS instruction upon initiation of the operand read operation and remain active until the completion of the instruction (i e , MPILCK remains active for the duration of the SBP9989 read-modify-write operation cycle for the ABS instruction) Consequently, MPILCK may be used in the implementation of a nonseparable test and set capability $\overline{\text{HOLD}}$ is sampled during MPILCK activation, so MPILCK can be used to control assertion of $\overline{\text{HOLD}}$

*When HOLDA is active, these terminals are high

SIGNATURE	PIN#	I/O	DESCRIPTION
$\overline{\text{XIPP}}$	58	IN	EXTENDED INSTRUCTION PROCESSOR PRESENT When activated (low) by an external device (an extended instruction processor, XIP) upon detection of the acquisition of an SBP9989 undefined op code $\overline{\text{XIPP}}$ indicates the XIP will execute the undefined instruction Recognition of $\overline{\text{XIPP}}$ will cause the SBP9989 to allow its memory bus signals to be pulled high, activate HOLDA and enter a hold state (i e , suspend internal operation) after it has stored its WP, PC and ST in the workspace defined by the interrupt level 2 trap vector Upon receipt of HOLDA, the XIP may then proceed to execute the undefined instruction During the instruction execution, the XIP may utilize the WP, PC and ST previously stored in memory by the SBP9989 Upon completion of its instruction execution, the XIP releases $\overline{\text{XIPP}}$ and allows the SBP9989 to resume bus control and restart instruction execution The SBP9989 will resume operation by reloading (from memory) its WP, PC and ST $\overline{\text{XIPP}}$ may also be used to initiate a trap to interrupt level 2 by going active during IAQ for any instruction This is useful for implementing break points or maintenance panels
$\overline{\text{LOAD}}$	4	IN	LOAD When active (low), $\overline{\text{LOAD}}$ causes the SBP9989 to set $\overline{\text{MPEN}}$ high, issue INTACK, store old PC, WP, and ST, set Status Register Bits 7 15 low and execute a nonmaskable interrupt with unmapped memory addresses FFFC$_{16}$ and FFFE$_{16}$ containing the associated trap vectors (WP and PC) The load sequence is initiated after the instruction being executed is completed $\overline{\text{LOAD}}$ will also terminate an idle state If $\overline{\text{LOAD}}$ is active at the end of a reset function, the $\overline{\text{LOAD}}$ trap will occur after the reset function is completed If $\overline{\text{LOAD}}$ is activated during a hold state (caused by $\overline{\text{XIPP}}$ or $\overline{\text{HOLD}}$), the SBP9989 will activate INTACK to indicate a pending $\overline{\text{LOAD}}$ needs to be serviced During hold states $\overline{\text{LOAD}}$ will remain active until the SBP9989 leaves the hold state and the above conditions are met $\overline{\text{LOAD}}$ may be used to implement bootstrap loaders Additionally, front panel routines may be implemented using CRU bits as front panel interface signals, and software control routines to direct the panel operations
$\overline{\text{RESET}}$	6	IN	RESET When active (low logic level), $\overline{\text{RESET}}$ causes the SBP9989 to reset itself, and inhibit $\overline{\text{WE}}$ and CRUCLK When $\overline{\text{RESET}}$ is released, the SBP9989 goes through a level zero interrupt sequence by causing $\overline{\text{MPEN}}$ to go to high, issuing INTACK, storing old PC, WP and ST, setting all status register bits low, acquiring the WP and PC trap vectors from memory locations 0000$_{16}$ and 0002$_{16}$, and then fetching the first instruction of the reset program environment if $\overline{\text{LOAD}}$ is not active The SBP9989 continuously samples $\overline{\text{RESET}}$ on low-to high clock transitions $\overline{\text{RESET}}$ must be active for one low-to-high transition of the clock and satisfy the hold time requirements of this signal

ARCHITECTURE AND TIMING

The memory word is 16-bits long as shown in Figure 1 Words are assigned even-numbered addresses in memory Each memory word contains two bytes of 8 bits each The instruction set of the SBP9989 allows both word and byte operations Byte instructions may address either byte as necessary Byte instructions that address a Workspace Register operate on the most significant byte (even address) of the Workspace Register and leave the least significant byte (odd address) unchanged Since the workspace is also addressable as a memory address, the least significant byte may be addressed if desired

The SBP9989 memory map is shown in Figure 2 The first 32 words are used for interrupt trap vectors and the next contiguous block of 32 memory words is used by the XOP (Extended Operation) instruction for trap vectors The last two unmapped memory words ($FFFC_{16}$ and $FFFE_{16}$) are used for the trap vector of the load signal The remaining memory is then available for programs, data, and Workspace Registers

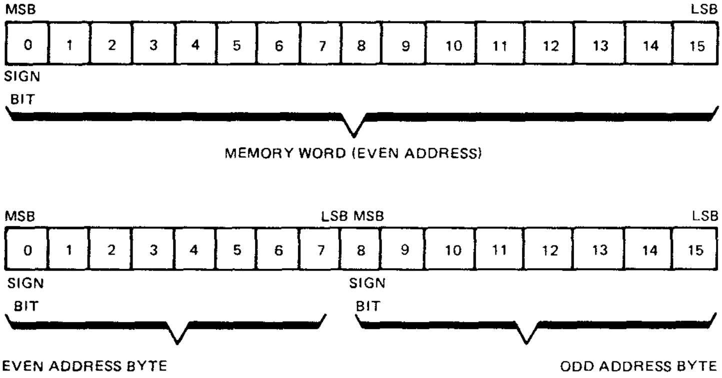


FIGURE 1 — WORD AND BYTE FORMATS

FUNCTIONAL DESCRIPTION

Block Diagram

The block diagram is shown on page 10 A flow chart, representative of functional operation, is shown on page 9 Addresses are supplied to the Address Bus (A0-A14) by the Memory Address Register (MA) Instructions read from memory are loaded into the Instruction Register (IR) via the Data Bus (D0-D15)

Bit-oriented input/output operations are provided by the Communication Register Unit (CRU) interface whereby 1 to 16 bits may be transferred by a single instruction

Arithmetic Logic Unit (ALU)

The arithmetic logic unit (ALU) is the computational component of the SBP9989 It performs all arithmetic and logic functions required to execute instructions The functions include addition, subtraction AND, OR, exclusive OR, and complement

The ALU is arranged in two 8-bit halves to accommodate byte operations Each half of the ALU operates on one byte of the operand During word operand operations, both halves of the ALU function in conjunction with each other However, during byte operand processing, the least significant half of the ALU operates in a passive mode, performing no operation on the data that it handles The most significant half of the ALU performs all operations on byte operands so that the status circuitry used in word operations is also used in byte operations

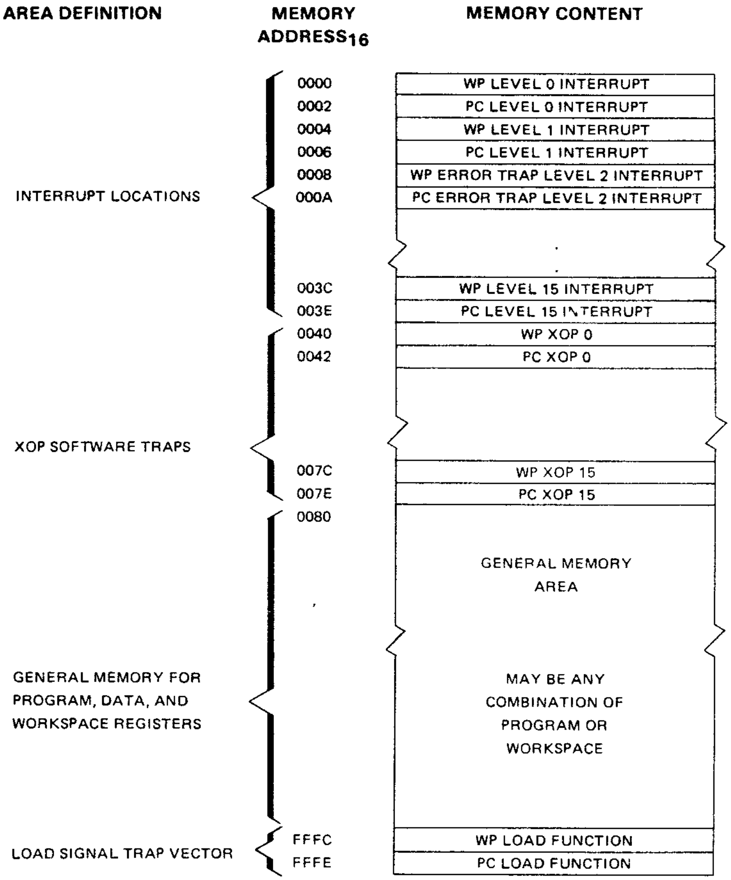


FIGURE 2 — MAP OF THE MEMORY ARRANGEMENT

Internal Registers

The following three internal registers are accessible to the user (programmer).

a. Program Counter (PC)
b. Status Register (ST)
c. Workspace Pointer (WP)

Other internal registers which are utilized during instruction acquisition or execution are inaccessible to the user.

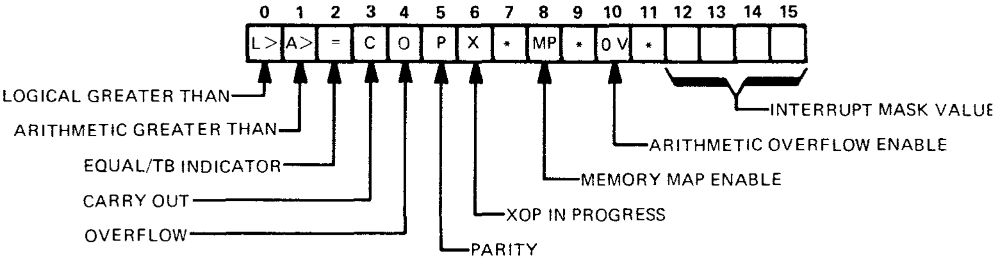
Program Counter (PC)

The program counter is a 15-bit register (left justified with the LSB hardwired to 0) that contains the word address of the instruction currently executing. The SBP9989 increments this address to fetch the next instruction from memory

If the current instruction in the microprocessor alters the contents of the PC, then a program branch occurs to the location specified by the altered contents of PC. All context instructions affect the contents of the PC.

Status Register (ST)

The status register is a fully implemented 16-bit register that reports the results of program comparisons, indicates program status conditions, supplies the arithmetic overflow enable and interrupt mask level to the interrupt priority circuits, and provides the external bit for memory expansion to 64K words and beyond. Each bit position in the register signifies a particular function or condition that exists in the SBP9989 as illustrated in Figure 3 Some instructions use the status register to check for a prerequisite condition, others affect the values of the bits in the register, and others load the entire status register with a new set of parameters. The description of the instruction set later in this document details the effect of each instruction on the status register



*These bits are functionally uncommitted and are available to the user

FIGURE 3 — STATUS REGISTER BIT ASSIGNMENTS

Workspace

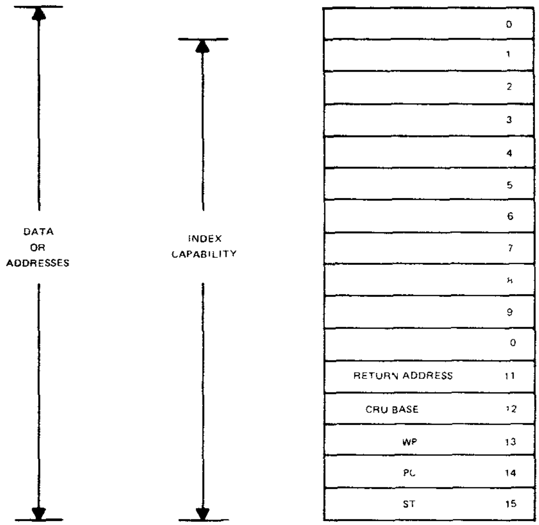
The SBP9989 uses blocks of memory words, called workspaces, for instruction operand manipulation (instead of internal hardware registers). A workspace occupies 16 contiguous memory words in any part of memory. The individual workspace registers may contain data or addresses, and function as operand registers, accumulators, address registers, or index registers Some workspace registers take on special significance during execution of certain instructions Figure 4 illustrates the workspace map A user-defined number of workspaces may exist in memory which provides a high degree of program flexibility

Workspace Pointer (WP)

To locate the workspace in memory, one hardware register called the Workspace Pointer (WP) is used The Workspace Pointer is a 15-bit register (left justified with the LSB hardwired to 0) that contains the memory address of the first word in the workspace The SBP9989 can then access any register in the workspace by adding two times the register number to the contents of the Workspace Pointer and initiating a memory request for that word Figure 5 illustrates the relationship between the Workspace Pointer and its corresponding workspace in memory.

Context Switching

The workspace concept is particularly valuable during operations that require a context switch, which is a change from one program environment to another or to a subroutine as in the case of an interrupt Such an operation using a conventional multi-register arrangement requires that at least part of the contents of the register file be stored and relocated using a memory cycle to store or fetch each word This operation is accomplished by changing the Workspace Pointer A complete context switch requires only three store cycles and three fetch cycles, exchanging the program counter, status register and Workspace Pointer After the switch, the Workspace Pointer contains the starting address of a new 16-word workspace in memory for use in the new routine A corresponding time saving occurs when the original context is restored Instructions that result in a context switch include branch and load workspace pointer (BLWP), return from subroutine (RTWP), and extended operation (XOP) instruction Device interrupts, the arithmetic overflow interrupt, illegal opcode detection trap, $\overline{\text{RESET}}$, and $\overline{\text{LOAD}}$ also cause a context switch by forcing a trap to a service subroutine.



NOTE THE WP REGISTER CONTAINS THE ADDRESS OF WORKSPACE REGISTER ZERO

FIGURE 4 — WORK SPACE MAP

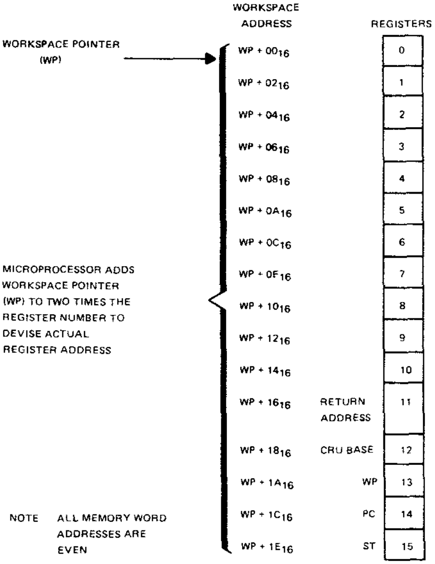


FIGURE 5 — WORKSPACE POINTER AND REGISTERS

In the case of a memory write cycle WE becomes active (logic level low) with the first high to low transition of the clock after MEMEN becomes active DBIN remains inactive (low) At the end of a memory write cycle WE and MEMEN become inactive (MEMEN on the low to high transition of clock WE on the preceding high to low transition of clock)

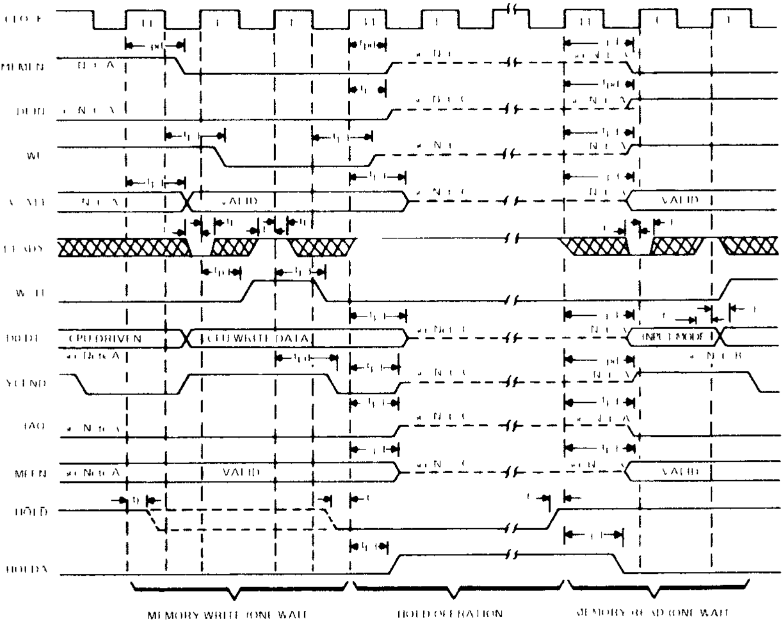
During either a memory read or a memory write operation READY may be used to extend the duration of the associated memory cycle such that the speed of the memory system may be coordinated with the speed of the SBP9989 If READY is inactive (logic level low) during the first low to high transition of the clock after MEMEN becomes active the SBP9989 will enter a wait state suspending further progress of the memory cycle The first low to high transition of the clock after READY becomes active terminates the wait state and allows normal completion of the memory cycle The cycle end signal (CYCEND) will be activated (logic level low) for one clock period during each microinstruction cycle (i e memory operation non memory internal operation etc) CYCEND will be activated on the low to high transition of the clock which initiates the last clock period of a microinstruction cycle and will be deactivated on the next low to high clock transition

HOLD, DMA Interface

The SBP9989 hold facilities allow both the microprocessor and external devices to share a common memory To gain memory bus control an external device requiring direct memory access (DMA) sends a hold request (HOLD) to the SBP9989 When the next cycle (memory or non memory) occurs the microprocessor enters a hold state and signals its surrender of the memory bus to the external device via a hold acknowledge (HOLDA) signal

Receiving the hold acknowledgement the external device can proceed to utilize the common memory After its memory requirements have been satisfied the external device returns memory bus control by releasing HOLD

When HOLD becomes active (logic level low) the SBP9989 enters a hold state at the beginning of the next available cycle as shown in Figure 9 Upon entering a hold state HOLDA becomes active (pulled to logic level high) with the following signals pulled to a high logic level by individual pullup resistors tied to each respective open collector output DBIN MEMEN IAQ MPEN WE CYCEND A0 through A14 and D0 through D15 When HOLD becomes inactive the SBP9989 exits the hold state and regains memory bus control



NOTE A THERE IS MAXIMUM DELAY OF 200 NANOSECONDS FOR THE SIGNALS TO RETURN TO [illegible] LEVEL ON LEAVING A HOLD STATE
B VALID MEMORY READ DATA
C DRIVEN BY EXTERNAL DEVICE

FIGURE 9 MEMORY BUS TIMING FOR HOLD OPERATION

xtended Instruction Processor Interface

The extended instruction processor (XIP) interface provides for easy extension of the SBP9989 arithmetic logic processing functions by facilitating the addition of external hardware instruction processors while also permitting the usage of software interpretive implementations of extended instructions The XIP interface provides user transparency regardless of the method of implementation (i e hardware or software) potentially eliminating software overhead It allows true software transportability so that programs generated for systems based on the SBP9989 employing XIP s and those without XIP s can be identical

The XIP interface utilizes the SBP9989 memory bus with direct memory access (DMA) capability and the extended instruction processor present ($\overline{\text{XIPP}}$) signal as shown in Figure 10 The XIP interface requires the XIP to gain control of the SBP9989 memory bus during execution of any extended instructions encountered in the device program instruction stream The extended instructions are assigned (by the user) operation codes (op codes) which are the illegal (undefined) operation codes NOP s (ILLOP s)

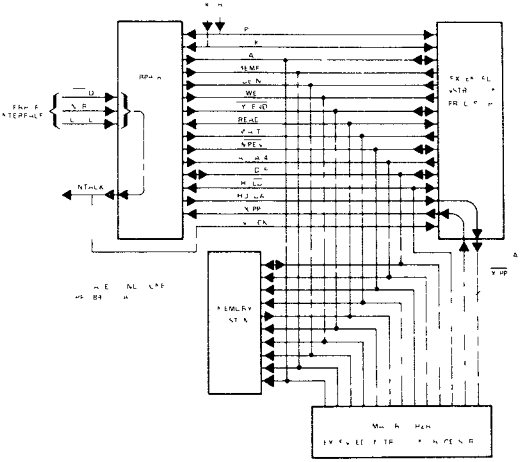
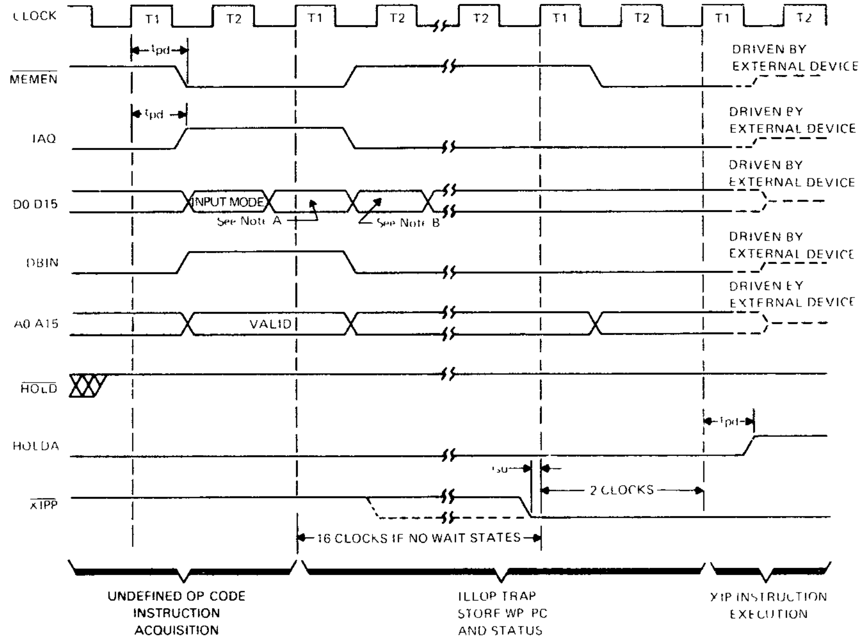


FIGURE 10 — EXTENDED-INSTRUCTION INTERFACE PROCESSOR

The sequence that characterizes a typical extended instruction execution is shown in Figures 11 and 12 As illustrated the SBP9989 fetches the instruction that contains the op code assigned to an extended instruction The XIP detects the occurrence of the instruction fetch operation (IAQ active) and latches the instruction op-code data present on the memory data bus The XIP decodes the latched op-code as one of its instructions and asserts $\overline{\text{XIPP}}$ to the SBP9989 (non-extended instruction op codes would be ignored by the XIP) Recognition of the illegal op-code causes the SBP9989 to execute a program trap and store its context (i e WP PC and ST) in memory Storage will be in registers 13 14 and 15 of the workspace defined by the WP value contained in the SBP9989 Level 2 interrupt vector locations If $\overline{\text{XIPP}}$ is also active (logic level low) the SBP9989 allows its memory bus signals to be pulled to a logic level high issues HOLDA (Hold Acknowledge) and suspends internal operation Having received HOLDA the XIP assumes control of the memory bus and proceeds with execution of the extended instruction(s) During its instruction execution the XIP may access the PC WP and ST values (previously stored in memory), via the interrupt Level 2 workspace address as required to derive instruction operands and indicate execution results (status) After completing instruction execution the XIP releases $\overline{\text{XIPP}}$ Detecting the removal of $\overline{\text{XIPP}}$ causes the SBP9989 to remove HOLDA activate its memory bus drivers (i e resume bus control) restore its context (WP PC and ST) from memory and continue instruction processing Where it resumes processing is determined by the PC value (updated by the XIP during execution) which it reacquires from memory after resuming control


CLOCK
T1
T2
T1
T2
T2
T1
T2
T1
T2
tpd
MEMEN
DRIVEN BY EXTERNAL DEVICE
tpd
IAQ
DRIVEN BY EXTERNAL DEVICE
D0 D15
INPUT MODE
See Note A
See Note B
DRIVEN BY EXTERNAL DEVICE
DBIN
DRIVEN BY EXTERNAL DEVICE
A0 A15
VALID
DRIVEN BY EXTERNAL DEVICE
HOLD
HOLDA
tpd
tsu
2 CLOCKS
XIPP
16 CLOCKS IF NO WAIT STATES
UNDEFINED OP CODE INSTRUCTION ACQUISITION
ILLOP TRAP STORE WP PC AND STATUS
XIP INSTRUCTION EXECUTION

NOTES A VALID UNDEFINED OP CODE DATA FROM MEMORY
B INPUT MODE

FIGURE 11 — EXTENDED INSTRUCTION PROCESSOR INTERFACE TIMING (ILLOP DETECTION)

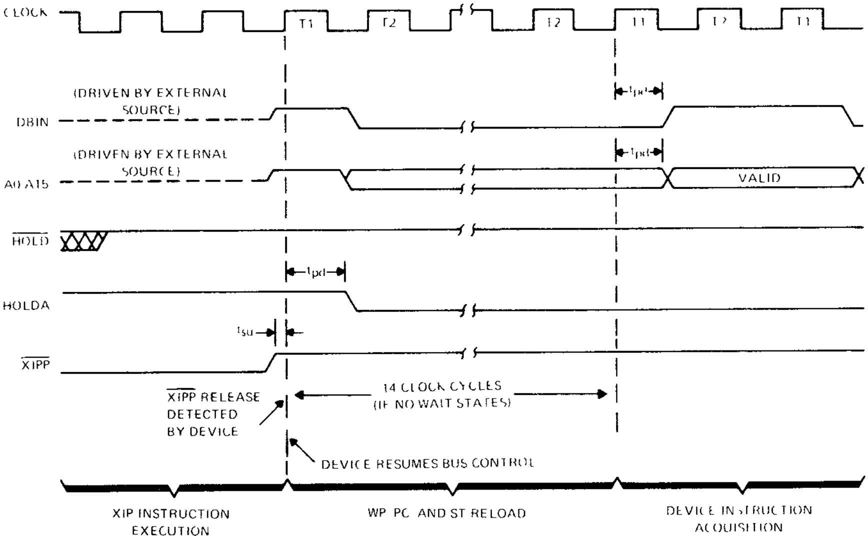

CLOCK
T1
T2
T2
T1
T2
T1
(DRIVEN BY EXTERNAL SOURCE)
DBIN
tpd
(DRIVEN BY EXTERNAL SOURCE)
A0 A15
tpd
VALID
HOLD
tpd
HOLDA
tsu
XIPP
XIPP RELEASE DETECTED BY DEVICE
14 CLOCK CYCLES (IF NO WAIT STATES)
DEVICE RESUMES BUS CONTROL
XIP INSTRUCTION EXECUTION
WP PC AND ST RELOAD
DEVICE INSTRUCTION ACQUISITION

FIGURE 12 — EXTENDED-INSTRUCTION PROCESSOR INTERFACE TIMING (WP,PC, ST RELOAD)

The XIP can be implemented with chaining capability, i e , the ability to execute a sequence of extended instructions without returning control to the SBP9989 If an interrupt or $\overline{\text{LOAD}}$ occurs and the interrupt mask conditions are satisfied, the SBP9989 will activate INTACK (interrupt acknowledge) to indicate that an interrupt needs to be serviced Servicing of the interrupt will occur upon release of $\overline{\text{XIPP}}$ by the XIP and completion of the above context restore bus control resumption sequence

As shown in Figure 10, the XIP interface provides for other Direct Memory Access (DMA) devices or more than one XIP device However, it is the responsibility of the XIP device to receive, condition and transmit the required interface signals (i e , $\overline{\text{HOLD}}$, HOLDA, and $\overline{\text{XIPP}}$) to satisfy additional DMA or XIP requirements

For configurations not containing external XIP hardware, the $\overline{\text{XIPP}}$ signal will not be activated Therefore the SBP9989 completes the context switch operations by loading PC data from the interrupt Level 2 trap vector Software starting at this PC location then executes the intended function(s)

Multiprocessor Interlock

The Multiprocessor Interlock (MPILCK) signal permits the implementation of an indivisible test and set-semaphore mechanism for use in multiprocessor applications The SBP9989 activates the MPILCK signal (logic level high) during the execution of the Absolute Value (ABS) instruction as shown in Figure 13 MPILCK is activated on the low-to-high clock transition, initiates the first ABS operand memory access and remains active until the completion of the ABS instruction (i e , occurrence of next IAQ or servicing of an interrupt, $\overline{\text{LOAD}}$ or $\overline{\text{RESET}}$) $\overline{\text{HOLD}}$ is sampled during this time, so MPILCK can be used to control the assertion of $\overline{\text{HOLD}}$

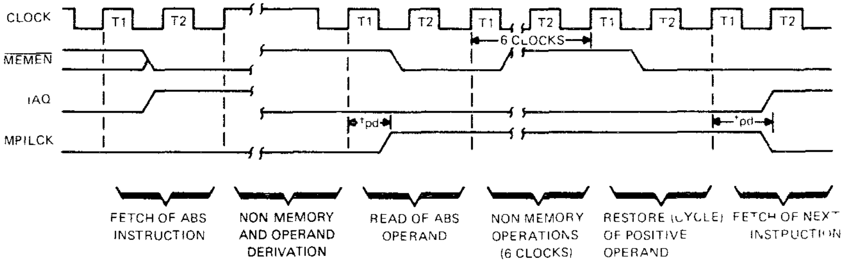

CLOCK
T1
T2
T1
T2
T1
T2
T1
T2
T1
T2
6 CLOCKS
MEMEN
IAQ
MPILCK
tpd
tpd
FETCH OF ABS INSTRUCTION
NON MEMORY AND OPERAND DERIVATION
READ OF ABS OPERAND
NON MEMORY OPERATIONS (6 CLOCKS)
RESTORE CYCLE OF POSITIVE OPERAND
FETCH OF NEXT INSTRUCTION

FIGURE 13a — MULTIPROCESSOR INTERLOCK TIMING (NEGATIVE-OPERAND)

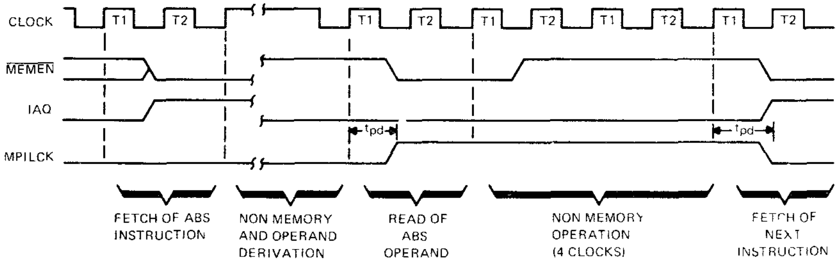
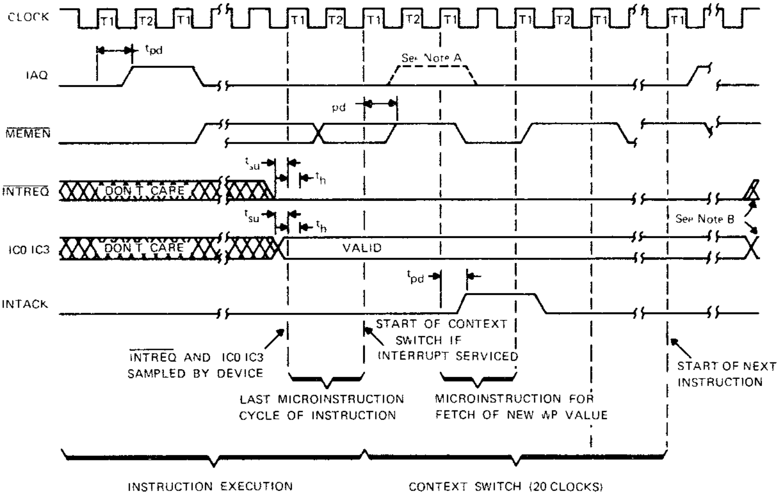

CLOCK
T1
T2
T1
T2
T1
T2
T1
T2
T1
T2
MEMEN
IAQ
MPILCK
tpd
tpd
FETCH OF ABS INSTRUCTION
NON MEMORY AND OPERAND DERIVATION
READ OF ABS OPERAND
NON MEMORY OPERATION (4 CLOCKS)
FETCH OF NEXT INSTRUCTION

FIGURE 13b — MULTIPROCESSOR INTERLOCK TIMING (POSITIVE-OPERAND)

INTERRUPTS

The SBP9989 employs 16 interrupt levels with the highest priority Level 0 and lowest priority Level 15 Level 0 is reserved for the RESET function Level 2 is reserved at the user's option for the arithmetic overflow interrupt and/or an illegal op-code trap Interrupt Levels 1 through 15 may be used for external device interrupts

External device interrupts are input via the interrupt request (INTREQ) signal line and the four interrupt code lines (IC0 IC3) Figure 14 shows the timing for the sampling of these inputs and their effect on SBP9989 operation Activation of the INTREQ input causes a comparison with the interrupt code (IC0-IC3) with the interrupt mask contained in status register bits ST12 through ST15 When the level of the pending interrupt is less than or equal to the enabling mask level (higher or equal priority interrupt), the SBP9989 recognizes the interrupt and initiates a context switch following completion of the currently executing instruction The new context (WP and PC) is fetched from the interrupt vector locations and the SBP9989 sets MPEN to one, activates the INTACK (interrupt acknowledge) signal line Then the previous context (WP, PC and ST) is stored in workspace registers 13, 14 and 15, respectively, of the new workspace Bits 7 through 11 of the status register are forced to zero and sets the interrupt mask to a value that is one less than the level of the interrupt being serviced, except for Level 0 interrupt, which loads zero into the mask This allows only interrupts of higher priority to interrupt a service routine The SBP9989 also inhibits interrupts until the first instruction of the device service routine has been executed All interrupt requests should remain active until recognized in the device service routine The individual service routines must reset the interrupt requests before the routine is completed


CLOCK
T1
T2
IAQ
t_{pd}
See Note A
MEMEN
pd
INTREQ
DON T CARE
t_{su}
t_h
IC0 IC3
VALID
See Note B
INTACK
t_{pd}
INTREQ AND IC0 IC3 SAMPLED BY DEVICE
START OF CONTEXT SWITCH IF INTERRUPT SERVICED
START OF NEXT INSTRUCTION
LAST MICROINSTRUCTION CYCLE OF INSTRUCTION
MICROINSTRUCTION FOR FETCH OF NEW WP VALUE
INSTRUCTION EXECUTION
CONTEXT SWITCH (20 CLOCKS)

NOTES A NEXT IAQ IF INTERRUPT NOT SERVICED
B INTREQ AND IC0 IC3 CHANGED BY INTERRUPT SERVICE ROUTING
C THIS DIAGRAM ASSUMES VALID IC IS OF HIGHER PRIORITY THAN CODE IN STATUS REGISTER

FIGURE 14 — INTERRUPT TIMING

If a higher-priority occurs, a second context switch occurs to service the higher priority interrupt When that routine is complete, a return instruction (RTWP) restores the routine parameters to complete processing of the lower priority interrupt All interrupt subroutines should terminate with the return instruction to restore original program parameters The interrupt vector locations, device assignments enabling mask value, and the interrupt codes are shown in Table 1

During a SBP9989 hold state resulting from the activation of $\overline{\text{HOLD}}$ or $\overline{\text{XIPP}}$, the SBP9989 will continue to sample the interrupt code lines Upon activation of $\overline{\text{INTREQ}}$ if the code is less than or equal to the device interrupt mask level the SBP9989 will activate the INTACK signal to indicate a pending interrupt needs servicing The INTACK signal will then remain active until $\overline{\text{HOLD}}$ or $\overline{\text{XIPP}}$ is released

TABLE 1 — INTERRUPT LEVEL DATA

INTERRUPT LEVEL	VECTOR LOCATION (MEMORY ADDRESS IN HEX)	DEVICE ASSIGNMENT	ENABLING MASK VALUES (ST12 THRU ST15)	INTERRUPT CODES (IC0 THRU IC3)
(Highest priority) 0	00	Reset	0 through F (see Note 1)	0000
1	04	External Device	1 through F	0001
2	08	Arithmetic Overflow or	2 through F (see Note 2)	See Note 2
2	08	Illegal op code or	0 through F (see Note 3)	See Note 3
2	08	$\overline{\text{XIPP}}$ Active during IAQ or		See Note 4
2	08	External Device	2 through F	0010
3	0C	'	3 through F	0011
4	10	''	4 through F	0100
5	14		5 through F	0101
6	18		6 through F	0110
7	1C		7 through F	0111
8	20	''	8 through F	1000
9	24		9 through F	1001
10	28	''	A through F	1010
11	2C		B through F	1011
12	30		C through F	1100
13	34		D through F	1101
14	38		E and F	1110
(Lowest priority) 15	3C	External Device	F only	1111

NOTES 1 Level 0 cannot be disabled

2 Arithmetic overflow interrupt is generated internal to the device and is enabled disabled by bit 10 of the status register

3 Illegal op code trap is generated internal to the device and it cannot be disabled by the interrupt mask

4 $\overline{\text{XIPP}}$ (pin 58) is inactive at logic level high

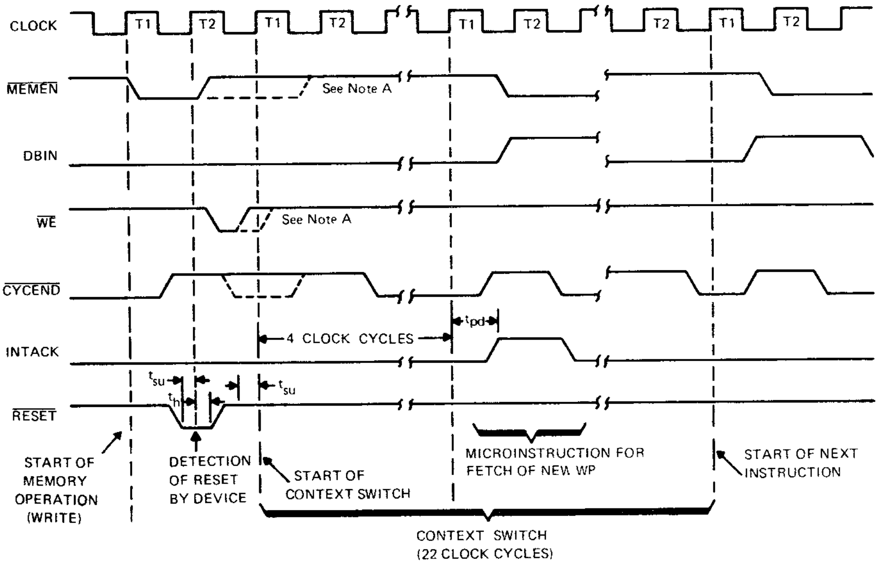
Interrupt Level 0 — RESET

Interrupt Level 0 is reserved for the $\overline{\text{RESET}}$ input to the SBP9989 When asserted (logic level low), $\overline{\text{RESET}}$ causes the SBP9989 to reset itself and inhibit $\overline{\text{WE}}$ and CRUCLK

When $\overline{\text{RESET}}$ is released, a Level 0 interrupt sequence is initiated acquiring the WP and PC trap vectors from memory locations 0000_{16} and 0002_{16}. INTACK is activated, all status register bits are set low, and the first instruction of the reset program environment is fetched. If $\overline{\text{LOAD}}$ is active, the $\overline{\text{LOAD}}$ trap occurs after the $\overline{\text{RESET}}$ function is completed The SBP9989 continuously samples $\overline{\text{RESET}}$ on low-to-high clock transitions as shown in Figure 15. To be recognized, $\overline{\text{RESET}}$ must be active for one low-to-high transition of clock and must satisfy the setup and hold time requirements

Interrupt Level 2

Interrupt Level 2 has two additional capabilities associated with its usage. Arithmetic overflow conditions, indicated by status register bit 4 (ST4) = 1, can cause a Level 2 interrupt to occur at the end of the instruction which generated the overflow condition. Servicing of this overflow interrupt can be enabled/disabled by status register bit 10 (ST10), the Arithmetic Overflow Enable Bit (i.e., ST10 = 1 enables overflow interrupt, ST10 = 0 disables overflow interrupt). The overflow interrupt can also be inhibited by the interrupt mask (ST12-ST15) or overridden by a pending Level 0 or Level 1 interrupt. If servicing the overflow interrupt is overridden by a higher priority (Level 0 or 1) interrupt, the overflow condition will be retained in the contents of the status register, which are saved by the higher-priority-interrupt context switch Returning from the higher-priority-interrupt subroutine via an RTWP instruction causes the overflow condition to be reloaded into status register bit 4 (ST4) and the overflow interrupt to occur upon completion of the RTWP instruction. Servicing of a Level 2 arithmetic overflow interrupt forces the interrupt mask to 0001_2 The arithmetic overflow interrupt service routine must reset ST4 to zero before the routine is complete


CLOCK
T1
T2
T1
T2
T1
T2
T2
T1
T2
MEMEN
See Note A
DBIN
WE
See Note A
CYCEND
INTACK
4 CLOCK CYCLES
tpd
tsu
th
tsu
RESET
START OF MEMORY OPERATION (WRITE)
DETECTION OF RESET BY DEVICE
START OF CONTEXT SWITCH
MICROINSTRUCTION FOR FETCH OF NEW WP
START OF NEXT INSTRUCTION
CONTEXT SWITCH (22 CLOCK CYCLES)

NOTES A DASHED LINES INDICATE SIGNAL WAVEFORMS IF $\overline{\text{RESET}}$ HAD NOT OCCURRED
B CASE SHOWN ASSUMES $\overline{\text{RESET}}$ IS DETECTED DURING MEMORY OPERATION
C SYNCHRONIZE $\overline{\text{RESET}}$ WITH IAQ ON T1 TO PREVENT LOSS OF INSTRUCTION IN PROGRESS
D $\overline{\text{RESET}}$ MAY OCCUR ON ANY LOW TO HIGH CLOCK EDGE FOLLOWING DEACTIVATION OF $\overline{\text{RESET}}$.
E CONTEXT SWITCH WILL START ON LOW TO HIGH CLOCK EDGE FOLLOWING DEACTIVATION OF $\overline{\text{RESET}}$

FIGURE 15 — RESET TIMING

Undefined Op-code Trap

The acquisition and execution of illegal SBP9989 (undefined) op codes cause a trap operation to occur using the information stored in the Level 2 vectors The op codes which cause the trap are 0000 007F 00A0 017F 0320 033F 0780-07FF 0C00 OFFF As described earlier if the $\overline{\text{XIPP}}$ signal is activated during the undefined instruction the SBP9989 will suspend operation and the instruction will be executed by the XIP However if the $\overline{\text{XIPP}}$ signal remains inactive the undefined op-code will cause a trap and a context switch will occur The occurrence of the trap is non maskable (i e not controlled by the interrupt mask value) and the trap will override any level interrupt Interrupts are inhibited until the first instruction of the trap subroutine is executed The occurrence of the undefined op code trap does not change the interrupt mask

Communication Register Unit (CRU) Interface

The Communications Register Unit (CRU) is a direct command driven bit oriented I O interface The CRU may directly address, in bit-fields of one to sixteen bits up to 4096 peripheral input bits and up to 4096 peripheral output bits The SBP9989 executes three single-bit and two multiple bit CRU instructions The single bit instructions include test bit (TB), set bit to one (SBO) and set bit to zero (SBZ) the multiple bit instructions include load CRU (LDCR) and store CRU (STCR)

As shown in Figure 16 the SBP9989 utilizes three dedicated I O signals CRUIN CRUOUT CRUCLK the least significant twelve bits of the address bus DBIN READY and WAIT to support the CRU interface CRU interface timing is shown in Figures 17 and 18

To transfer a data bit to a CRU device the SBP9989 outputs the corresponding CRU bit address on address bus bits A3 through A14, a CRUCLK pulse and the respective data bit on CRUOUT (Address bus bits A0 through A2 are set to zero during CRU transfers) This process is repeated until transfer of the entire field of data bits specified by the CRU instruction has been accomplished To transfer a data bit from a CRU device the SBP9989 outputs the corresponding CRU bit address on address bits A3 through A14 and receives the respective data bit on CRUIN No CRUCLK pulses occur during a CRU input operation

During all CRU input and output operations the SBP9989 will activate the DBIN output (logic level high) and place the 16-bit Data Bus in the input mode Activation of DBIN will allow users to detect the occurrence of a CRU cycle and facilitate cycle stealing by a DMA device since all uses of the DATA and ADDRESS busses will be indicated by the activation of DBIN or $\overline{\text{MEMEN}}$

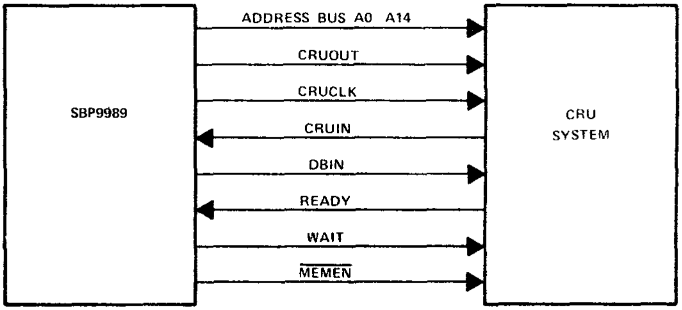


FIGURE 16 — CRU INTERFACE SUPPORT

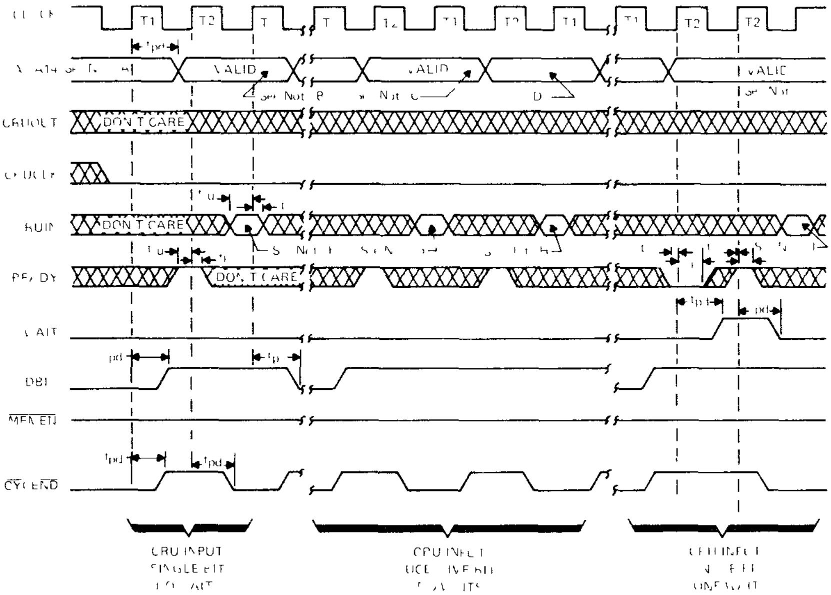
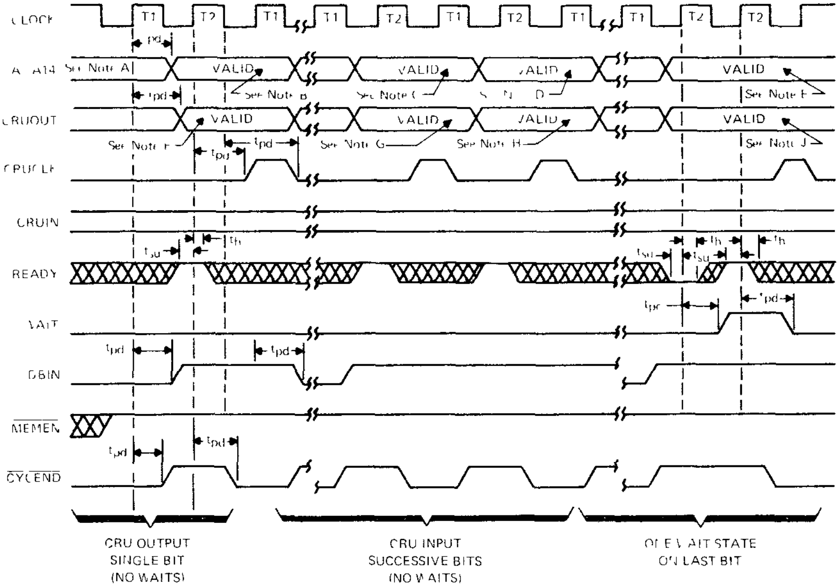


FIGURE 17 — CRU TIMING (INPUT OPERATION)



NOTE A A(0-2) ARE SET TO ZERO FOR CRU OPERATIONS
B CRU BIT ADDRESS M
C CRU BIT ADDRESS N

D CRU BIT ADDRESS N + 1
E CRU BIT ADDRESS R
F CRU DATA OUT ADDRESS M

G CRU DATA OUT ADDRESS N
H CRU DATA OUT ADDRESS N
J CRU DATA OUT ADDRESS R

FIGURE 18 — CRU TIMING (OUTPUT OPERATION)

During either a CRU input or output operation, READY may be used to extend the duration of the associated CRU operation such that the speed of the CRU device may be coordinated with the speed of the SBP9989 If READY is inactive (logic level low) during the first low-to-high transition of clock after the initiation of the CRU operation the SBP9989 will enter a wait state (i e , WAIT becomes active) suspending further progress of the CRU cycle The first low to high transition of clock after READY becomes active terminates the wait state and allows normal completion of the CRU operation

Single Bit CRU Operation

The SBP9989 performs three single-bit CRU functions Test bit (TB) set bit to one (SBO) and set bit to zero (SBZ) To identify the bit to be operated upon, a CRU bit address is developed and placed on the address bus, A3 to A14 For the two output operations (SBO and SBZ), the device also activates DBIN, generates a CRUCLK pulse, indicating an output operation to the CRU device, and placed bit 7 of the instruction word on the CRUOUT line to accomplish the specified operation (Bit 7 is a one for SBO and a zero for SBZ) A test bit (TB) instruction transfers the addressed CRU bit from the CRUIN input line to bit 2 of the status register (ST12)

The SBP9989 develops a CRU-bit address for the single-bit operations from the CRU base address contained in work-space register 12 and the signed displacement count contained in bits 8 through 15 of the instruction The displacement allows two's complement addressing from base minus 128 bits through base plus 127 bits The base address from workspace register 12 is added to the signed displacement specified in the instruction and the result is loaded onto the address bus Figure 19 illustrates the development of a single-bit CRU address

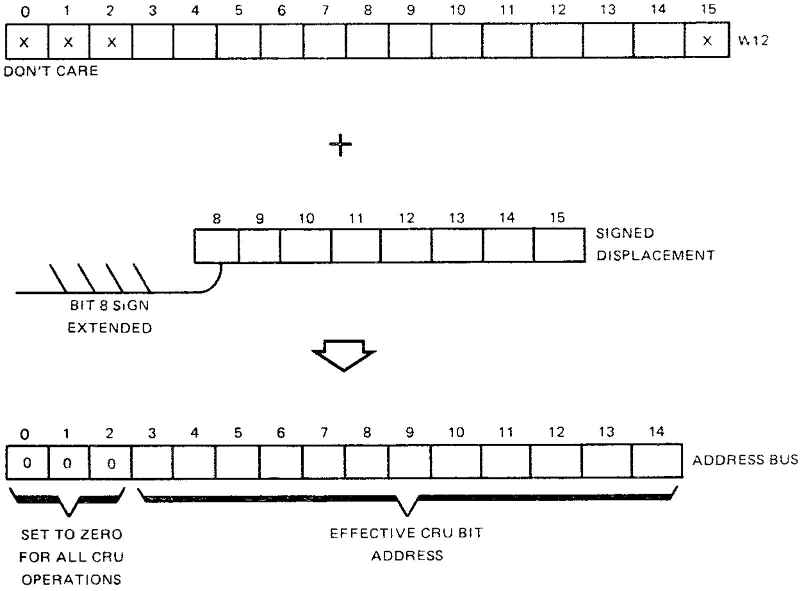


FIGURE 19 — BIT ASSIGNMENT FOR SINGLE-BIT CRU ADDRESS

Multiple Bit CRU Operations

The SBP9989 performs two multiple-bit CRU operations store communication register (STCR) and load communications register (LDCR). Both operations perform a data transfer from the CRU-to-memory or from memory-to-CRU as illustrated in Figure 20 Although the figure illustrates a full 16-bit transfer operation, any number of bits from 1 through 16 may be involved The LDCR instruction fetches a word from memory and right shifts it to transfer it serially to CRU output bits. The LDCR involves eight or fewer bits, those bits come from the right-justified field within the addressed byte of the memory word. If the LDCR involves nine or more bits, those bits come from the right-justified field within the whole memory word. When transferred to the CRU interface, each successive bit receives an address that is sequentially greater than the address for the previous bit. This addressing mechanism results in an order reversal of the bits, that is, bit 15 of the memory word (or bit 7) becomes the lowest addressed bit in the CRU and bit 0 becomes the highest addressed bit in the CRU field

A STCR instruction transfers data from the CRU to memory. If the operation involves a byte or less transfer, the transferred data will be stored right-justified in the memory byte with leading bits set to zero If the operation involves from nine to sixteen bits, the transferred data is stored right-justified in the memory word with leading bits set to zero When the input from the CRU device is complete, the first bit from the CRU is the least-significant bit position in the memory word or byte

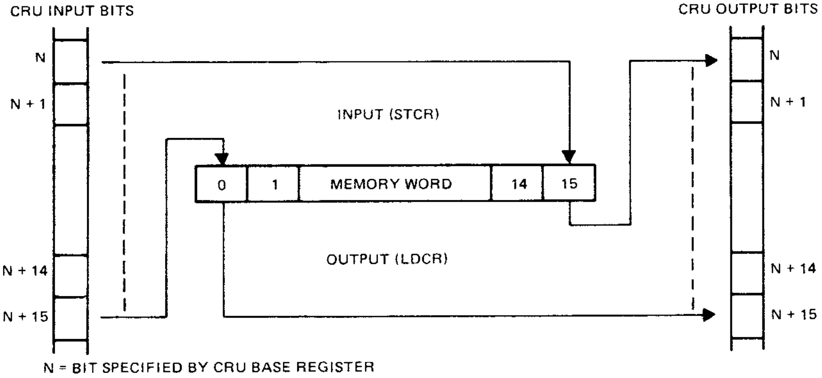


FIGURE 20 — BIT TRANSFER WITH CRU INTERFACE

xternal Instructions

The SBP9989 has five external instructions that allow user-defined external functions to be initiated under program control These instructions are CKON, CKOF, RSET, IDLE, and LREX. These mnemonics, except for IDLE, relate to functions implemented in the 990 minicomputer and do not restrict use of the instructions to initiate various user-defined functions IDLE causes the device to enter the idle state and remain until an interrupt, $\overline{\text{RESET}}$, or $\overline{\text{LOAD}}$ occurs When any of these five instructions are executed, a unique 3-bit code appears on the most-significant 3 bits of the address bus (A0 through A2) along with activation of DBIN and the occurrence of a CRUCLK pulse When in an idle state the codes on the address bus and CRUCLK pulse occur repeatedly until the idle state is terminated The 3-bit codes are shown in Table 2

OAD Function

The $\overline{\text{LOAD}}$ signal allows cold start ("bootstrap") ROM loaders and front panel functions to be implemented When active, $\overline{\text{LOAD}}$ causes a trap immediately following the instruction being executed Unmapped memory locations FFFC$_{16}$ and FFFE$_{16}$ are used to obtain the trap vector (WP and PC respectively) The interrupt acknowledge (INTACK) output is activated during the fetch of the new WP value. The old PC, WP and ST are loaded into the new workspace, status register bits ST7-ST15 are set to 0 (logic level low), and $\overline{\text{MPEN}}$ is set to one Then, program execution resumes using the new PC and WP During a hold state (caused by activation of $\overline{\text{HOLD}}$ or $\overline{\text{XIPP}}$), the SBP9989 will continue to sample the $\overline{\text{LOAD}}$ input If $\overline{\text{LOAD}}$ is activated, the SBP9989 will generate INTACK to indicate a pending $\overline{\text{LOAD}}$ needs servicing INTACK will then remain active until $\overline{\text{HOLD}}$ or $\overline{\text{XIPP}}$ is released

During non-hold states, $\overline{\text{LOAD}}$ shall remain active for one instruction period (i e , $\overline{\text{LOAD}}$ should go active during IAQ and remain active until the next IAQ) or until the external device detects the activation of INTACK concurrent with a memory read operation at location FFFC$_{16}$ During hold states, $\overline{\text{LOAD}}$ shall remain active until the external device detects the activation of INTACK concurrent with a memory read operation at location FFFC$_{16}$

TABLE 2 — EXTERNAL INSTRUCTION FUNCTION TABLE

EXTERNAL INSTRUCTION	A0	A1	A2
LREX	H	H	H
CKOF	H	H	L
CKON	H	L	H
RSET	L	H	H
IDLE	L	H	L

STANDARD INSTRUCTION SET

Each SBP9989 instruction performs one of the following operations

1 Arithmetic, logical, comparison, or manipulation operation on data
2. Loading or storage of internal registers (program counter, workspace pointer, or status)
3 Data transfer between memory and external devices via the CRU
4 Control functions

TERMS AND DEFINITIONS

The terms used in describing the instructions and status bits of the SBP9989 are defined below.

TERMS AND DEFINITIONS

TERM	DEFINITION
B	Byte Indicator (1 = byte, 0 = word)
C	Bit Count
D	Destination address register
DA	Destination address
IOP	Immediate operand
LSB(n)	Least significant (right-most) bit of n
MSB(n)	Most significant (left most) bit of n
PC	Program Counter
Result	Result of operation performed by instruction
S	Source address register
SA	Source address
ST	Status register
STn	Bit n of status register
T_D	Destination address-mode control
T_S	Source address-mode control
WR	Workspace register
WRn	Workspace register n
WR(0,1)	Concatenation of WR0 and WR1 to form a 32 bit register
a → b	a is transferred to b
$\|n\|$	Absolute value of n
+	Arithmetic addition
–	Arithmetic subtraction
AND	Logical AND
OR	Logical OR
$\oplus$	Logical exclusive OR
$\overline{n}$	Logical complement of n
x	Arithmetic multiplication

ADDRESSING MODES

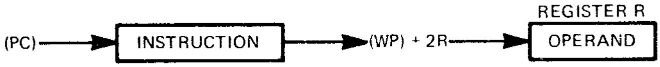
The SBP9989 instructions contain a variety of available modes for addressing random memory data (e.g., program parameters and flags), or formatted memory data (character strings, data lists, etc.). These addressing modes are:

- o Workspace Register Addressing
- o Workspace Register Indirect Addressing
- o Workspace Register Indirect Auto Increment Addressing
- o Symbolic (Direct) Addressing
- o Indexed Addressing
- o Immediate Addressing
- o Program Counter Relative Addressing
- o CRU Relative Addressing

The following figures graphically describe the derivation of the effective address for each addressing mode. The applicability of addressing modes to particular instructions is described next along with the description of the operations performed by the instruction. The symbols following the names of the addressing modes (R, *R, *R+, @LABEL, or @TABLE(R)) are general forms used by 9900 assemblers to select the addressing modes for Register R.

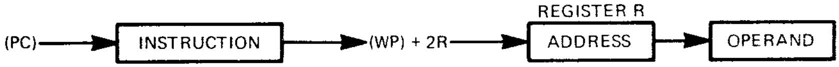
Workspace Register Addressing . . R

The workspace register addressing mode is specified by setting the 2 bit T field (T_S or T_D) of the instruction word equal to 00. Workspace register R contains the operand.



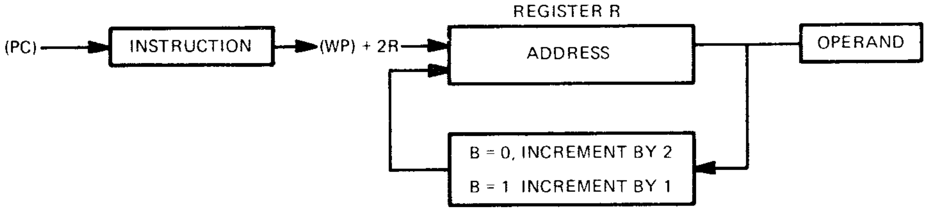
Workspace Register Indirect Addressing *R

The workspace register indirect addressing mode is specified by setting the 2 bit T field (T_S or T_D) in the instruction word equal to 01. Workspace register R contains the address of the operand.



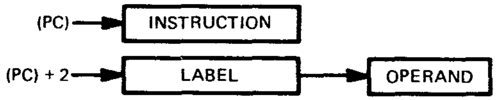
Workspace Register Indirect Auto-Increment Addressing . . . *R+

The workspace register indirect auto increment addressing mode is specified by setting the 2 bit T field (T_S or T_D) in the instruction word equal to 11. Workspace register R contains the address of the operand. After the address of the operand is acquired, the contents of workspace register R is incremented.



Symbolic (Direct) Addressing . . . @LABEL

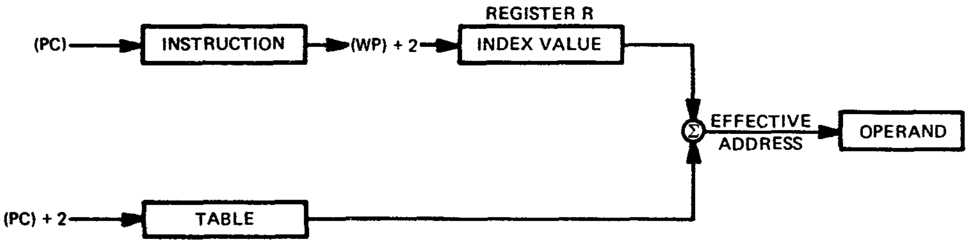
The symbolic addressing mode is specified by setting the 2-bit T field (T_S or T_D) in the instruction word equal to 10 and setting the corresponding S or D field equal to 0. The word following the instruction contains the address of the operand.



Indexed Addressing . . . @TABLE(R)

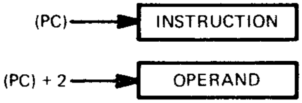
The indexed addressing mode is specified by setting the 2-bit T field (T_S or T_D) of the instruction word equal to 10. The value in the corresponding S or D field is the register which contains the index value Register 0 may not be used for indexed addressing.

The word following the instruction contains the base address. Workspace register R contains the index value. The sum of the base address and the index value results in the effective address of the operand.



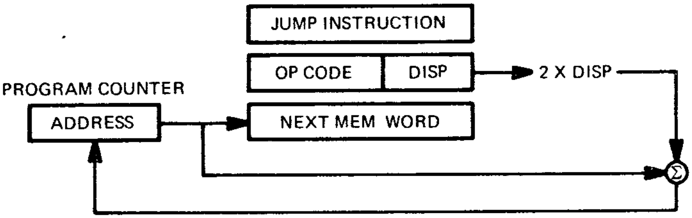
Immediate Addressing

The word following the instruction contains the operand



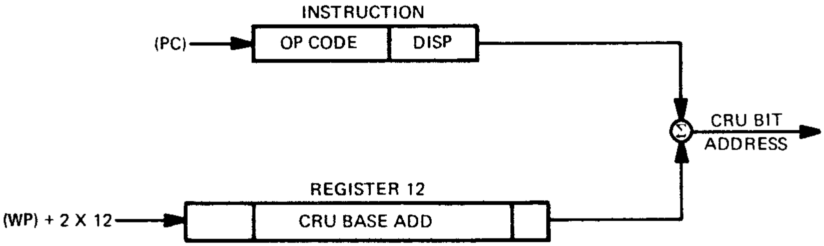
Program Counter Relative Addressing

The 8-bit signed displacement in the right byte (bits 8-15) of the instruction is multiplied by 2 and added to the updated contents of the program counter The result is placed in the PC



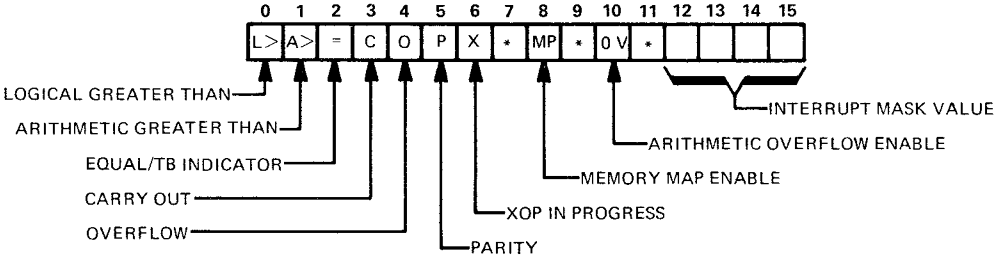
CRU Relative Addressing

The 8-bit signed displacement in the right byte of the instruction is added to the CRU base address (bits 3 14 of the workspace register 12) The result is the CRU address of the selected CRU bit



STATUS REGISTER MANIPULATION

Various SBP9989 machine instructions affect the status register The figure below shows the status register bit assignments and the following table lists the effects of the instructions on each status bit.



*These bits are functionally uncommitted and are available to the user.

TABLE 3 — STATUS REGISTER BIT DEFINITIONS

BIT	NAME	INSTRUCTION	CONDITION TO SET BIT TO 1
ST0	Logical Greater Than	C, CB	If MSB(SA) = 1 and MSB(DA) = 0, or if MSB(SA) = MSB(DA) and MSB OF [(DA) – (SA)] = 1.
		CI	If MSB(WR) = 1 and MSB of IOP = 0, or if MSB(WR) = MSB of IOP and MSB of [IOP – (WR)] = 1.
		ABS, LDCR	If (SA) ≠ 0.
		RTWP	If Bit (0) of WR15 is 1.
		LST	If Bit (0) of selected WR is 1.
		All others	If result ≠ 0.
ST1	Arithmetic Greater Than	C, CB	If MSB(SA) = 0 and MSB(DA) = 1, or if MSB(SA) = MSB(DA) and MSB of [(DA) – (SA)] = 1.
		CI	If MSB(WR) = 0 and MSB of IOP = 1, or if MSB(WR) = MSB of IOP and MSB of [IOP – (WR)] = 1.
		ABS, LDCR	If MSB(SA) = 0 and (SA) ≠ 0.
		RTWP	If Bit (1) of WR15 is 1.
		LST	If Bit (1) of selected WR is 1.
		All others	If MSB of result = 0 and result ≠ 0
ST2	Equal	C, CB	If (SA) = (DA).
		CI	If (WR) = IOP.
		COC	If (SA) and $\overline{(DA)}$ = 0.
		CZC	If (SA) and (DA) = 0.
		TB	If CRUIN = 1.
		ABS, LDCR	If (SA) = 0.
		RTWP	If Bit (2) of WR15 is 1.
		LST	If Bit (2) of selected WR is 1
		All others	If result = 0.

STATUS REGISTER BIT DEFINITIONS (Continued)

BIT	NAME	INSTRUCTION	CONDITION TO SET BIT TO 1
ST3	CARRY	A AB, ABS, AI,	
		DEC, DECT, INC	
		INCT NEG, S, SB	If CARRY OUT - 1
		SRA, SLA,	
		SRL SRC	If last bit shifted out - 1
		RTWP	If Bit (3) of WR15 is 1.
		LST	If Bit (3) of selected WR is 1
ST4	OVERFLOW	A, AB	If MSB(SA) = MSB(DA) and MSB of result ≠ MSB (DA)
		AI	If MSB(WR) = MSB of IOP and MSB of result ≠ MSB(WR)
		S, SB	If MSB(SA) ≠ MSB(DA) and MSB of result ≠ MSB(DA).
		DEC, DECT	If MSB(SA) = 1 and MSB of result = 0
		INC, INCT	If MSB(SA) = 0 and MSB of result = 1.
		SLA	If MSB changes during shift
		DIV	If MSB(SA) = 0 and MSB(DA) = 1 or if MSB(SA) = MSB(DA) and MSB of [(DA) − (SA)] = 0.
		DIVS	If (SA) = 0000 or if MSB(SA) ≠ MSB(WR0) and $\|(2^{15} + 1) \times (SA)\| <$ WR(0,1)
		ABS, NEG	If (SA) = 8000_{16}.
		RTWP	If Bit (4) of WR15 is 1.
		LST	If Bit (4) of selected WR is 1.
ST5	PARITY	CB, MOVB	If (SA) had odd number of 1's.
		LDCR	If 1 ≤ C ≤ 8 and (SA) has odd number of 1 s.
		AB, SB, SOCB, SZCB	If result has odd number of 1's.
		RTWP	If Bit (5) of WR15 is 1.
		LST	If Bit (5) of selected WR is 1
		STCR	If 1 ≤ C ≤ 8 and result has odd number of 1 s.
ST6	XOP	XOP	If XOP instruction is executed
		RTWP	If Bit (6) of WR15 is 1.
		LST	If Bit (6) of selected WR is 1.
ST7, ST9	User defined	RTWP	If corresponding bit of WR15 is 1 or
or ST11		LST	If corresponding bit of selected WR is 1.
ST8	MEMORY MAP	RTWP	If Bit (8) of WR15 is 1.
		LST	If Bit (8) of selected WR is 1.
ST10	ARITHMETIC	RTWP	If Bit (10) of WR15 is 1.
	OVERFLOW	LST	If Bit (10) of selected WR is 1.
	ENABLE		
ST12	INTERRUPT	LIMI	If corresponding bit of IOP is 1.
THRU	MASK	RTWP	If corresponding bit of WR15 is 1.
ST15		LST	If corresponding bit of selected WR is 1.

NOTE Interrupt $\overline{\text{LOAD}}$, XOPs, ILLOPs, and $\overline{\text{RESET}}$ operations sets Bits (7 11) to 0

INSTRUCTIONS

Dual-Operand Instructions with Multiple Addressing for Source and Destination Operands

0 1 2	3	4 5	6 7 8 9	10 11	12 13 14 15
OP CODE	B	T_D	D	T_S	S

If B = 1, the operands are bytes and the operand addresses are byte addresses. If B = 0, the operands are words and the operand addresses are word addresses The addressing mode for each operand is determined by the T field of that operand.

T_S or T_D	S or D	ADDRESSING MODE	NOTES
00	0 15	Workspace register	1
01	0 - 15	Workspace register indirect	
10	0	Symbolic	4
10	1 - 15	Indexed	2, 4
11	0 - 15	Workspace register indirect auto-increment	3

NOTES 1 Wnen a workspace register is the operand of a byte instruction (Bit (3) = 1), the most significant (left) byte (Bits (0 7)) is the operand and the least significant (right) byte (Bits (8 15)) remains unchanged

2. Workspace register 0 may not be used for indexing

3 The workspace register is incremented by 1 for byte instructions (Bit (3) = 1) and is incremented by 2 for word instructions (Bit (3) = 0)

4 When T_S and T_D = 10, two words are required in addition to the instruction word The first word is the source operand base address and the second word is the destination operand base address

MNEMONIC	MEANING	OP CODE	B	RESULT COMPARED TO 0	STATUS BITS AFFECTED	DESCRIPTION
A	Add	101	0	Yes	0 4	(SA) + (DA) → (DA)
AB	Add bytes	101	1	Yes	0 5	(SA) + (DA) → (DA)
C	Compare	100	0	No	0 - 2	Compare (SA) to (DA) and set appropriate status bits
CB	Compare bytes	100	1	No	0 - 2, 5	Compare (SA) to (DA) and set appropriate status bits
S	Subtract	011	0	Yes	0 4	(DA) – (SA) → (DA)
SB	Subtract bytes	011	1	Yes	0 - 5	(DA) – (SA) → (DA)
SOC	Set ones corresponding	111	0	Yes	0 - 2	(DA) OR (SA) → (DA)
SOCB	Set ones corresponding bytes	111	1	Yes	0 - 2, 5	(DA) OR (SA) → (DA)
SZC	Set zeros corresponding	010	0	Yes	0 - 2	(DA) AND $\overline{(\mathrm{SA})}$ → (DA)
SZCB	Set zeros corresponding bytes	010	1	Yes	0 2, 5	(DA) AND $\overline{(\mathrm{SA})}$ → (DA)
MOV	Move	110	0	Yes	0 - 2	(SA) → (DA)
MOVB	Move bytes	110	1	Yes	0 - 2, 5	(SA) → (DA)

Dual Operand Instructions with Multiple Addressing for Source Operand and Workspace Register Addressing for the Destination

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP CODE						D				T_S		S			

The addressing mode for the.source operand is determined by the T_S field

T_S	S	ADDRESSING MODE	NOTES
00	0 - 15	Workspace register	
01	0 15	Workspace register indirect	
10	0	Symbolic	
10	1 15	Indexed	1
11	0 15	Workspace register indirect auto increment	2

NOTES 1 Workspace register 0 may not be used for indexing

2 The workspace register is incremented by 2

MNEMONIC	MEANING	OP CODE	RESULT COMPARED TO 0	STATUS BITS AFFECTED	DESCRIPTION
COC	Compare ones corresponding	001000	No	2	Test (D) to determine if 1 s are in each bit position where 1 s are in (SA) If so, set ST2
CZC	Compare zeros corresponding	001001	No	2	Test (D) to determine if 0 s are in each bit position where 1's are in (SA) If so, set ST2
XOR	Exclusive OR	001010	Yes	0 - 2	(D)⊕(SA) → (D)
MPY	Multiply	001110	No		Multiply unsigned (D) by unsigned (SA) and place unsigned 32 bit product in D (most significant) and D + 1 (least significant) If WR15 is D, the next word in memory after WR15 will be used for the least significant half of the product
DIV	Divide	001111	No	4	If unsigned (SA) is less than or equal to unsigned (D), perform no operation and set ST4 Otherwise, divide unsigned (D) and (D + 1) by unsigned (SA) Quotient → (D), remainder → (D + 1) If D is WR15, the next word in memory after WR15 will be used for the remainder

Signed Multiply and Divide Instructions

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP CODE										T_S		S			

The addressing mode for the source operand is determined by the T_S field.

T_S	S	ADDRESSING MODE	NOTES
00	0 - 15	Workspace register	1
01	0 - 15	Workspace register indirect	1
10	0	Symbolic	1
10	1 - 15	Indexed	1, 2
11	0 - 15	Workspace register indirect auto increment	1, 3

NOTES 1 Workspace registers 0 and 1 contain operands used in the signed multiply and divide operations
2 Workspace register 0 may not be used for indexing
3 The workspace register is incremented by 2

MNEMONIC	MEANING	OP CODE	RESULT COMPARED TO 0	STATUS BITS AFFECTED	DESCRIPTION
MPYS	Signed Multiply	0000000111	Yes	0 2	Multiply signed 2's-complement integer in WR0 by signed 2's complement integer (SA) and place signed 32-bit product in WR0 (most significant) and WR1 (least significant)
DIVS	Signed Divide	0000000110	Yes	0 - 2, 4	If (SA) = 0000 or if MSB (SA) = MSB (WR0) and $\lvert 2^{15} \times (SA) \rvert \leq \lvert WR(0, 1) \rvert$ or if MSB (SA) ≠ MSB (WR0) and $\lvert (2^{15} + 1) \times (SA) \rvert \leq \lvert WRO\ (0, 1) \rvert$, set ST4 Otherwise, divide signed 2's-complement integer in WRO and WR1 by the signed 2's complement integer (SA) and place the signed quotient in WR0 and the signed remainder in WR1 The sign of the quotient is determined by algebraic rules The sign of the remainder is the same as the sign of the dividend

Single Operand Instructions

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP CODE										T_S		S			

The T_S and S fields provide multiple mode addressing capability for the source operand

MNEMONIC	MEANING	OP CODE	RESULT COMPARED TO 0	STATUS BITS AFFECTED	DESCRIPTION
B	Branch	0000010001	No		SA (PC)
BL	Branch and link	0000011010	No		(PC) → (WR11) SA → (PC)
BLWP	Branch and load workspace pointer	0000010000	No		(SA) → (WP), (SA + 2) → (PC) (old WP) → (new WR13), (old PC) → (new WR14) (old ST) → (new WR15) the interrupt input ($\overline{\text{INTREQ}}$) is not tested upon completion of the BLWP instruction
CLR	Clear operand	0000010011	No		0 → (SA)
SETO	Set to ones	0000011100	No		$FFFF_{16}$ (SA)
INV	Invert	0000010101	Yes	0 2	$\overline{(SA)}$ (SA)
NEG	Negate	0000010100	Yes	0 4	(SA) (SA)
ABS	Absolute value*	0000011101	No	0 4	|(SA)| → (SA)
SWPB	Swap bytes	0000011011	No		Bits (0 7) of SA → Bits (8 15) of SA Bits (8 15) of SA → Bits (0 7) of SA
INC	Increment	0000010110	Yes	0 4	(SA) + 1 → (SA)
INCT	Increment by 2	0000010111	Yes	0 4	(SA) + 2 → (SA)
DEC	Decrement	0000011000	Yes	0 4	(SA) 1 (SA)
DECT	Decrement by 2	0000011001	Yes	0 4	(SA) 2 (SA)
X**	Execute	0000010010	No		Execute the instruction at SA

*Operand is compared to zero for status bit

**If additional memory words for the execute instruction are required to define the operands of the instruction located at SA these words will be accessed from PC and the PC will be updated accordingly The instruction acquisition signal (IAQ) will not be true when the SBP9989 accesses the instruction at SA Status bits are affected in the normal manner for the instruction executed

CRU Multiple Bit Instructions

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP CODE						C				T_S		S			

The C field specifies the number of bits to be transferred If C = 0, 16 bits will be transferred The CRU base register (WR12, Bits 3 14) defines the starting CRU bit address The bits are transferred serially and the CRU address is incremented with each bit transfer, although the contents of WR12 are not affected T_S and S provide multiple mode addressing capability for the source operand If 8 or fewer bits are transferred (1 ≤ C ≤ 8), the source address is a byte address If 9 or more bits are transferred (C 0, C ≥ 9), the source address is a word address If the source is addressed in the workspace register indirect auto increment mode, the workspace register is incremented by 1 ≤ C ≤ 8, and is incremented by 2 otherwise

MNEMONIC	MEANING	OP CODE	RESULT COMPARED TO 0	STATUS BITS AFFECTED	DESCRIPTION
LDCR	Load communication register	001100	Yes	0 2 5*	Beginning with LSB of (SA), transfer the specified number of bits from (SA) to the CRU
STCR	Store communication register	001101	Yes	0 2,5*	Beginning with LSB of (SA), transfer the specified number of bits from the CRU to (SA) Load unfilled bit positions with 0

*ST5 is affected only if 1 ≤ C ≤ 8

CRU Single Bit Instructions

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP CODE								SIGNED DISPLACEMENT							

CRU relative addressing is used to address the selected CRU bit

MNEMONIC	MEANING	OP CODE	STATUS BITS AFFECTED	DESCRIPTION
SBO	Set bit to one	00011101		Set the selected CRU output bit to 1
SBZ	Set bit to zero	00011110		Set the selected CRU output bit to 0
TB	Test bit	00011111	2	If the selected CRU input bit 1 set ST2 to 1 If the selected CRU input bit 0 set ST2 to 0

Jump Instructions

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP CODE								DISPLACEMENT							

Jump instructions cause the PC to be loaded with the value selected by PC relative addressing if the status register condition is met Otherwise, no operation occurs and the next instruction is executed since the PC points to the next instruction The displacement field in 2's-complement form is a word count to be added to PC Thus, the jump instruction has a range of −128 to 127 words from the memory word address following the jump instruction. No ST bits are affected by jump instructions

MNEMONIC	MEANING	OP CODE	STATUS REGISTER CONDITION TO LOAD PC
JEQ	Jump equal	00010011	ST2 = 1
JGT	Jump greater than	00010101	ST1 = 1
JH	Jump high	00011011	ST0 = 1 and ST2 = 0
JHE	Jump high or equal	00010100	ST0 = 1 or ST2 = 1
JL	Jump low	00011010	ST0 = 0 and ST2 = 0
JLE	Jump low or equal	00010010	ST0 = 0 or ST2 - 1
JLT	Jump less than	00010001	ST1 = 0 and ST2 = 0
JMP	Jump unconditional	00010000	unconditional
JNC	Jump no carry	00010111	ST3 = 0
JNE	Jump not equal	00010110	ST2 = 0
JNO	Jump no overflow	00011001	ST4 = 0
JOC	Jump on carry	00011000	ST3 = 1
JOP	Jump odd parity	00011100	ST5 = 1

Shift Instructions

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP CODE								C				WR			

If C ≠ 0, Bits 12 - 15 of WR0 contain the shift count. If C = 0 and Bits 12 - 15 of WR0 = 0, the shift count is 16.

MNEMONIC	MEANING	OP CODE	RESULT COMPARED TO 0	STATUS BITS AFFECTED	DESCRIPTION
SLA	Shift left arithmetic	00001010	Yes	0 - 4	Shift (WR) left Fill vacated bit positions with 0
SRA	Shift right arithmetic	00001000	Yes	0 - 3	Shift (WR) right. Fill vacated bit positions with original MSB of (WR)
SRC	Shift right circular	00001011	Yes	0 3	Shift (WR) right Shift previous LSB into MSB
SRL	Shift right logical	00001001	Yes	0 - 3	Shift (WR) right. Fill vacated bit positions with 0's

Immediate Register Instructions

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP CODE											▨	WR			
IOP															

MNEMONIC	MEANING	OP CODE	RESULT COMPARED TO 0	STATUS BITS AFFECTED	DESCRIPTION
AI	Add immediate	00000010001	Yes	0 - 4	(WR) + IOP → (WR)
ANDI	AND immediate	00000010010	Yes	0 - 2	(WR) AND IOP → (WR)
CI	Compare immediate	00000010100	No	0 - 2	Compare (WR) to IOP and set appropriate status bits
LI	Load immediate	00000010000	Yes	0 - 2	IOP → (WR)
ORI	OR immediate	00000010011	Yes	0 - 2	[(WR) OR IOP] → (WR)

Internal Register Load Immediate Instructions

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP CODE											▨				
IOP															

MNEMONIC	MEANING	OP CODE	DESCRIPTION
LWPI	Load workspace pointer immediate	00000010111	IOP → (WP), no ST bits affected
LIMI	Load interrupt mask immediate	00000011000	Bits (12 15) of IOP → ST (12 15)

Internal Register Load and Store Instructions

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP CODE												WR			

MNEMONIC	MEANING	OP CODE	STATUS BITS AFFECTED	DESCRIPTION
STST	Store status register	00000010110X		(ST) → (WR)
LST	Load status register	000000001000	0 15	(WR) → (ST)
STWP	Store workspace pointer	00000010101X		(WP) → (WR)
LWP	Load workspace pointer	000000001001		(WR) → (WP)

X ≡ don't care

Extended Operation (XOP) Instruction

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	1	0	1	1	D				T_S		S			

The T_S and S fields provide multiple-mode addressing capability for the source operand When the XOP is executed, ST6 is set and the following transfers occur

$$\overline{\text{MPEN}} \rightarrow 1$$
$$(40_{16} + 4 \times \text{D}) \rightarrow (\text{WP})$$
$$(42_{16} + 4 \times \text{D}) \rightarrow (\text{PC})$$
$$(\text{ST7} \quad \text{ST11}) \rightarrow 00000$$
$$\text{SA} \rightarrow (\text{New WR11})$$
$$(\text{Old WP}) \rightarrow (\text{New WR13})$$
$$(\text{Old PC}) \rightarrow (\text{New WR14})$$
$$(\text{Old ST}) \rightarrow (\text{New WR15})$$

The SBP9989 does not test interrupt requests ($\overline{\text{INTREQ}}$) upon completion of the XOP instruction

Return Workspace Pointer (RTWP) Instructions

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0	0	0	0	0	1	1	1	0	0	▨				

The RTWP instruction causes the following transfers to occur·

(WR15) → (ST)
(WR14) → (PC)
(WR13) → (WP)

External Instructions

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
OP CODE											C				

External instructions cause the three most significant address lines (A0 A2) to be set to the levels described below, address lines A3 · A7 to be set to the 5-bit value specified in the C field of the instruction, the DBIN output to be activated, and the CRUCLK line to be pulsed, allowing external control functions to be initiated.

MNEMONIC†	MEANING	OP CODE	STATUS BITS AFFECTED	DESCRIPTION	ADDRESS BUS		
					A0	A1	A2
IDLE	Idle	00000011010		Suspend SBP9989 instruction execution until an Interrupt, $\overline{\text{LOAD}}$ or $\overline{\text{RESET}}$ occurs	L	H	L
RSET	Reset	00000011011	7 15	0 → ST (7 15)	L	H	H
CKOF	User defined	00000011110			H	H	L
CKON	User defined	00000011101			H	L	H
LREX	User defined	00000011111			H	H	H

†The mnemonics associated with these instructions relate to their use in the TI 990/4 minicomputer and have no special significance

Microinstruction Cycle

The SBP9989 includes circuitry which will indicate the completion of a microinstruction cycle. Designated as the $\overline{\text{CYCEND}}$ function, it provides CPU status that can simplify system design. The $\overline{\text{CYCEND}}$ output will go to a low logic level as a result of the low-to-high transition of each clock pulse which initiates the last clock cycle of a microinstruction

nstruction Execution Times

Instruction execution times for the SBP9989 are a function of

1) Clock cycle time, t_C
2) Addressing mode used where operands have multiple addressing mode capability
3) Number of wait states required per memory access
4) Numbei of wait states required per CRU operation

The following instruction execution listing provides the number of clock cycles memory-access cycles and CRU operations required to execute each SBP9989 instruction For instructions with multiple addressing modes for either or both operands the table lists the number of clock cycles and memory access cycles with all operands addressed in the workspace register mode To determine the additional number of clock cycles and memory-access cycles required for modified addressing, add the appropriate values from Table 4 For the five CRU instructions (i e , STCR LDCR, SBO, SBZ, TB), the table lists the number of clock cycles assuming no wait states for CRU operations To determine the additional number of CRU-related clock cycles, add one clock cycle for each wait state incurred as the result of a CRU operation The total execution time for an instruction is given by

$$T = t_C\ [C_T + (W_M \times M_T)]\ + t_C\ (W_C \times P)$$

where

T = total instruction execution time,

t_C = clock cycle time,

C_T = total number of clock cycles (clock cycles for instruction execution plus clock cycles for address modification),

W_M = number of required wait states per memory access,

M_T = total number of memory-accesses (memory accesses for instruction execution plus memory accesses for address modification),

P = number of CRU operations,

W_C = number of required wait states per CRU operation

As an example, the instruction MOVB is used in a system with t_C = 0 250 μs and no wait states are required to access memory. Both operands are addressed in the workspace register mode. The instruction execution time is given by

$$\begin{aligned} T &= t_C\ [C_T + (W_M \times M_T)]\ + t_C\ (W_C \times P) \\ &= 0.250\ [12 + (0 \times 4)]\ + 0\ 250\ (0) = 3\ \mu s \end{aligned}$$

If two wait states per memory access were required, the execution time would become

$$T = 0\ 250\ [12 + (2 \times 4)]\ + 0\ 250\ (0) = 5\ \mu s$$

If the source operand were addressed in the symbolic mode and two wait states were required

$$\begin{aligned} T &= t_C\ [C_T + (W_M \times M_T)]\ + t_C\ (W_C \times P) \\ C_T &= 12 + 6 = 18 \\ M_T &= 4 + 1 = 5 \\ T &= 0\ 250\ [18 + (2 \times 5)]\ + 0\ 250\ (0) = 7\ \mu s \end{aligned}$$

INSTRUCTION EXECUTION

INSTRUCTION	CLOCK CYCLES		MEMORY CYCLES		ADDRESS MODIFICATION		CRU OPERATIONS
	SBP9989	Δ FROM SBP9900A	SBP9989	Δ FROM SBP9900A	SOURCE	DEST.	P
A	12	−2	4		Table 4	Table 4	
AB	12	−2	4		Table 4	Table 4	
ABS (MSB=0)	10	−2	2		Table 4		
(MSB=1)	14		3		Table 4		
AI	14		4				
ANDI	14		4				
B	6	−2	1	−1	Table 4		
BL	10	−2	2	−1	Table 4		
BLWP	24	−2	6		Table 4		
C	12	−2	3		Table 4	Table 4	
CB	12	−2	3		Table 4	Table 4	
CI	12	−2	3				
CKOF	10	−2	1				
CKON	10	−2	1				
CLR	8	−2	2	−1	Table 4		
COC	12	−2	3		Table 4		
CZC	12	−2	3		Table 4		
DEC	10		3		Table 4		
DECT	12		3		Table 4		
DIV (ST4 is set)	20	+4	4	+1	Table 4		
(ST4 is reset)	56	−38 to −68	6		Table 4		
DIVS (ST4 is set)	56	new	4		Table 4		
(ST4 is reset)	60	new	6		Table 4		
IDLE	10	−2	1				
INC	10		3		Table 4		
INCT	10		3		Table 4		
INV	10		3		Table 4		
JUMP (PC is changed)	6	−4	1				
(PC is not changed)	6	−2	1				
LDCR (C=0)	48	−4	3		Table 4		16
(1 ≤ C ≤ 15)	16 + 2C	−4	3		Table 4		C
LI	12		3				
LIMI	12	−2	2				
LREX	10	−2	1				
LST	10	new	2				
LWP	10	new	2				
LWPI	12	+2	2				
MOV	10	−4	3	−1	Table 4	Table 4	
MOVB	12	−2	4		Table 4	Table 4	
MPY	52		5		Table 4		
MPYS	56	new	5		Table 4		
NEG	12		3		Table 4		
ORI	14		4				
RSET	10	−2	1				
RTWP	16	+2	4				

INSTRUCTION	CLOCK CYCLES		MEMORY CYCLES		ADDRESS MODIFICATION		CRU OPERATIONS P
	SBP9989	Δ FROM SBP9900A	SBP9989	Δ FROM SBP9900A	SOURCE	DEST	
S	12	−2	4		Table 4	Table 4	
SB	12	−2	4		Table 4	Table 4	
SBO	12		2				1
SBZ	12		2				1
SETO	8	−2	2	−1	Table 4		
SHIFTS							
(C≠0)	12 + 2C		3				
(C=0, Bits (12 - 15)							
of WR0 = 0)	52		4				
(C=0, Bits (12 15)							
of WR0 ≠ 0)	Note 1		4				
SOC	12	−2	4		Table 4	Table 4	
SOCB	12	−2	4		Table 4	Table 4	
STCR (C=0)	56	−4	4		Table 4		16
(1 ⩽ C ⩽ 8)	40	−2 to −4	4		Table 4		C
(9 ⩽ C ⩽ 15)	56	−2	4		Table 4		C
STST	8		2				
STWP	8		2				
SWPB	10		3		Table 4		
SZC	12	−2	4		Table 4	Table 4	
SZCB	12	−2	4		Table 4	Table 4	
TB	12		2				1
X*	4		1		Table 4		
XOP	28	−8	7	−1			
XOR	12	−2	4		Table 4		
Reset function	22	−4	5				
Load function	20	−2	5				
Interrupt							
Context Switch	20	−2	5				
UNDEFINED OP CODES **							
0000_{16}-$007F_{16}$							
$00A0_{16}$-$017F_{16}$							
0320_{16}-$033F_{16}$	24	+18	6	+5			
0780_{16}-$07FF_{16}$							
$0C00_{16}$-$0FFF_{16}$							

*Execution time is added to the execution time of the source address

**Execution time includes time to perform a trap (i e , subroutine call) operation resulting from $\overline{\text{XIPP}}$ being inactive

NOTE 1 The number of clock cycles is twenty plus twice the value of Bits (12 15) of WR0

TABLE 4 — ADDRESS MODIFICATION

ADDRESSING MODE	CLOCK CYCLES		MEMORY CYCLES
	SBP9989	Δ FROM SBP9900A	
WR (T_S or T_D = 00)	0		0
WR indirect (T_S or T_D = 01)	4		1
WR indirect auto-increment (T_S or T_D = 11)	6	0 to −2	2
Symbolic (T_S or T_D = 10, S or D = 0)	6	−2	1
Indexed (T_S or T_D = 10, S or D = 0)	6	−2	2

MACHINE CYCLES

This section completes the description of instruction execution, by giving the individual instruction execution cycles. Each machine cycle consists of two or more clock cycles (depending upon addressing mode) as defined herein. Three categories describe the 9989 machine cycles: ALU cycle, Memory cycle and CRU cycle.

ALU Cycle

The ALU cycle performs an internal operation of the microprocessor. The memory interface control signals and CRU control signals are not affected by the execution of an ALU cycle, which takes two clock cycles to execute.

Memory Cycle

The memory cycle primarily performs a data transfer between the microprocessor and the external memory device. Appropriate memory bus control signals are generated by the microprocessor as a result of a memory cycle execution. The memory cycle takes 2 + W (where W is number of wait states) clock cycles to execute.

CRU Cycle

The CRU cycle performs a bit transfer between the microprocessor and I/O devices It takes two clock cycles to execute. The address of the CRU bit is set up during the first clock cycle. For an input operation the CRUIN line is sampled by the microprocessor during the third clock cycle For an output operation, the data bit is set up on the CRUOUT line at the same time the address is set up. The CRUCLK line is pulsed during the second clock cycle of CRU output cycle.

A special feature of the 9989's CRU operation not available on its forerunner the SBP9900A, is the capability to insert wait states. The 9989 will sample the READY line at the beginning of the second clock cycle.

SBP9989 MACHINE CYCLE SEQUENCES

Most SBP9989 instructions consist of two parts 1) the data derivation and 2) operation execution The data derivation sequence depends on the addressing mode for the data Since the addressing modes are common to all instructions, the data derivation sequence is the same for the same addressing mode, regardless of the instruction Therefore data derivation sequences are described first These are then referenced in appropriate sequence in the instruction execution description.

Data Derivation Sequence

WORKSPACE REGISTER

CYCLE	TYPE	NOTE
0	Memory read	Fastest addressing mode, no additional clock cycles for Source or Destination Acquisition Read will either have already occurred in the op-code instruction Fetch or at appropriate operand Fetch Therefore Ns = Nd = 0 when using Workspace Register Addressing Mode

WORKSPACE REGISTER INDIRECT

CYCLE	TYPE	NOTE
1	Memory read	Ns = Nd = 2
2	ALU	

WORKSPACE REGISTER INDIRECT AUTO-INCREMENT

CYCLE	TYPE	NOTE
1	Memory read	Ns = Nd = 3
2	ALU	
3	Memory write	

SYMBOLIC

CYCLE	TYPE	NOTE
1	ALU	Ns = Nd = 3
2	Memory read	
3	ALU	

INDEXED

CYCLE	TYPE	NOTE
1	Memory read	Ns = Nd = 3
2	Memory read	
3	ALU	

INSTRUCTION EXECUTION SEQUENCE

A, AB, MOVB, S, SB, SOC, SOCB, SZC, SZCB

CYCLE	TYPE
1	Memory read
2	ALU
Ns	Source acquisition
3 + Ns	Memory read
Nd	Destination acquisition
4 + Ns + Nd	Memory read
5 + Ns + Nd	ALU
6 + Ns + Nd	Memory write

ABS (MSB = 1)

CYCLE	TYPE
1	Memory read
2	ALU
Ns	Source acquisition
3 + Ns	Memory read
4 + Ns	ALU
5 + Ns	ALU
6 + Ns	ALU
7 + Ns	Memory write

ABS (MSB = 0)

CYCLE	TYPE
1	Memory read
2	ALU
Ns	Source acquisition
3 + Ns	Memory read
4 + Ns	ALU
5 + Ns	ALU

AI, ANDI, ORI

CYCLE	TYPE
1	Memory read
2	ALU
3	ALU
4	Memory read
5	Memory read
6	ALU
7	Memory write

B

CYCLE	TYPE
1	Memory read
2	ALU
Ns	Source acquisition
3 + Ns	ALU

BL

CYCLE	TYPE
1	Memory read
2	ALU
Ns	Source acquisition
3 + Ns	ALU
4 + Ns	ALU
5 + Ns	Memory write

BLWP

CYCLE	TYPE
1	Memory read
2	ALU
Ns	Source acquisition
3 + Ns	Memory read
4 + Ns	ALU
5 + Ns	ALU
6 + Ns	Memory write
7 + Ns	ALU
8 + Ns	Memory write
9 + Ns	ALU
10 + Ns	Memory write
11 + Ns	Memory read
12 + Ns	ALU

C, CB, COC, CZC

CYCLE	TYPE
1	Memory read
2	ALU
Ns	Source acquisition
3 + Ns	Memory read
Nd	Destination acquisition
4 + Ns + Nd	Memory read
5 + Ns + Nd	ALU
6 + Ns + Nd	ALU

CI

CYCLE	TYPE
1	Memory read
2	ALU
3	Memory read
4	Memory read
5	ALU
6	ALU

CKOF, CKON, LREX, IDLE, RSET

CYCLE	TYPE
1	Memory read
2	ALU
3	ALU
4	ALU
5	CRU

CLR

CYCLE	TYPE
1	Memory read
2	ALU
Ns	Source acquisition
3 + Ns	ALU
4 + Ns	Memory write

DEC, DECT, INC, INCT, INV, SWPB

CYCLE	TYPE
1	Memory read
2	ALU
Ns	Source acquisition
3 + Ns	Memory read
4 + Ns	ALU
5 + Ns	Memory write

DIV (ST4 = 0)

CYCLE	TYPE
1	Memory read
2	ALU
Ns	Source acquisition
3 + Ns	Memory read
4 + Ns	Memory read
5 + Ns	ALU
6 + Ns	ALU
7 + Ns	Memory read
8 + Ns	ALU
9 + Ns	ALU
10 + Ns	(14 ALU cycles)
24 + Ns	ALU
25 + Ns	ALU
26 + Ns	Memory write
27 + Ns	ALU
28 + Ns	Memory write

DIV (ST4 = 1)

CYCLE	TYPE
1	Memory read
2	ALU
Ns	Source acquisition
3 + Ns	Memory read
4 + Ns	Memory read
5 + Ns	ALU
6 + Ns	ALU
7 + Ns	Memory read
8 + Ns	ALU
9 + Ns	ALU
10 + Ns	ALU

DIVS (ST4 = 0)

CYCLE	TYPE
1	Memory read
2	ALU
Ns	Source acquisition
3 + Ns	Memory read
4 + Ns	ALU
5 + Ns	Memory read
6 + Ns	ALU
7 + Ns	Memory read
8 + Ns	ALU
9 + Ns	ALU
10 + Ns	(14 ALU cycles)
24 + Ns	ALU
25 + Ns	ALU
26 + Ns	ALU
27 + Ns	ALU
28 + Ns	Memory write
29 + Ns	ALU
30 + Ns	Memory write

DIVS (ST4 = 1)

CYCLE	TYPE
1	Memory read
2	ALU
Ns	Source acquisition
3 + Ns	Memory read
4 + Ns	ALU
5 + Ns	Memory read
6 + Ns	ALU
7 + Ns	Memory read
8 + Ns	ALU
9 + Ns	ALU
10 + Ns	(14 ALU cycles)
24 + Ns	ALU
25 + Ns	ALU
26 + Ns	ALU
27 + Ns	ALU
28 + Ns	Memory write

ILLOP

CYCLE	TYPE
1	Memory read
2	ALU
3	ALU
4	Memory read
5	ALU
6	Memory write
7	ALU
8	Memory write
9	ALU
10	Memory write
11	Memory read
12	ALU

JUMPS

CYCLE	TYPE
1	Memory read
2	ALU
3	ALU

LDCR

CYCLE	TYPE
1	Memory read
2	ALU
Ns	Source acquisition
3 + Ns	Memory read
4 + Ns	ALU
5 + Ns	ALU
6 + Ns	Memory read
7 + Ns	ALU
Nc	CRU (for C = 0, Nc = 16) (for 1 ⩽ C ⩽ 15, Nc = C)
8 + Ns + Nc	ALU

LI

CYCLE	TYPE
1	Memory read
2	ALU
3	ALU
4	Memory read
5	ALU
6	Memory write

LIMI, LWPI

CYCLE	TYPE
1	Memory read
2	ALU
3	ALU
4	Memory read
5	ALU
6	ALU

LOAD, INTERRUPT CONTEXT SWITCH

CYCLE	TYPE
1	ALU
2	Memory read
3	ALU
4	Memory write
5	ALU
6	Memory write
7	ALU
8	Memory write
9	Memory read
10	ALU

LST, LWP

CYCLE	TYPE
1	Memory read
2	ALU
3	Memory read
4	ALU
5	ALU

MOV (word)

CYCLE	TYPE
1	Memory read
2	ALU
Ns	Source acquisition
3 + Ns	Memory read
Nd	Destination acquisition
4 + Ns + Nd	ALU
5 + Ns + Nd	Memory write

MPY (unsigned)

CYCLE	TYPE
1	Memory read
2	ALU
Ns	Source acquisition
3 + Ns	Memory read
4 + Ns	Memory read
5 + Ns	ALU
6 + Ns	ALU
7 + Ns	(16 ALU cycles)
23 + Ns	Memory write
24 + Ns	ALU
25 + Ns	ALU
26 + Ns	Memory write

MPYS (signed)

CYCLE	TYPE
1	Memory read
2	ALU
Ns	Source acquisition
3 + Ns	Memory read
4 + Ns	ALU
5 + Ns	ALU
6 + Ns	Memory read
7 + Ns	ALU
8 + Ns	(15 ALU cycles)
23 + Ns	ALU
24 + Ns	Memory write
25 + Ns	ALU
26 + Ns	ALU
27 + Ns	ALU
28 + Ns	Memory write

NEG

CYCLE	TYPE
1	Memory read
2	ALU
Ns	Source acquisition
3 + Ns	Memory read
4 + Ns	ALU
5 + Ns	ALU
6 + Ns	Memory write

$\overline{\text{RESET}}$ (hardware)

CYCLE	TYPE
1	ALU
2	ALU
3	Memory read
4	ALU
5	Memory write
6	ALU
7	Memory write
8	ALU
9	Memory write
10	Memory read
11	ALU

RTWP

CYCLE	TYPE
1	Memory read
2	ALU
3	ALU
4	Memory read
5	Memory read
6	Memory read
7	ALU
8	ALU

SBO, SBZ, TB

CYCLE	TYPE
1	Memory read
2	ALU
3	ALU
4	Memory read
5	ALU
6	CRU

SETO

CYCLE	TYPE
1	Memory read
2	ALU
Ns	Source acquisition
3 + Ns	ALU
4 + Ns	Memory write

SHIFTS (C = 0)

CYCLE	TYPE
1	Memory read
2	ALU
3	Memory read
4	ALU
5	ALU
6	Memory read
7	ALU
8	ALU
Nc	(If WR0(Bits 12 15) = 0 Nc 16) (otherwise Nc WR0(Bits 12 15)
9 + Nc	Memory write
10 + Nc	ALU

SHIFTS (C ≠ 0)

CYCLE	TYPE
1	Memory read
2	ALU
3	Memory read
4	ALU
Nc	(Nc – C ALU cycles)
5 + Nc	Memory write
6 + Nc	ALU

STCR

CYCLE	TYPE
1	Memory read
2	ALU
Ns	Source acquisition
3 + Ns	Memory read
4 + Ns	ALU
5 + Ns	ALU
6 + Ns	Memory read
7 + Ns	ALU
Nc	CRU (for C = 0, 9 C ≤ 15, Nc = 16) (for 1 ≤ C < 8, Nc = 8)
8 + Ns + Nc	ALU
9 + Ns + Nc	ALU
10 + Ns + Nc	ALU
11 + Ns + Nc	ALU
12 + Ns + Nc	Memory write

STST, STWP

CYCLE	TYPE
1	Memory read
2	ALU
3	ALU
4	Memory write

X

CYCLE	TYPE
1	Memory read
2	ALU
Ns	Source acquisition
3 + Ns	Memory read

XOP

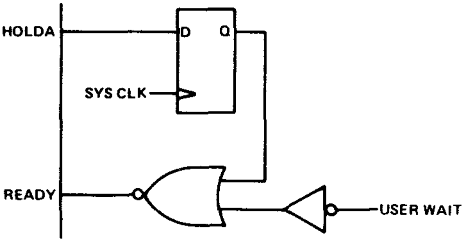
CYCLE	TYPE
1	Memory read
2	ALU
Ns	Source acquisition
3 + Ns	ALU
4 + Ns	Memory read
5 + Ns	ALU
6 + Ns	Memory write
7 + Ns	ALU
8 + Ns	Memory write
9 + Ns	ALU
10 + Ns	Memory write
11 + Ns	ALU
12 + Ns	Memory write
13 + Ns	Memory read
14 + Ns	ALU

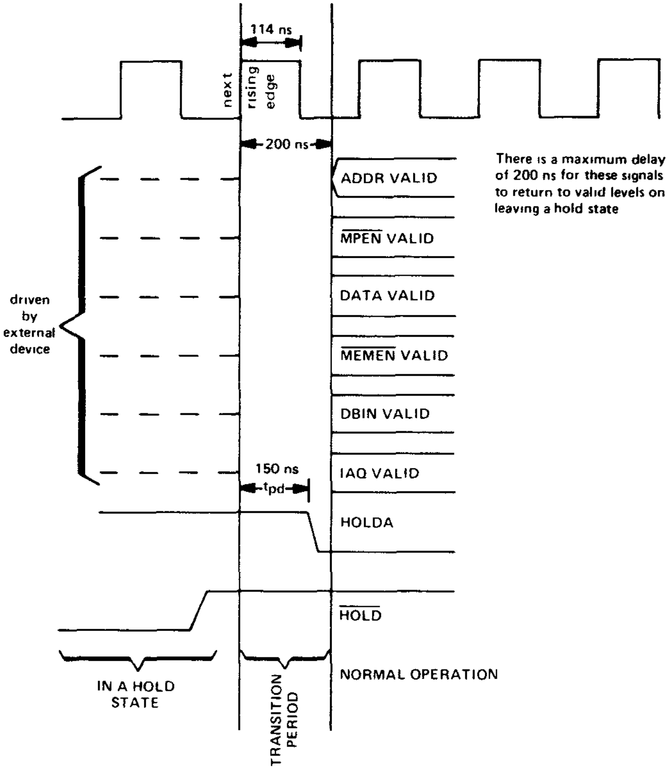
XOR

CYCLE	TYPE
1	Memory read
2	ALU
Ns	Source acquisition
3 + Ns	Memory read
4 + Ns	Memory read
5 + Ns	ALU
6 + Ns	Memory write

Leaving a Hold State

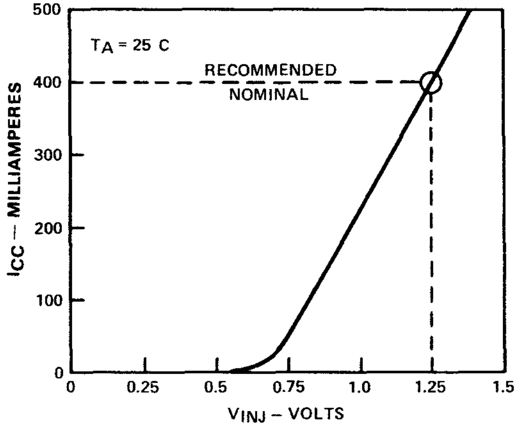
When the SBP9989 leaves a hold state, the time required for the signals to return to their proper levels is different from the other delay times This difference is on the order of 10 to 20 nanoseconds. If your system does not use hold states or can tolerate the longer delay times, there will be no problems If these delay times are too long, add a wait state at the end of a hold state to allow the signals to come to their proper levels as shown below.

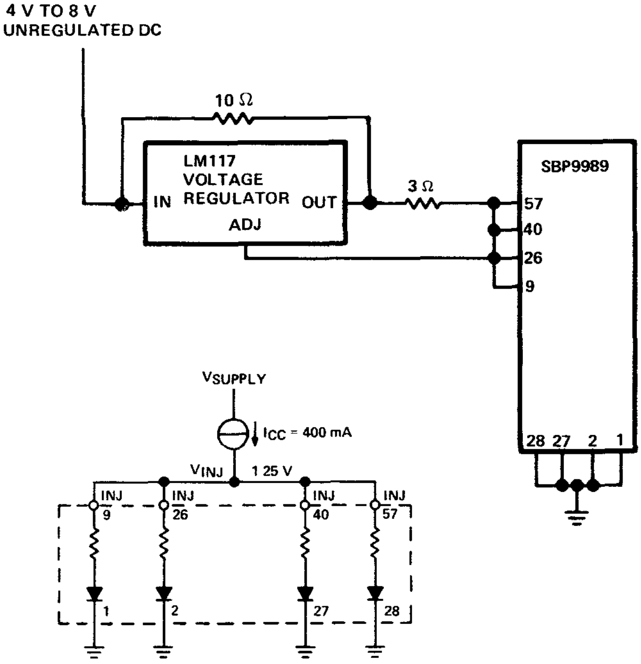




POWER SOURCE

I^2L is a current-injected logic. When the injector and ground pins are placed across a curve tracer, the processor V I characteristic will resemble that of a silicon diode Although any voltage or current source capable of supplying the desired current at the injector voltage required will suffice, a regulated current source is recommended This is because the injector voltage will vary over the temperature range One approach to a suitable, highly stable regulator is shown in the figure below.





ELECTRICAL DATA

INPUT/OUTPUT STRUCTURE

Interfacing

The input/output (I/O) accommodations have been designed for TTL compatibility. Direct interfacing is supportable by entire families of support devices. System designers will note that propagation delays, set-up time, and hold time requirements do not depend on the frequency of operation.

Input Circuit

An input clamping diode is incorporated to limit negative excursions (ringing) when the device is on the end of a transmission line. Since the input circuit is independent of injector current, input threshold compatibility is maintained over the entire speed/power range. This circuit provides a high impedance characteristic to reduce input loading and improve the low-logic level input noise immunity over some standard TTL inputs. Full compatibility is maintained with virtually all 5-volt-logic families even when the device is powered down (injector current reduced).

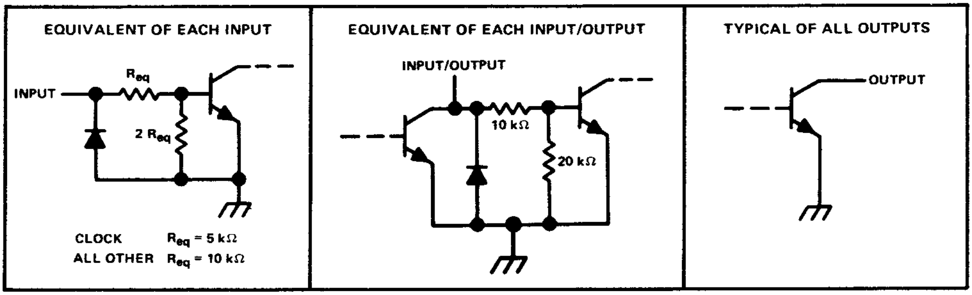
Terminating Unused Inputs

Inputs which are selected to be hardwired to a low logic level may be connected directly to ground. Inputs selected to be hardwired to a high logic level may be connected directly to +5 volts.

Output Circuit

The output circuit selected is an injection open-collector transistor Since this transistor is injected, output sourcing capability is directly related to injector current. In other words, the number of loads which may be sourced by an output is directly reduced as injector current is reduced.

EQUIVALENT SCHEMATICS



ABSOLUTE MAXIMUM RATINGS

Injection voltage, V_{INJ}	2*V
Injection current, I_{INJ}	800*mA
Input voltage, V_I	5 5 V
Operating case temperature range	−55°C to 125°C
Storage temperature range	−65°C to 150°C
Lead temperature for 10 seconds	300°C

*These values may require modification to observe the rated case temperature range Thermal resistance junction to-case is typically 13°C/W

RECOMMENDED OPERATING CONDITIONS

		MIN	NOM	MAX	UNIT
Supply current, I_{CC}		380	400	420	mA
High-level output voltage, V_{OH}				5 5	V
Low-level output current, I_{OL}				16	mA
Clock frequency, f_{clock}		0		4 4	MHz
Width of clock pulse, t_w		114			ns
Clock rise time, t_r				20	ns
Clock fall time, t_f				20	ns
Setup time, t_{su}	$\overline{\text{HOLD}}$	0			ns
	READY	25			
	D0 D15	45			
	CRUIN	70			
	$\overline{\text{INTREQ}}$	55			
	IC0 IC3	55			
	$\overline{\text{XIPP}}$	50			
	$\overline{\text{LOAD}}$	20			
	$\overline{\text{RESET}}$	0			
Hold time, t_h	$\overline{\text{HOLD}}$	25			ns
	READY	30			
	D0 D15	35			
	CRUIN	25			
	$\overline{\text{INTREQ}}$	30			
	IC0 - IC3	30			
	$\overline{\text{XIPP}}$	5			
	$\overline{\text{LOAD}}$	25			
	$\overline{\text{RESET}}$	45			
Operating free-air temperature, T_A	SBP9989NJ, SBP9989NFD	−55		125	°C
	SBP9989CJ	0		70	

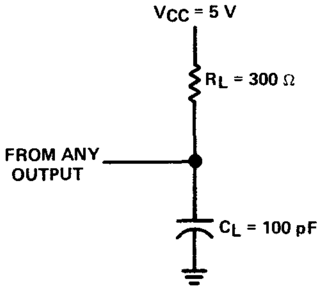
ELECTRICAL CHARACTERISTICS OVER RECOMMENDED OPERATING TEMPERATURE RANGE

PARAMETER			TEST CONDITIONS[†]		MIN	TYP	MAX	UNIT
V_{IH}	High-level input voltage		I_{CC} = NOM		2			V
V_{IL}	Low-level input voltage		I_{CC} = NOM				0 7	V
V_{IK}	Input clamp voltage		I_{CC} = NOM,	I_{IH} = −12 mA			−1 5	V
I_{OH}	High-level output current	I/O Pins	I_{CC} = NOM,	V_{IH} = 2 V,			1000	µA
		Other outputs	V_{IL} = 0.7,	V_{OH} = 5 5 V			250	
V_{OL}	Low-level output voltage		I_{CC} = NOM, V_{IL} = 0.7 V,	V_{IH} = 2 V, I_{OL} = 16 mA			0 4	V
I_{IH}	Input current	Clock	I_{CC} = NOM,	V_{IN} = 2 4 V			600	µA
		Other inputs					300	
V_{INJ}	Injector voltage		I_{CC} = NOM			1 25		V

[†] For test conditions shown as NOM, see the appropriate value under Recommended Operating Conditions

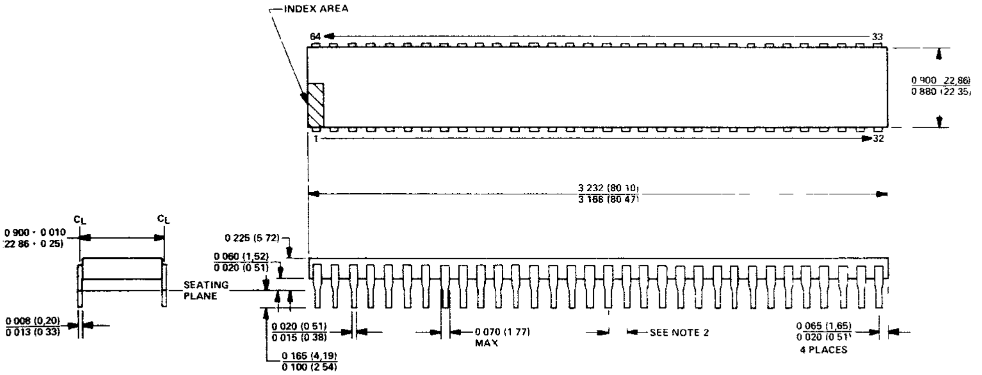
SWITCHING CHARACTERISTICS OVER RECOMMENDED OPERATING TEMPERATURE RANGE

PARAMETER	FROM	TO	TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{PD}	CLK↑	Address bus (A0 A14)	C_L = 100 pF			140	ns
	CLK↑	Memory map enable ($\overline{MPEN}$)				140	
	CLK↑	Data bus (D0 - D15)				140	
	CLK↓	Write enable ($\overline{WE}$)				180	
	CLK↑	Cycle end ($\overline{CYCEND}$)				175	
	CLK↑	Data bus in (DBIN)				155	
	CLK↑	Memory enable ($\overline{MEMEN}$)				140	
	CLK↑	↑CRU clock (CRUCLK)				185	
	CLK↓	↓CRU clock (CRUCLK)				185	
	CLK↑	CRU data out (CRUOUT)				175	
	CLK↑	Hold acknowledge (HOLDA)				150	
	CLK↑	Wait				140	
	CLK↑	Instruction Acquisition (IAQ)				140	
	CLK↑	Multiprocessor interlock (MPILCK)				145	
	CLK↑	Interrupt acknowledge (INTACK)				150	
t_{PD}	CLK↑	Address bus (A0 - A14)	When leaving a Hold State, C_L = 100 pF			200	ns
	CLK↑	Memory map enable ($\overline{MPEN}$)				200	
	CLK↑	Data bus (D0 - D15)				200	
	CLK↑	Memory enable ($\overline{MEMEN}$)				200	
	CLK↑	Data bus in (DBIN)				200	
	CLK↑	Instruction Acquisition (IAQ)				200	



SWITCHING-TIME LOAD CIRCUIT

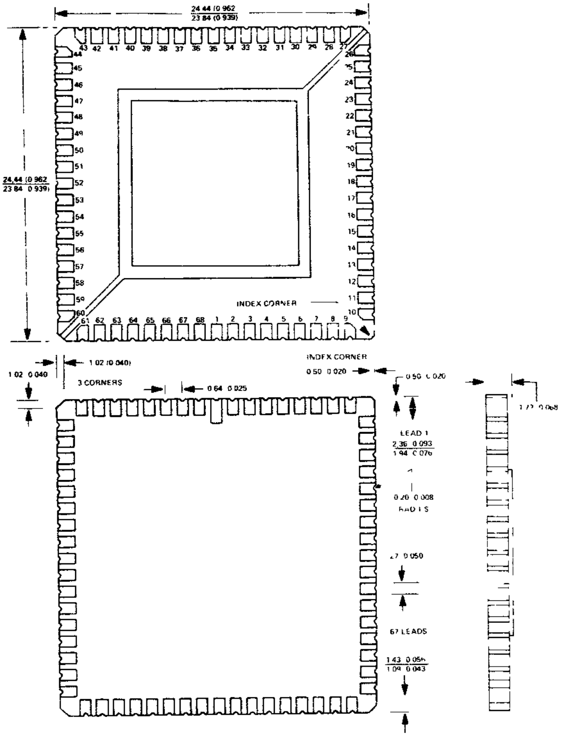
64-PIN CERAMIC DUAL-IN-LINE PACKAGE



NOTES 1 Dimensions are in inches and parenthetically in millimeters

2 Pin spacing is 100 (2 54 mm) between centerlines Each pin centerline shall be located within · 010 (25 mm) of its exact longitudinal position relative to pin 1 and 64

68-TERMINAL CHIP CARRIER



NOTES 3 Dimensions are in millimeters and parenthetically in inches

4 Contact spacing is 0 64 (0 025) between centers The center is indented with a half cylindrical center with a radius of 0 20 (0 008)

United Kingdom
Regional Technology Centre
Texas Instruments Ltd
Manton Lane
Bedford MK41 7PA
Tel (0234) 67466
Telex 82178
Technical Enquiry Service
Tel (0234) 223000

France
Texas Instruments
France
Boite Postale 5
06270 Villeneuve Loubet
Tel (93) 200101
Telex 470127

Germany
Texas Instruments
Deutschland GmbH
Haggertystrasse 1
8050 Freising
Tel 08161 800
Telex 526529

Italy
Texas Instruments
Semiconduttori Italia S.P.A.
Divisione Semiconduttori
Vialle delle Scienze 1
02015 Cittaducale (Rieti)
Tel (0746) 6941
Telex 611003

Austria
Texas Instruments GmbH
Industriestrasse B/16
A-2345 Brunn/Gebirge
Tel (02236) 846210
Telex 79304

Denmark
Texas Instruments AS
Marielundvej 46E
2730 Herlev
Tel (02) 917400
Telex 35123

Norway
Texas Instruments
Norway AS
KR August GT 13
Oslo 1
Tel (02) 206040
Telex 19628

Sweden
Texas Instruments
International Trade Corpn
Norra Hamnvagen 3
100 54 Stockholm
Tel (08) 235480
Telex 10377

Belgium
S.A. Texas Instruments
Belgium N.V.
Raketstraat,
100, Rue de la Fusee
1130 Brussels
Tel (02) 720 8000
Telex 61161

Finland
Texas Instruments
Finland OY
Teollisuuskatu 19D
PL 56
00510 Helsinki 51
Tel (90) 701 31 33
Telex 121457

Spain
Texas Instruments
Espana S.A.
C/Jose Lazaro Galdiano No 6
16 Madrid
Tel 1250 6105

Switzerland
Texas Instruments
Switzerland A.G.
Riedstrasse 6
CH-8953 Dietikon
Tel (01) 74 02 220
Telex 56260

TEXAS INSTRUMENTS
INCORPORATED

TEXAS INSTRUMENTS

SEPTEMBER 1982
SCG730

Manton Lane,
Bedford, England.

MPL 47
Printed in U.K.