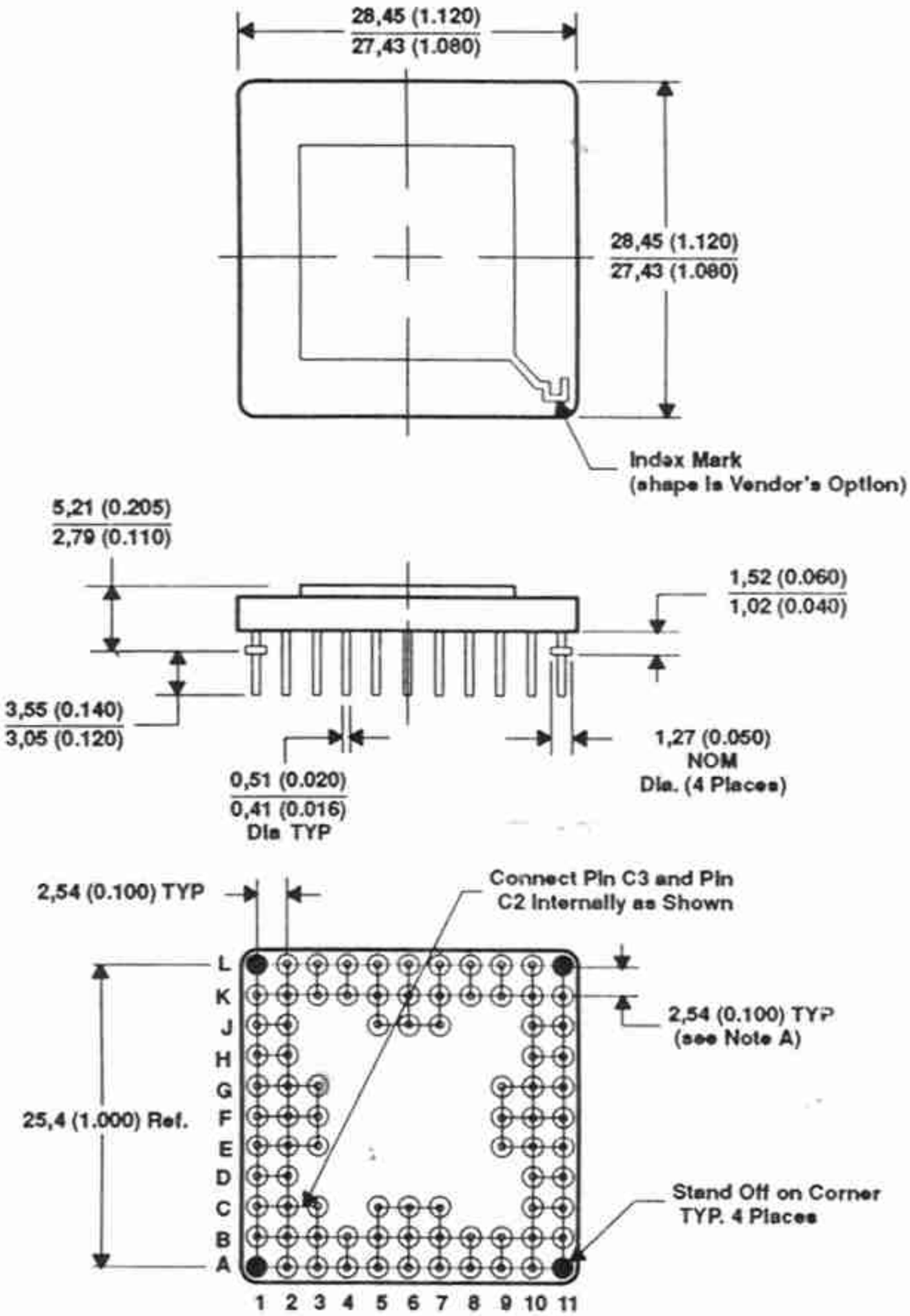
MECHANICAL DATA

84-PGA CERAMIC PACKAGE — **Designation Per JEDEC-Std-30: S-CPGA-P84**



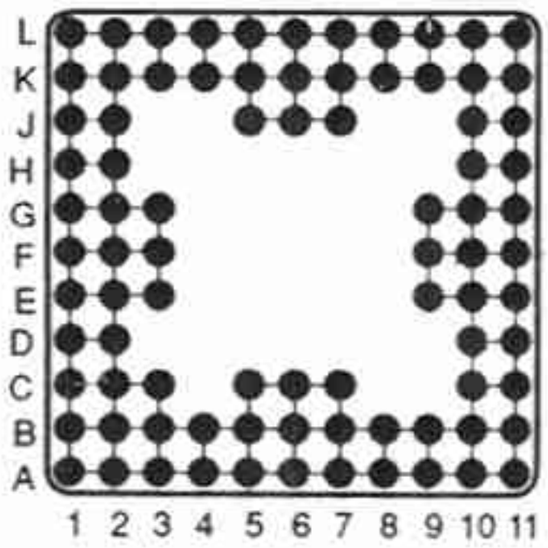
ALL LINEAR DIMENTIONS ARE IN MILLIMETERS AND PARENTHETICALLY IN INCHES

NOTE A. Pins are located within 0,13 (0.005) radius of true position relative to each other at maximum material condition and within 0,38 (0.051) radius relative to the center of the ceramic.

SGUS015 – D3997 – JUNE 1992

- **Certified In Accordance With Class B MIL-STD-883C Over Full – 55 to 125°C Military Temperature Range**
- **Uses the Latest TI 1.2-Micron CMOS EPIC 1Z Process**
- **Memory-to-Memory Architecture**
- **73 Basic Instructions Include All 69 SBP9900A Instructions Plus Signed Multiply, Signed Divide, Load WP and Load ST**
- **Allows User-Defined Extensions to the Basic Instruction Set**
- **Direct Access to 128K Bytes of Memory**
- **Full Static Design Allows Operation from Single-Step Up to 16 MHz**
- **Designed for Use With Single 50% Duty Cycle Clock**
- **Fully TTL-Compatible I/O Buffers**

84-PIN CERAMIC PGA PACKAGE
(TOP VIEW)



description

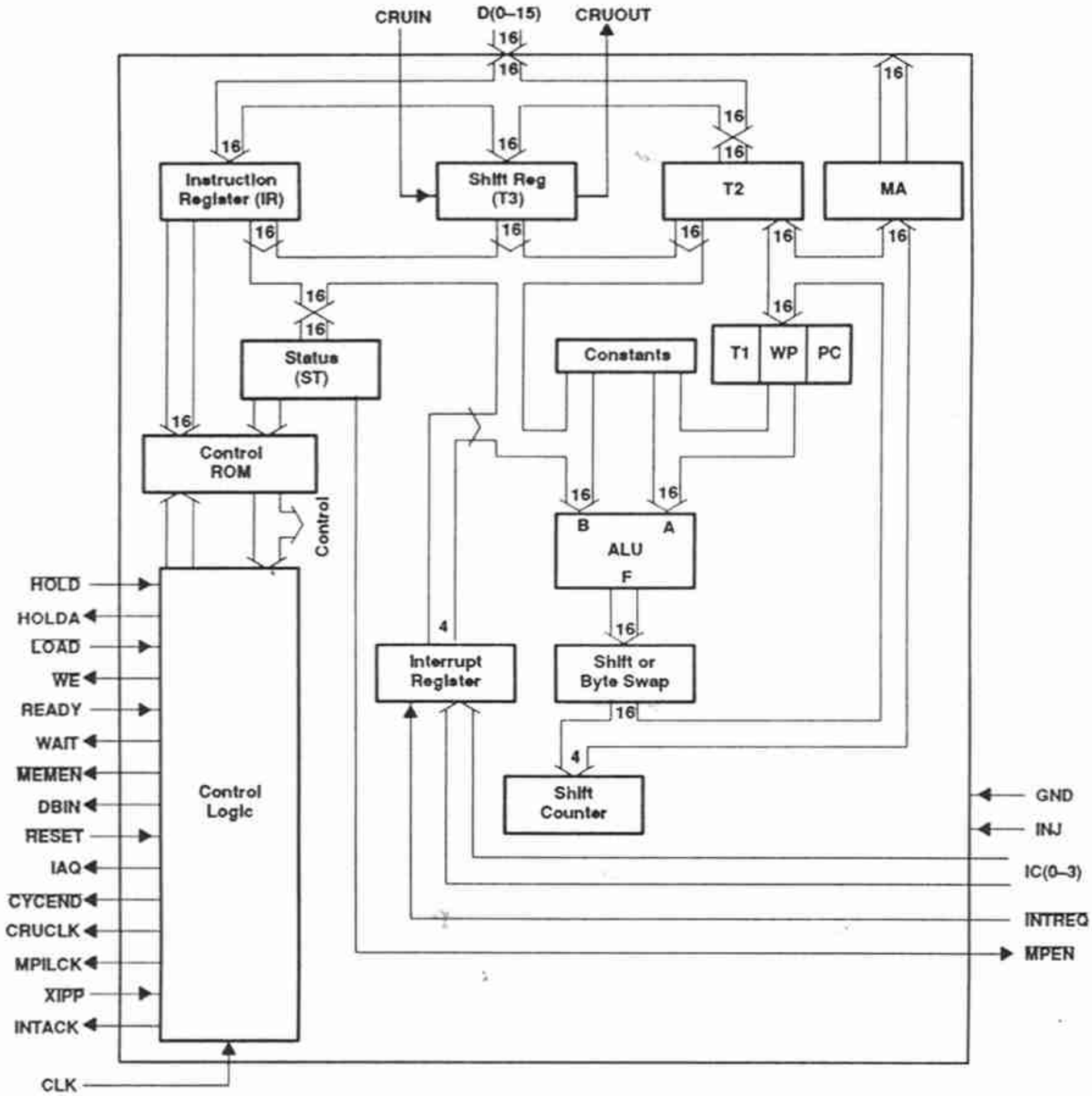
The SMJ68689 is an option for the SBP9989J 16-bit I^2L microprocessor using the advantages of CMOS technology with low power consumption and high speed.

The single most important feature of the '68689 is its memory-to-memory architecture. 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 register architecture) in only 5 memory cycles.

DRAFT # 7 DATE 7-01-92 JOB # 51000-18
WRITER _____ SUPV. OK _____
~~FRED MANN~~ QC R. Wong DATE 7-01-92
PRODUCT DEPT OK AS IS _____ W/CHANGES _____
BY _____ DATE _____

TEXAS INSTRUMENTS
POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

functional block diagram



GB PACKAGE – PIN FUNCTION ASSIGNMENTS

PIN	FUNCTION	PIN	FUNCTION	PIN	FUNCTION
B2	GND	L5	D6	D10	GND
C2	MPEN	K6	GND	C11	RESET
B1	GND	J6	D7	B11	IAQ
C1	DBIN	J7	D8	C10	GND
D2	CRUOUT	L7	V_{CC}	A11	CLK
D1	GND	K7	D9	B10	A14
E3	CRUIN	L6	D10	B9	A13
E2	V_{CC}	L8	GND	A10	V_{CC}
E1	INTREQ	K8	D11	A9	A12
F2	IC3	L9	D12	B8	A11
F3	GND	L10	D13	A8	A10
G3	IC2	K9	GND	B6	V_{CC}
G1	V_{CC}	L11	D14	B7	A9
G2	IC1	K10	D15	A7	A8
F1	IC0	J10	GND	C7	GND
H1	GND	K11	XIPP	C6	A7
H2	GND	J11	CYCEND	A6	A6
J1	INTACK	H10	CRUCLK	A5	V_{CC}
K1	MPILK	H11	GND	B5	A5
J2	GND	F10	WE	C5	A4
L1	GND	G10	V_{CC}	A4	GND
K2	D0	G11	READY	B4	A3
K3	D1	G9	MEMEN	A3	A2
L2	V_{CC}	F9	GND	A2	A1
L3	D2	F11	HOLD	B3	GND
K4	D3	E11	WAIT	A1	A0
L4	D4	E10	V_{CC}		
J5	V_{CC}	E9	LOAD		
K5	D5	D11	HOLDA		

Terminal Functions

PIN NAME	I/O	DESCRIPTION
A0–A14	O	Address bus. This open-collector bus provides the memory address to the memory system when MEMEN is active and CRU I/O bit addresses to the I/O system when MEMEN is inactive and DBIN is active.
CLK	I	Single-phase clock input
CRUCLK	O	CRU clock. When active (high), CRUCLK indicates either the presence of output data on CRUOUT or the presence of an encoded external instruction on A0–A2.
CRUIN	I	CRU data in. CRUIN receives input data from the external interface logic. When the SM/SMJ68689 executes a STCR or TB instruction, it samples CRUIN for the level of the CRU bit specified by the address bus (A3–A14).
CRUOUT	O	CRU data out. CRUOUT outputs serial data when the SM/SMJ68689 executes a LDCR, SBZ, or SB instruction. The data on CRUOUT should be sampled by the external interface logic when CRUCLK goes active.
CYCEND	O	End of cycle. When active (low), CYCEND indicates that the SM/SMJ68689 will initiate a new microinstruction cycle on the next low-to-high transition of the clock.
D0–D15	I/O	Data bus (open-collector). This bus transfers memory data to (when writing) and from (when reading) the external memory system when MEMEN is active.
DBIN	O	Data bus in. When asserted (high) by the SM/SMJ68689 during MEMEN, DBIN indicates that the SM/SMJ68689 has disabled its output buffers to allow the memory system to place memory read data on the bus. The SM/SMJ68689 also activates DBIN during all CRU operations and during the execution of the five external instructions. In all other cases except when HOLDA is active, the SM/SMJ68689 maintains DBIN at a low logic level.
GND		Ground
HOLD	I	Hold. When active (low), HOLD indicates to the SM/SMJ68689 that an external controller (e.g., DMA device) desires to use the memory bus for direct data transfers. In response, the SM/SMJ68689 enters the hold state after completion of its present cycle (memory or non-memory). The SM/SMJ68689 then asserts HOLDA and allows its address bus, MPEN, WE, DBIN, IAQ and CYCEND to be pulled into the high logic state. When HOLDA is released, the SM/SMJ68689 resumes bus control and continues operation by resuming execution of the suspended instruction.
HOLDA	O	Hold acknowledge. When active (high), HOLDA indicates that the SM/SMJ68689 is in a hold state and that its address bus, MPEN, data bus, MEMEN, WE, DBIN, IAQ, and CYCEND are pulled to the high state. The SM/SMJ68689 enters a hold state in response to the activation of HOLD or XIPP (during the execution of an ILLOP or XOP instruction).
IAQ	O	Instruction acquisition. IAQ is asserted (high) by the SM/SMJ68689 during any SM/SMJ68689-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 operation code data.
IC0–IC3	I	Interrupt codes. IC0 (MSB)–IC3 (LSB), indicating an interrupt identity code, are sampled by the SM/SMJ68689 when 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.
INJ		Injector-supply current
INTACK	O	Interrupt acknowledge. When active (high) during non-hold states, INTACK indicates the SM/SMJ68689 has initiated a trap sequence caused by the receipt of a valid interrupt, LOAD or RESET. INTACK is activated in the trap sequence while the SM/SMJ68689 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 SM/SMJ68689 is in a hold state (caused by activation of XIPP or HOLD) INTACK indicates the SM/SMJ68689 has received a valid interrupt (level is less than or equal to the value of the interrupt mask), a LOAD or RESET. INTACK remains valid high until the SM/SMJ68689 leaves a hold state (HOLD or XIPP released) or until the signal requesting the interrupt is released.
INTREQ	I	Interrupt request. When active (low), INTREQ indicates that an external interrupt is requesting service. If INTREQ is active, the SM/SMJ68689 loads the data on 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 SM/SMJ68689 initiates the interrupt sequence. If the comparison fails, the SM/SMJ68689 ignores the interrupt request. In that case, INTREQ should be held active. The SM/SMJ68689 continues to sample IC0–IC3 until the program enables a sufficiently low interrupt level to accept the requesting interrupt.

Terminal Functions (continued)

PIN NAME	I/O	DESCRIPTION
$\overline{LOAD}$	I	Load. When active (low), $\overline{LOAD}$ causes the SM/SMJ68689 to set $\overline{MPEN}$ high, issue INTACK, store the old PC, WP, and ST, set status register bits 7–15 low, and execute a nonmaskable interrupt with unmapped memory addresses FFFC and FFFE containing the associated trap vector (WP and PC). The load sequence is initiated after the instruction being executed is completed. $\overline{LOAD}$ also terminates an idle state. If $\overline{LOAD}$ is active at the end of a reset function, the $\overline{LOAD}$ trap will occur after the reset function is completed. If $\overline{LOAD}$ is activated during a hold state (caused by $\overline{XIPP}$ or $\overline{HOLD}$), the SM/SMJ68689 activates INTACK to indicate a pending $\overline{LOAD}$ needs to be serviced. During hold states, $\overline{LOAD}$ should remain active until the SM/SMJ68689 leaves the hold state and the above conditions are met. $\overline{LOAD}$ 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.
$\overline{MEMEN}$	O	Memory enable. When active (low), $\overline{MEMEN}$ indicates that the address bus contains a valid memory address.
$\overline{MPEN}$	O	Memory map enable. $\overline{MPEN}$ represents the inverted value of status register bit 8 (ST8). $\overline{MPEN}$ can be changed by any instruction (i.e., LST, etc.) affecting ST8 and is set to 1 during SM/SMJ68689 trap addressing (interrupts, $\overline{LOAD}$, $\overline{RESET}$, XOP, and ILLOP). $\overline{MPEN}$ may be used to allow memory expansion to 128K bytes.
MPILCK	O	Multiprocessor interlock. When active (high), MPILCK indicates the SM/SMJ68689 is performing the operations associated with operand transfer and manipulation for the ABS instruction. MPILCK is asserted by the SM/SMJ68689 during any ABS instruction upon initiation of the operand read operation and remains active until the completion of the instruction (e.g., MPILCK remains active for the duration of the SM/SMJ68689 read-modify-write operation cycle for the ABS instruction). Consequently, MPILCK may be used in the implementation of a nonseperable test-and-set capability. $\overline{HOLD}$ is sampled during MPILCK activation, so MPILCK can be used to control assertion of $\overline{HOLD}$.
READY	I	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 SM/SMJ68689 enters a wait state and suspends internal operations until the memory system or CRU device activates ready.
$\overline{RESET}$	I	Reset. When active (low), $\overline{RESET}$ causes the SM/SMJ68689 to reset itself and inhibit $\overline{WE}$, CRUCLK and $\overline{MEMEN}$. When $\overline{RESET}$ is released, the SM/SMJ68689 goes through a level-zero interrupt sequence by causing $\overline{MPEN}$ to go high, asserting INTACK, storing the old PC, WP and ST, setting all status register bits to zero, acquiring the WP and PC trap vector from memory locations 0000 and 0002, and then fetching the first instruction of the reset program environment if $\overline{LOAD}$ is not active. The SM/SMJ68689 continuously samples $\overline{RESET}$ on low-to-high clock transitions. $\overline{RESET}$ must be active for one low-to-high transition of the clock and satisfy the setup and hold time requirements of this signal.
WAIT	O	Wait. When active (high), WAIT indicates the SM/SMJ68689 has entered a wait state in response to a not READY condition from a memory system or a CRU device.
$\overline{WE}$	O	Write enable. When active (low), $\overline{WE}$ indicates that the SM/SMJ68689 data bus is outputting data to be written into memory.
$\overline{XIPP}$	I	Extended instruction processor present. When activated (low) by an external device (an extended instruction processor, or XIP) upon detection of the acquisition of an SM/SMJ68689 undefined opcode, $\overline{XIPP}$ indicates the XIP will execute the undefined instruction. Recognition of $\overline{XIPP}$ causes the SM/SMJ68689 to allow its memory bus signals to be pulled high, assert 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 then proceeds to execute the undefined instruction. During the instruction execution, the XIP may utilize the WP, PC and ST previously stored in memory by the SM/SMJ68689. Upon completion of its instruction execution, the XIP releases $\overline{XIPP}$ and allows the SM/SMJ68689 to resume bus control and restart instruction execution. The SM/SMJ68689 resumes operation by reloading (from memory) its WP, PC and ST. $\overline{XIPP}$ 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.

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 SM/SMJ68689 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 SM/SMJ68689 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 $\overline{LOAD}$ signal. The remaining memory is then available for programs, data, and workspace registers.

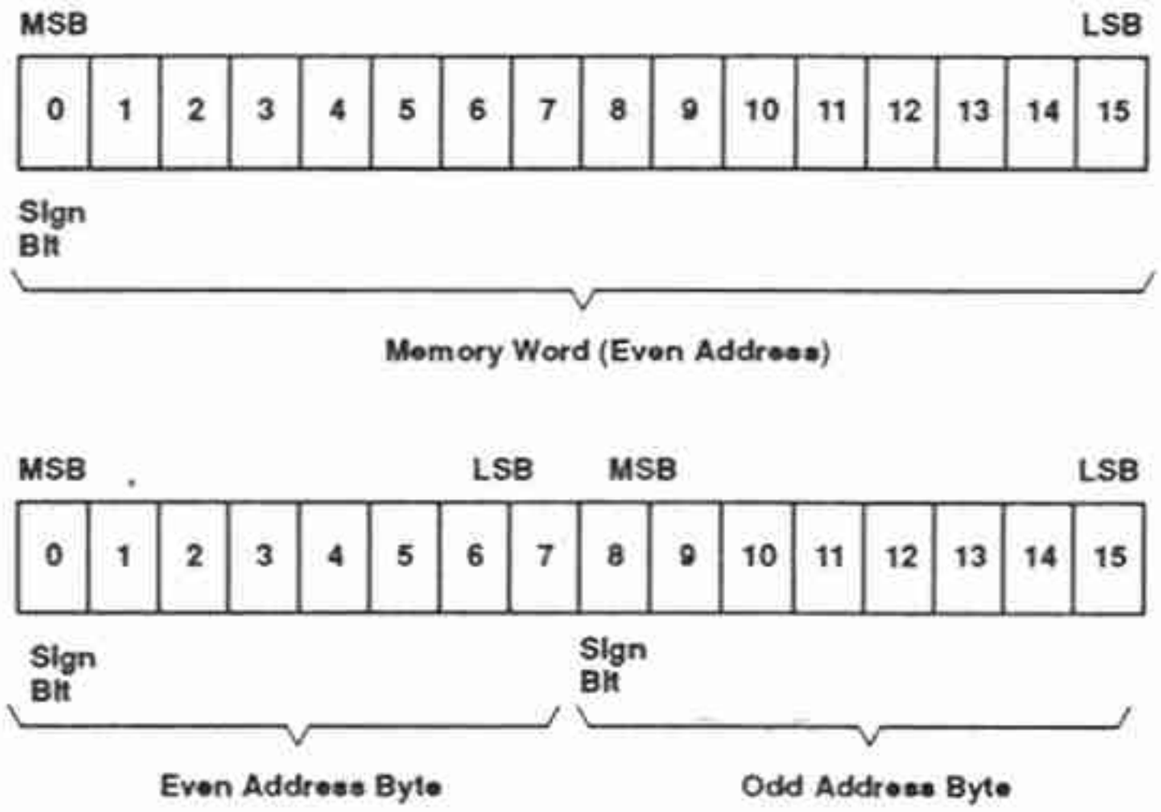


Figure 1. Memory Word and Byte Formats

functional description

functional block diagram

The functional block diagram is shown on page 2. 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 SM/SMJ68689. 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 is handled. 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.

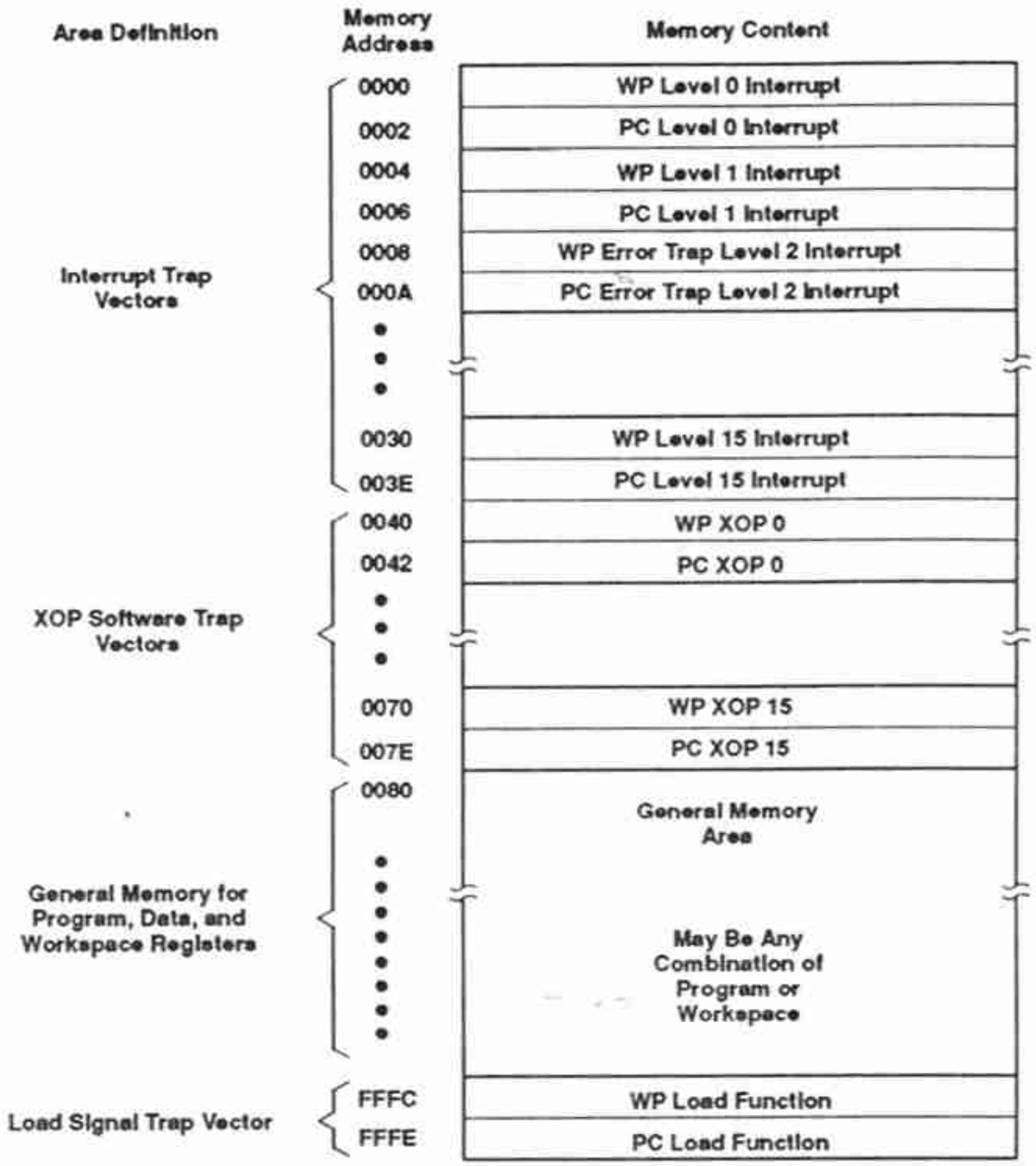


Figure 2. Map of the Memory Arrangement

Internal registers

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

1. Program counter (PC)
2. Status register (ST)
3. Workspace pointer (WP)

Other internal registers that 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 SM/SMJ68689 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 the PC. All context instructions affect the contents of the PC.

status register (ST)

The status register is a 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 SM/SMJ68689 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 the document details the effect of each instruction on the status register.

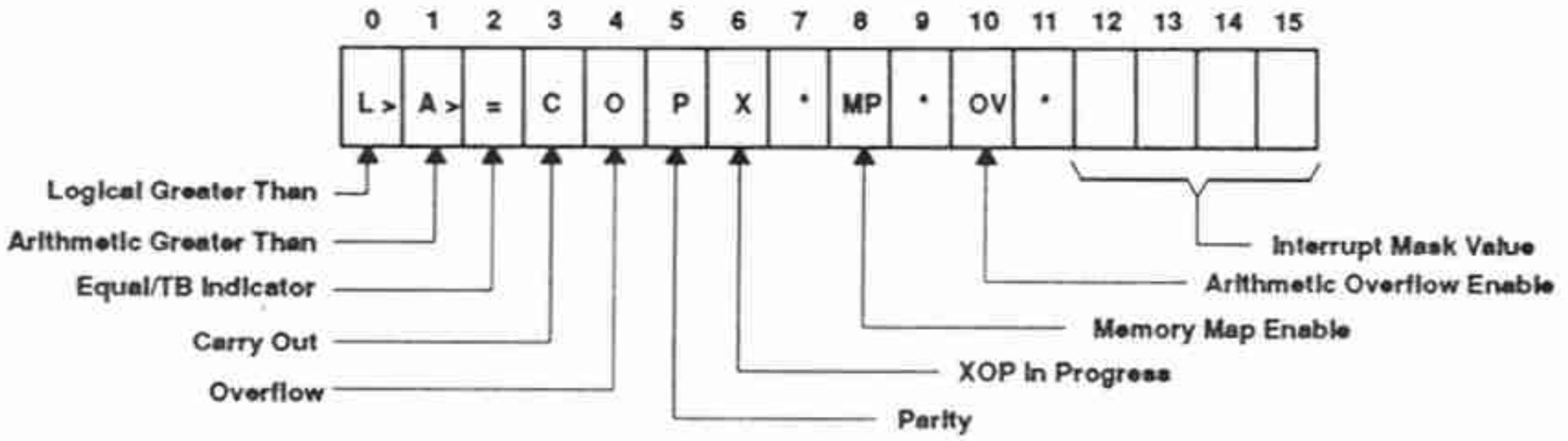


Figure 3. Status Register Bit Assignments

workspace

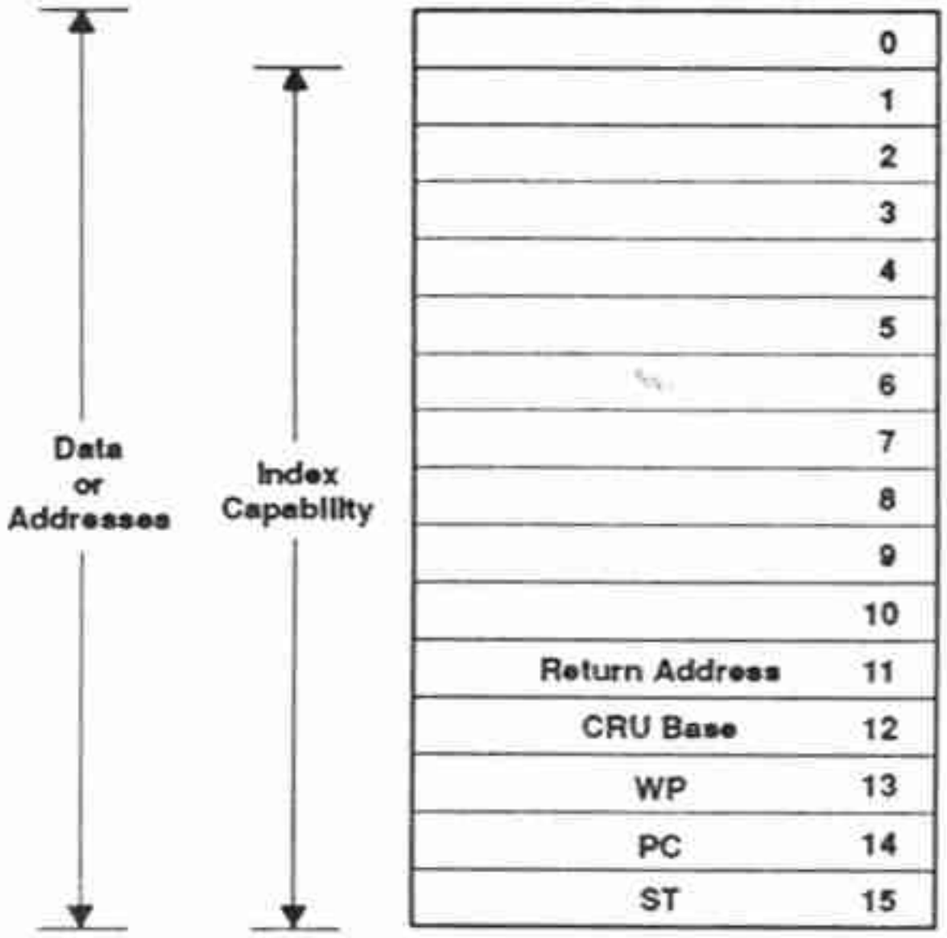
The SM/SMJ68689 uses blocks of memory words, called workspaces, for instruction operand manipulation (instead of internal hardware registers). A workspace occupies 16 contiguous memory words in any part of memory. The individual workspace registers may contain data or addresses and function as operand 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, providing 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 SM/SMJ68689 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.

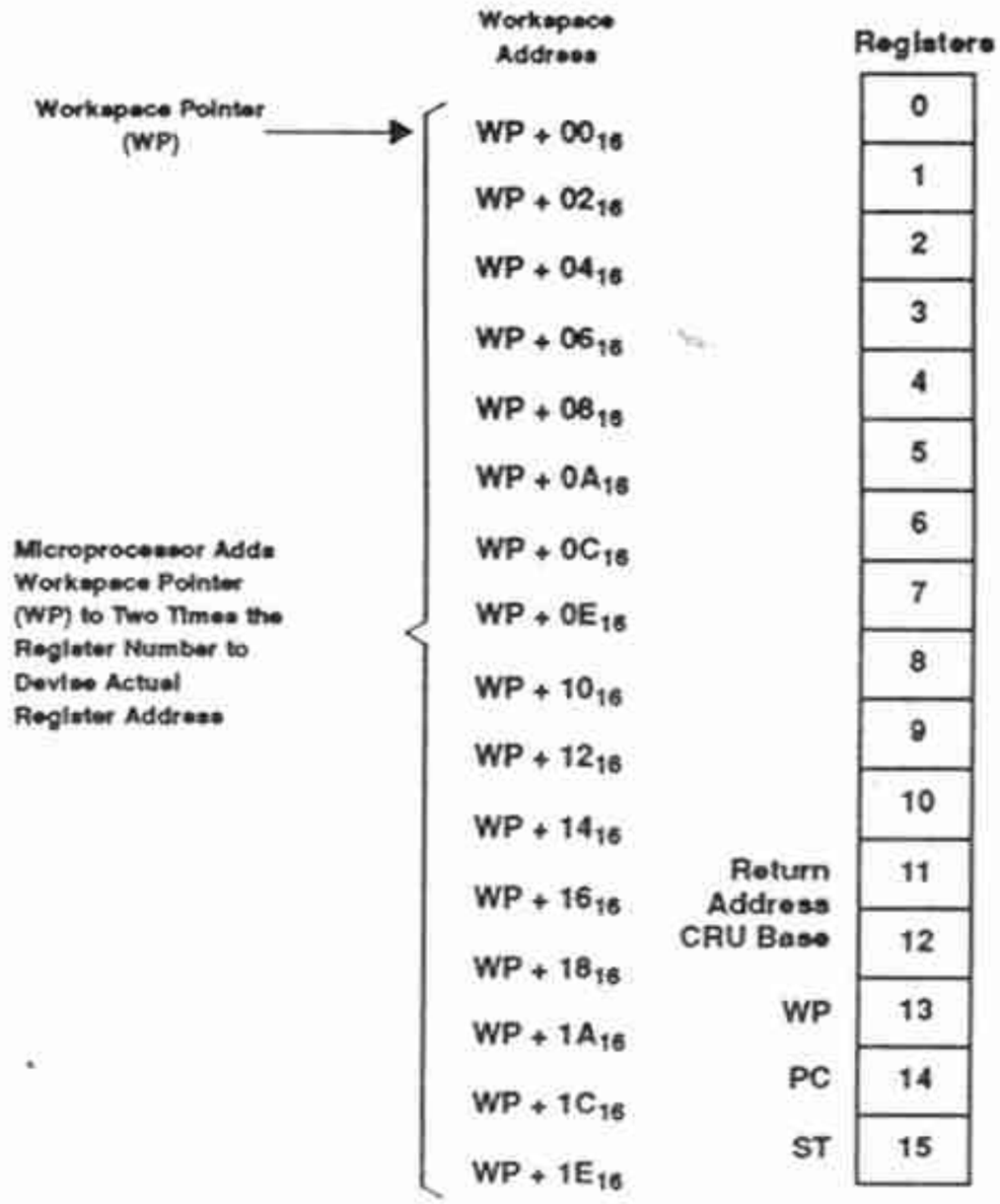
context switching

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 multiregister 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 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 savings 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). 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. Workspace Map



NOTE: All memory word addresses are even.

Figure 5. Workspace Pointer and Registers

Interface

memory Interface

The memory bus provides the interface between the SM/SMJ68689 and the memory system. It consists of a data bus, an address bus, and control signals. Figure 6 illustrates the signals contained in the memory interface. Figures 12 and 13 illustrate the timing relationships between these signals. During each memory-read or memory-write cycle, $\overline{\text{MEMEN}}$ becomes active (logic low level) along with valid memory address data appearing on the address bus (A0 through A14). The memory map enable ($\overline{\text{MPEN}}$) output is provided by the SM/SMJ68689 to allow memory expansion to 64K words. $\overline{\text{MPEN}}$ represents the inverse of the value of the status register bit 8 (ST8) to external devices. $\overline{\text{MPEN}}$ can be changed by any instruction affecting ST8 (i.e., LST, ect.) and is set to one during the SM/SMJ68689 trap addressing: namely, interrupts, LOAD, RESET, XOPs, and illegal opcodes (ILLOPs). $\overline{\text{MPEN}}$ remains set to one until ST8 is set to one, at which time $\overline{\text{MPEN}}$ is set to zero.

In the case of the memory-read cycle, DBIN becomes active (pulled to a logic high level) at the same time memory-address data becomes valid; the memory write strobe $\overline{\text{WE}}$ remains inactive (pulled to a logic high level). If the memory-read cycle is initiated for acquisition of an instruction, IAQ becomes active (pulled to a logic high level) at the same time $\overline{\text{MEMEN}}$ becomes active. At the end of a memory-read cycle, $\overline{\text{MEMEN}}$ and DBIN together become inactive. At that time, through the address may change, the data bus remains in the input mode until terminated by the next low-to-high transition of the clock. On consecutive memory accesses, $\overline{\text{MEMEN}}$ remains low.

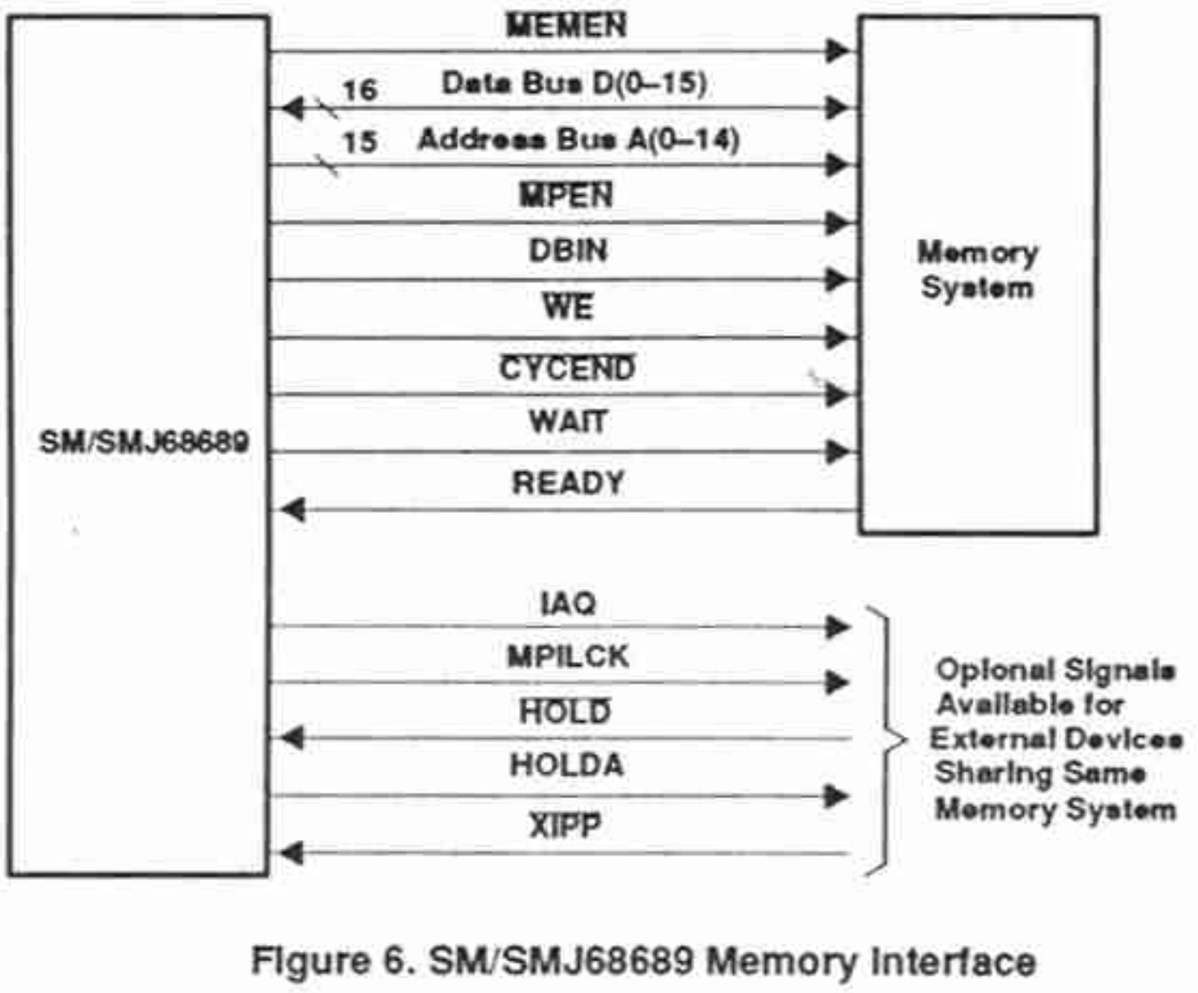


Figure 6. SM/SMJ68689 Memory Interface

In the case of a memory-write cycle, $\overline{WE}$ becomes active (logic level low) with the first high-to-low transition of the clock after $\overline{MEMEN}$ becomes active; DBIN remains inactive (low). At the end of a memory-write cycle, $\overline{WE}$ and $\overline{MEMEN}$ become inactive ($\overline{MEMEN}$ on the low-to-high transition of clock; $\overline{WE}$ 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 so that the speed of the memory system may be coordinated with the speed of the SM/SMJ68689. If READY is inactive (logic low level) during the first low-to-high transition of clock after $\overline{MEMEN}$ becomes active, the SM/SMJ68689 enters 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 ($\overline{CYCEND}$) is activated (logic low level) for one clock period during each microinstruction cycle (i.e., memory operation, non-memory internal operation, etc). $\overline{CYCEND}$ is activated on the low-to-high transition of the clock that initiates the last clock period of a microinstruction cycle and is deactivated on the next low-to-high clock transition.

HOLD, DMA Interface

The SM/SMJ68689 hold facilities allow 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 ($\overline{HOLD}$) to the SM/SMJ68689. 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 acknowledgment, 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 $\overline{HOLD}$.

When $\overline{HOLD}$ becomes active (logic low level), SM/SMJ68689 enters a hold state at the beginning of the next available cycle as shown in Figure 14. Upon entering a hold state, HOLDA becomes active (pulled to a logic high level), and the following signals are pulled to a high logic level by individual pullup resistors tied to each respective open-collector output: DBIN, $\overline{MEMEN}$, IAQ, $\overline{MPEN}$, $\overline{WE}$, $\overline{CYCEND}$, A0 through A14, and D0 through D15. When $\overline{HOLD}$ becomes inactive, the SM/SMJ68689 exits the hold state and regains memory bus control.

extended Instruction processor Interface

The extended instruction processor (XIP) interface provides for easy extension of the SM/SMJ68689 arithmetic/logic processing function by facilitating the addition of external hardware instruction processors while also permitting the usage of software interpretive implementation of extended instructions. The XIP interface

extended instruction processor interface (continued)

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 SM/SMJ68689 employing XIPs and those without XIPs can be identical.

The XIP interface utilizes the SM/SMJ68689 memory bus with direct memory access (DMA) capability and the extended instruction processor present ($\overline{\text{XIPP}}$) signal as shown in Figure 7. The XIP interface requires the XIP to gain control of the SM/SMJ68689 memory bus during execution of any extended instructions encountered in the device program stream. The extended instructions are assigned illegal (undefined) operation codes (ILLOPs) by the user.

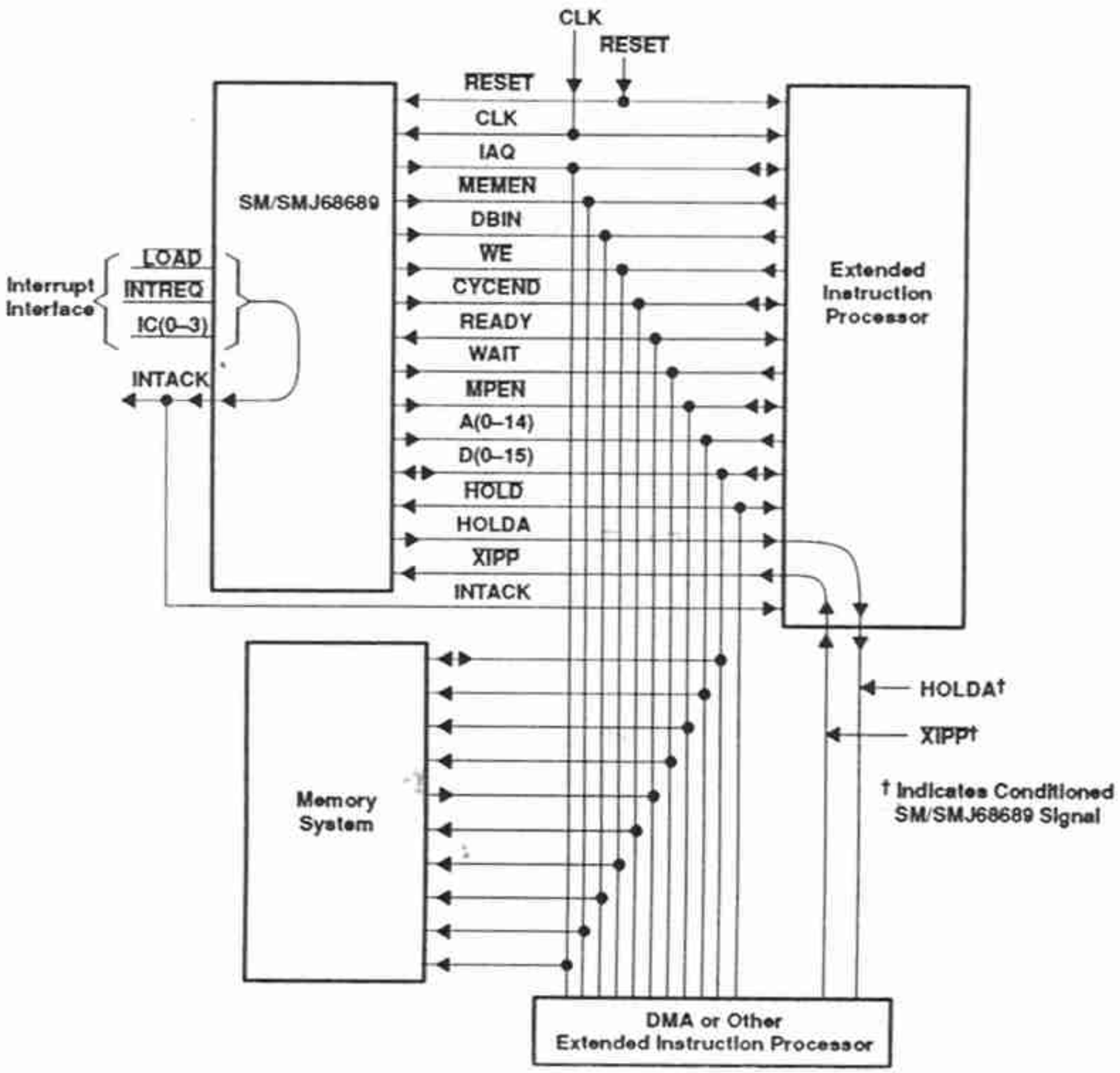


Figure 7. Extended-Instruction Interface Processor

extended instruction processor interface (continued)

The sequence that characterizes a typical extended-instruction execution is shown in Figures 15 and 16. As illustrated, the SM/SMJ68689 fetches the instruction that contains the opcode assigned to an extended instruction. The XIP detects the occurrence of the instruction fetch operation (IAQ active) and latches the instruction opcode data present on the memory data bus. The XIP decodes the latch opcode as one of its instructions and asserts $\overline{\text{XIPF}}$ to the SM/SMJ68689 (non-extended instruction opcodes are ignored by XIP). Recognition of the illegal opcode causes the SM/SMJ68689 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 SM/SMJ68689 level 2 interrupt vector location. If $\overline{\text{XIPF}}$ is also active (logic low level), the SM/SMJ68689 allows its memory bus signals to be pulled to a logic high level, 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 instructions. During its 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 $\overline{\text{XIPF}}$. Detecting the removal of $\overline{\text{XIPF}}$ causes the SM/SMJ68689 to remove HOLDA, activate its memory bus drivers (i.e., resume bus control), retrieve the previous 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.

The XIP can be implemented with chaining capability; i.e., the ability to execute a sequence of extended instructions without returning control to the SM/SMJ68689. If an interrupt or $\overline{\text{LOAD}}$ occurs and the interrupt mask conditions are satisfied, the SM/SMJ68689 activates INTACK (interrupt acknowledge) to indicate that an interrupt needs to be serviced. Servicing of the interrupt occurs upon release of $\overline{\text{XIPF}}$ by the XIP and completion of the above context restore/bus control resumption sequence.

As shown in Figure 7, 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 $\overline{\text{XIPF}}$) to satisfy additional DMA or XIP requirements.

For configurations not containing external XIP hardware, the $\overline{\text{XIPF}}$ signal is not activated. Therefore, the SM/SMJ68689 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 functions.

multiprocessor interlock

The multiprocessor interlock (MPILCK) signal permits the implementation of an indivisible test-and-set-semaphore mechanism for use in multiprocessor applications. The SM/SMJ68689 activates the MPILCK signal (logic high level) during the execution of the absolute value (ABS) instruction as shown in Figure 17. 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 $\overline{\text{HOLD}}$.

Interrupts

The SM/SMJ68689 employs 16 interrupt levels, with level 0 being the highest priority and level 15 being lowest priority. Level 0 is reserved for the $\overline{\text{RESET}}$ function. Level 2 is reserved at the user's option for the arithmetic overflow interrupt and/or an illegal opcode trap. Interrupt levels 1 and 3 through 15 may be used for external device interrupts.

External device interrupts are input via the interrupt request ($\overline{\text{INTREQ}}$) signal line and the four interrupt code lines (IC0–IC3). Figure 19 shows the timing for the sampling of these inputs and their effect on SM/SMJ68689 operation. Activation of the $\overline{\text{INTREQ}}$ 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 SM/SMJ68689 recognizes

Interrupts (continued)

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 while the SM/SMJ68689 releases $\overline{\text{MPEN}}$ and asserts the INTACK (Interrupt acknowledge) signal line. The previous context (WP, PC and ST) is then 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 the interrupt mask is set to a value that is one less than the level of the interrupt being serviced (except for a level 0 interrupt, which loads into the mask). This allows only interrupts of higher priority to interrupt a service routine. The SM/SMJ68689 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 should reset the interrupt requests before the service routines complete.

Table 1. Interrupt Level Data

INTERRUPT LEVEL	VECTOR LOCATION (MEMORY ADDRESS IN HEX)	DEVICE ASSIGNMENT	ENABLING MASK VALUES (ST12–ST15)	INTERRUPT CODES (IC1–IC3)
Highest priority				
0	00	Reset	0 thru F (see Note 1)	0000
1	04	External device	1 thru F	0001
		Arithmetic overflow	2 thru F (see Note 2)	See Note 2
		Illegal opcode	0 thru F (see Note 3)	See Note 3
		$\overline{\text{XIPP}}$ active during IAQ		See Note 4
		External device	2 thru F	0010
3	0C	External device	3 thru F	0011
4	10	External device	4 thru F	0100
5	14	External device	5 thru F	0101
6	18	External device	6 thru F	0110
7	1C	External device	7 thru F	0111
8	20	External device	8 thru F	1000
9	24	External device	9 thru F	1001
10	28	External device	A thru F	1010
11	2C	External device	B thru F	1011
12	30	External device	C thru F	1100
13	34	External device	D thru F	1101
14	38	External device	E and F	1110
15	3C		F only	1111
Lowest priority				

NOTES: 1. Level 0 cannot be disabled.
2. Arithmetic overflow interrupt is generated internal to the device and enabled/disabled by bit 10 of the status register.
3. Illegal opcode trap is generated internal to the device, and it cannot be disabled by the interrupt mask.
4. $\overline{\text{XIPP}}$ is inactive at logic high level.

If a higher-priority interrupt occurs, a second context switch occurs to service the higher-priority interrupt. When that routine is complete, a return with-workspace-pointer 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.

Interrupts (continued)

During an SM/SMJ68689 hold state resulting from the activation of $\overline{HOLD}$ or $\overline{XIPP}$, the SM/SMJ68689 continues to sample the interrupt code lines. Upon activation of $\overline{INTREQ}$, if the code is less than or equal to the device interrupt mask level, the SM/SMJ68689 activates the INTACK signal to indicate a pending interrupt needs servicing. The INTACK signal remains active until $\overline{HOLD}$ or $\overline{XIPP}$ is released.

undefined opcode trap

The acquisition and execution of illegal (undefined) SM/SMJ68689 opcodes causes a trap operation to occur using the information stored in the level-2 vectors. The opcodes that cause the trap are 0000–007F, 00A0–017F, 0320–033F, 0780–07FF, and 0C00–0FFF. If the $\overline{XIPP}$ signal is activated during the undefined instruction, the SM/SMJ68689 suspends operation, and the instruction is executed by the XIP. However, if the XIP signal remains inactive, the undefined opcode causes a trap, and a context switch occurs. The occurrence of the trap is nonmaskable (i.e., not controlled by the interrupt mask value), and the trap overrides any level interrupt. Interrupts are inhibited until the first instruction of the trap subroutine is executed. The occurrence of the undefined opcode trap does not change the interrupt mask.

Interrupt level 0 (reset)

Interrupt level 0 is reserved for the $\overline{RESET}$ input to the SM/SMJ68689. When asserted (logic low level), $\overline{RESET}$ causes the SM/SMJ68689 to reset itself and to inhibit $\overline{WE}$ and CRUCLK.

When $\overline{RESET}$ 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 $\overline{LOAD}$ is active, the LOAD trap occurs after the RESET function is completed. The SN/SMJ68689 continuously samples $\overline{RESET}$ on low-to-high clock transitions as shown in Figure 20. To be recognized, $\overline{RESET}$ must be active for one low-to-high transition of the clock and must satisfy the setup and hold time requirements.

Interrupt level 2

Interrupt level 2 has two additional capabilities associated with its use. 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 that 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 the overflow interrupt; ST10 = 0 disables overflow interrupt). The overflow interrupt can be inhibited by the interrupt mask (ST12–ST15) or overridden by a pending level-0 or level-1 interrupt. If servicing the overflow condition is overridden by a higher priority (level 0 or 1) interrupt, the overflow condition is 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 the RTWP instruction causes the overflow condition to be reloaded into status register bit (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.

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 SM/SMJ68689 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 8, the SM/SMJ68689 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 21 and 22.

communication register unit (CRU) interface (continued)

To transfer a data bit to a CRU device, the SM/SMJ68689 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 SM/SMJ68689 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 SM/SMJ68689 activates the DBIN output (logic high level) and places the 16-bit data bus in the input mode. Activation of DBIN allows users to detect the occurrence of a CRU cycle and facilitates cycle stealing by a DMA device, since all uses of the data and address buses are indicated by the assertion of DBIN or $\overline{\text{MEMEN}}$.

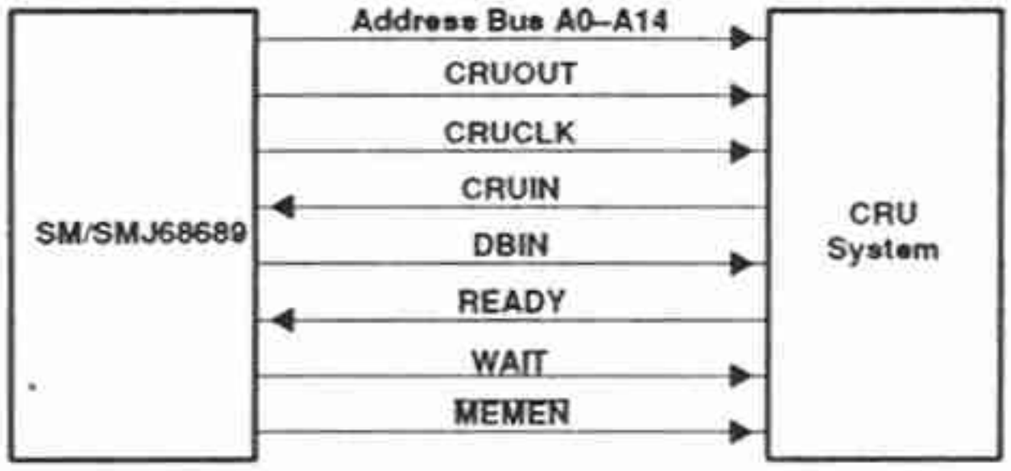


Figure 8. CRU Interface Support

During either a CRU input or output operation, READY may be used to extend the duration of the associated CRU operation so that the speed of the CRU device may be coordinated with the speed of the SM/SMJ68689. If READY is asserted (logic high level) during the first low-to-high transition of clock after the initiation of the CRU operation, the SM/SMJ68689 enters a wait state (i.e., WAIT is asserted), suspending further progress of the CRU cycle. The first low-to-high transition of clock after READY is asserted terminates the wait state and allows normal completion of the CRU operation.

single-bit CRU operation

The SM/SMJ68689 performs three single-bit CRU functions: test bit (TB), set bit to one (SBO), and set bit to zero (SBZ). The device also activates DBIN, generates a CRUCLK pulse, indicating an output operation to the CRU device, and places 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). The test bit (TB) instruction transfers the addressed CRU bit from the CRUIN input line to bit 2 of the status register (ST12).

The SM/SMJ68689 develops a CRU-bit address for the single-bit operations from the CRU base address contained in workspace register 12 and the signed displacement count contained in bits 8 through 15 of the instruction. The displacement allows twos 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 into the address bus. Figure 9 illustrates the development of a single-bit CRU address.

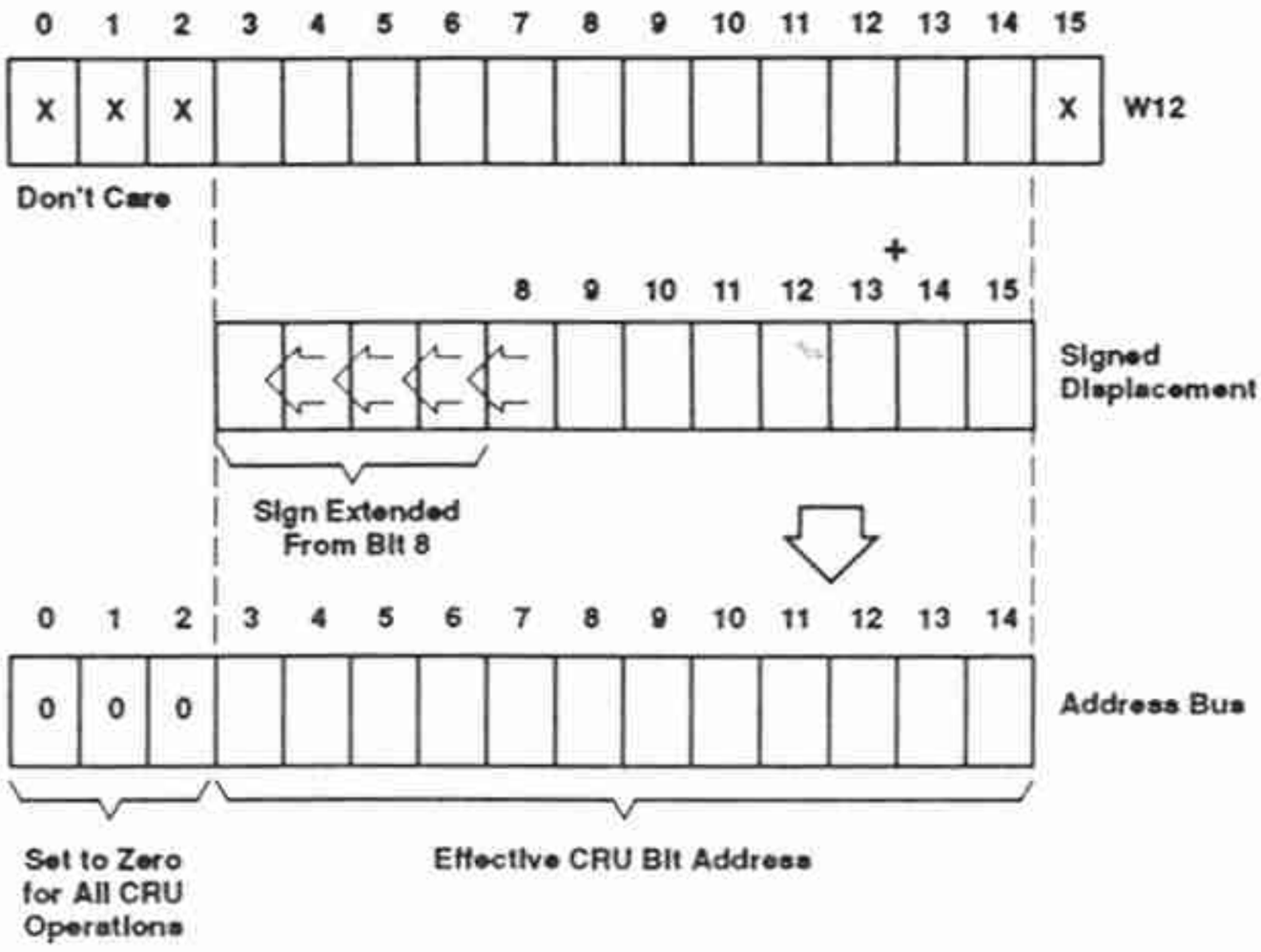


Figure 9. Bit Assignment for Single-Bit CRU Address

multiple-bit CRU operations

The SM/SMJ68689 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 10. Although Figure13 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. If the LDCR involves eight or fewer bits, those bits come from the right-justified field within the address 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 smaller transfer, the transferred data is 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 in the least-significant bit position in the memory word or byte.

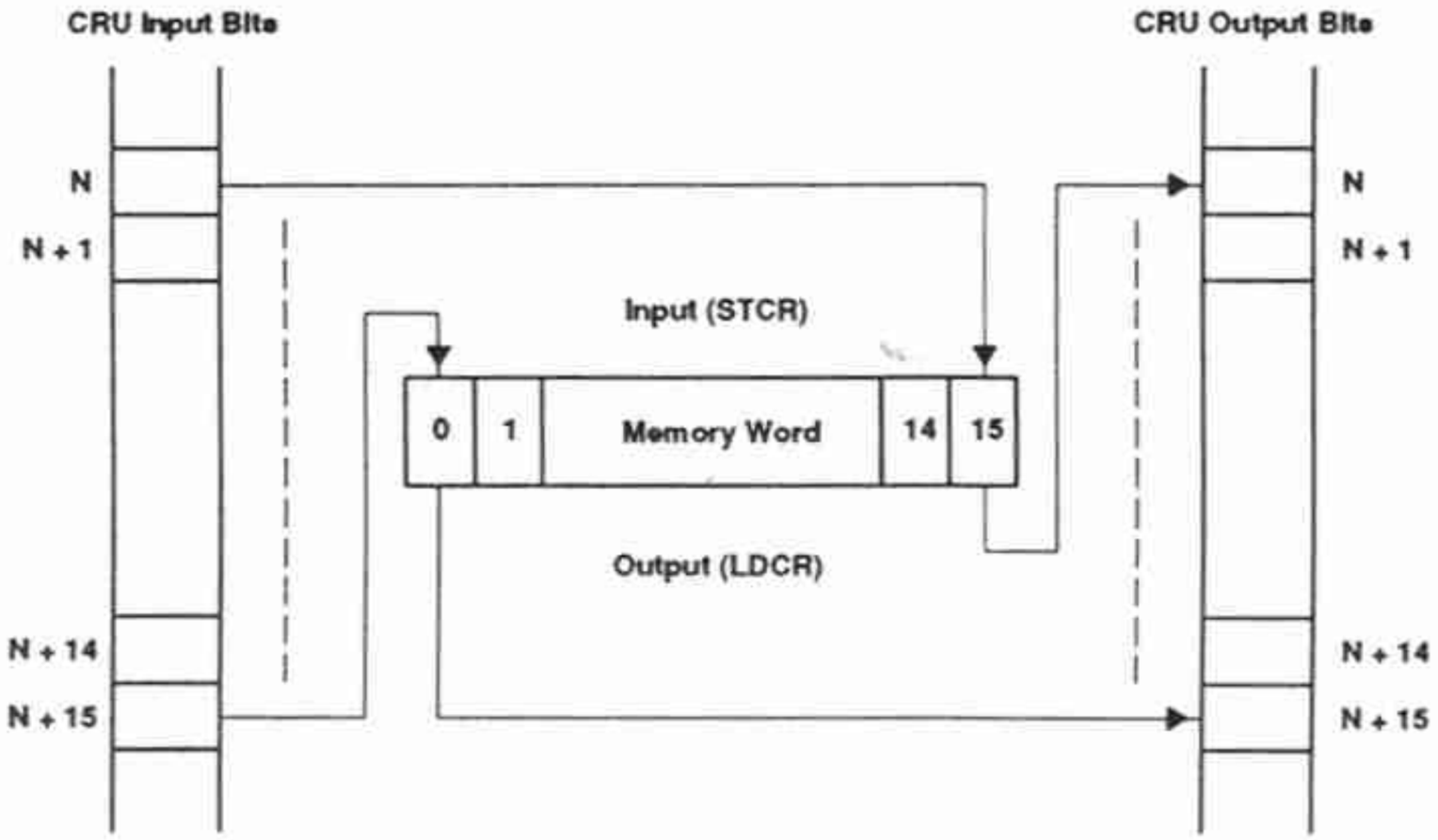


Figure 10. Bit Transfer With CRU Interface

external instructions

The SM/SMJ68689 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 and remain in the idle state 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 (A0–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 the CRUCLK pulse occur repeatedly until the idle state is terminated. The 3-bit codes are shown in Table 2.

load 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 asserted 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, and $\overline{\text{MPEN}}$ goes high. Program execution then resumes using the new PC and WP. During a hold state (caused by assertion of HOLD or $\overline{\text{XIPP}}$), the SM/SMJ68689 continues to sample the $\overline{\text{LOAD}}$ input. If $\overline{\text{LOAD}}$ is activated, the SM/SMJ68689 generates INTACK to indicate a pending $\overline{\text{LOAD}}$ needs servicing. INTACK remains active until $\overline{\text{HOLD}}$ or $\overline{\text{XIPP}}$ is released.

During non-hold states or after the SM/SMJ68689 has exited a hold state, $\overline{\text{LOAD}}$ must remain active until sampled by the SM/SMJ68689 near the end of an instruction period (i.e., $\overline{\text{LOAD}}$ could 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 $FFFC_{16}$. If $\overline{\text{LOAD}}$ remains active until the first instruction of the load routine is completed, the load trap sequence will repeat, overwriting the stored return vector in WR13–14. During hold states, $\overline{\text{LOAD}}$ shall remain active until an external device detects the activation of INTACK concurrent with a memory read operation at location $FFFC_{16}$.

load function (continued)

To use $\overline{\text{LOAD}}$ to initiate a bootstrap ROM sequence, $\overline{\text{LOAD}}$ should be asserted before $\overline{\text{RESET}}$ is released and remain active until either 1) the external device defeats the assertion of INTACK concurrent with a memory read operation at location $FFFC_{16}$ or 2) the first instruction execution begins as indicated by an active IAQ.

Table 2. External Instruction Function Table

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

standard instruction set

Each SM/SMJ68689 instruction performs one of the following operations:

1. Arithmetic, logical, comparison, or manipulation operation on data
2. Loading or storing 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 SM/SMJ68689 are defined below.

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 (rightmost) bit of n
MSB(n)	Most significant (leftmost) 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 bit 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

terms and definitions (continued)

TERM	DEFINITION
+	Arithmetic addition
–	Arithmetic subtraction
OR	Logical OR
⊗	Logical exclusive OR
n_	Logical complement of n
X	Arithmetic multiplication

addressing modes

The SM/SMJ68689 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:

- Workspace register addressing
- Workspace register indirect addressing
- Workspace register indirect autoincrement addressing
- Symbolic (direct) addressing
- Indexed addressing
- Immediate addressing
- Program-counter-relative addressing
- CRU-relative addressing

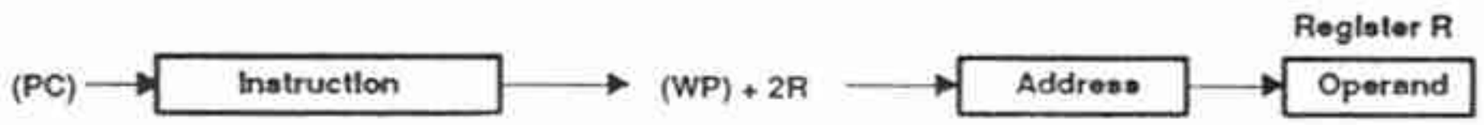
The following figures graphically describe the derivation of the effective address for each addressing mode. The applicability of addressing modes to particular instructions are described along with the description of the operations performed. The symbols following the names of the addressing modes (r, *R*R+, @LABLE, 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.

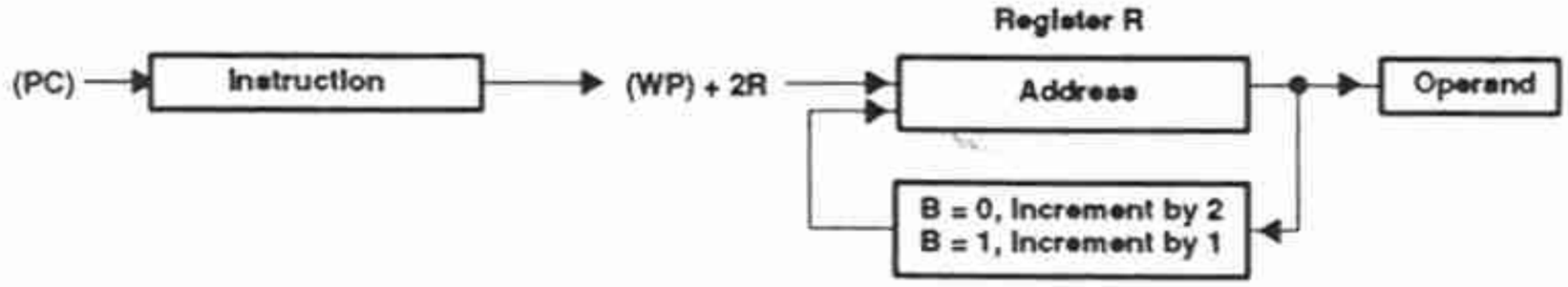
workspace register indirect addressing – *R

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.



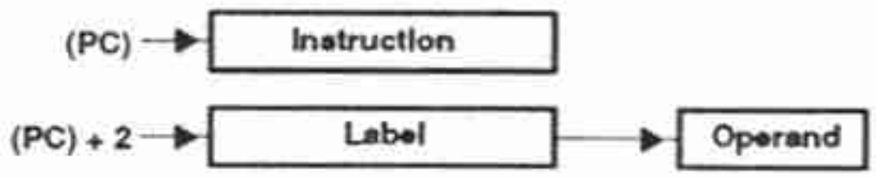
workspace register indirect autoincrement addressing – *R+

The workspace register indirect autoincrement 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 content of workspace register R is incremented.



symbolic (direct) addressing – @LABEL

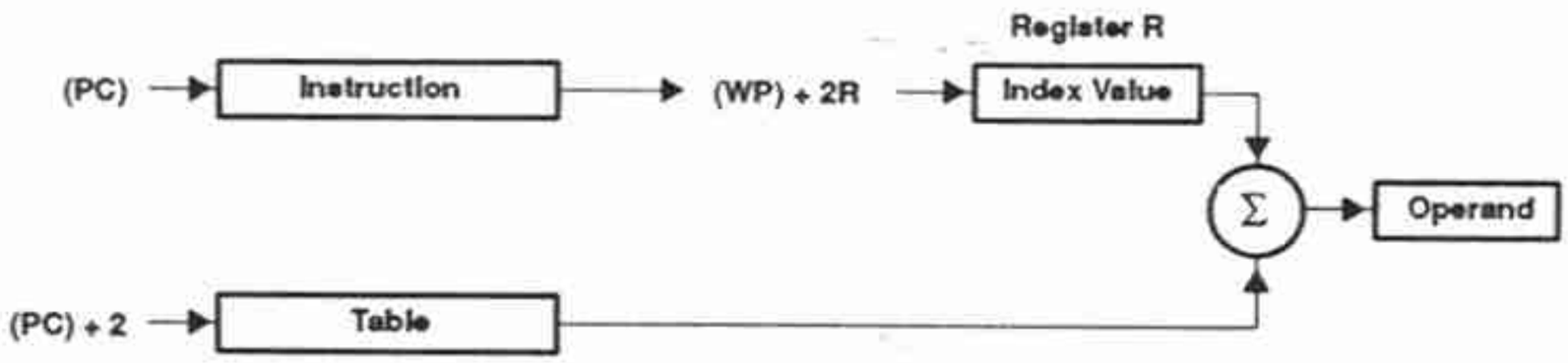
The symbolic addressing mode is specified by setting the 2-bit T field (T_S or T_D) in the instruction word equal to 10 and setting the corresponding S or D field equal to 0. The word following the instruction contains the address of the operand.



Indexed addressing – @TABLE(R)

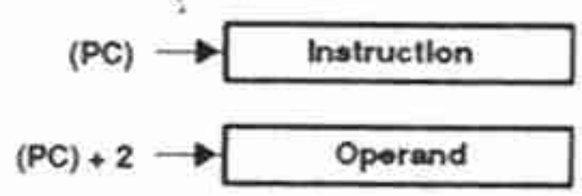
The indexed addressing mode is specified by setting the 2-bit T field (T_S or T_D) of the instruction word equal to 10. The value in the corresponding S or D field is the register which contains the index value. Register 0 may not be used for indexed addressing.

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.



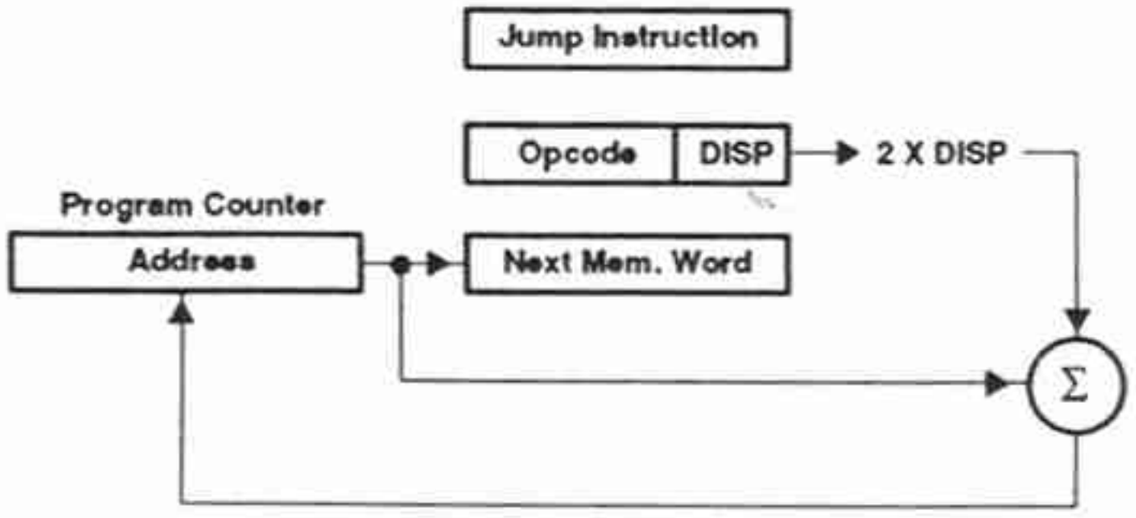
Immediate addressing

The word following the instruction contains the operand.



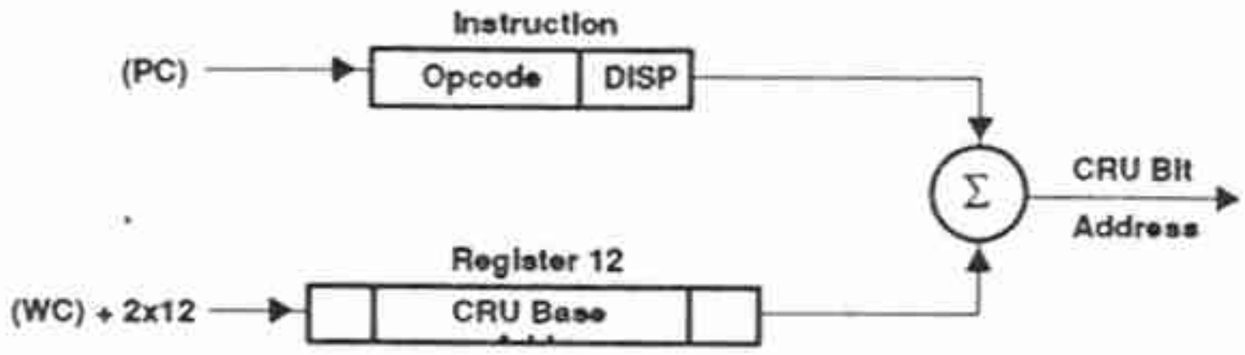
program-counter-relative addressing

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.



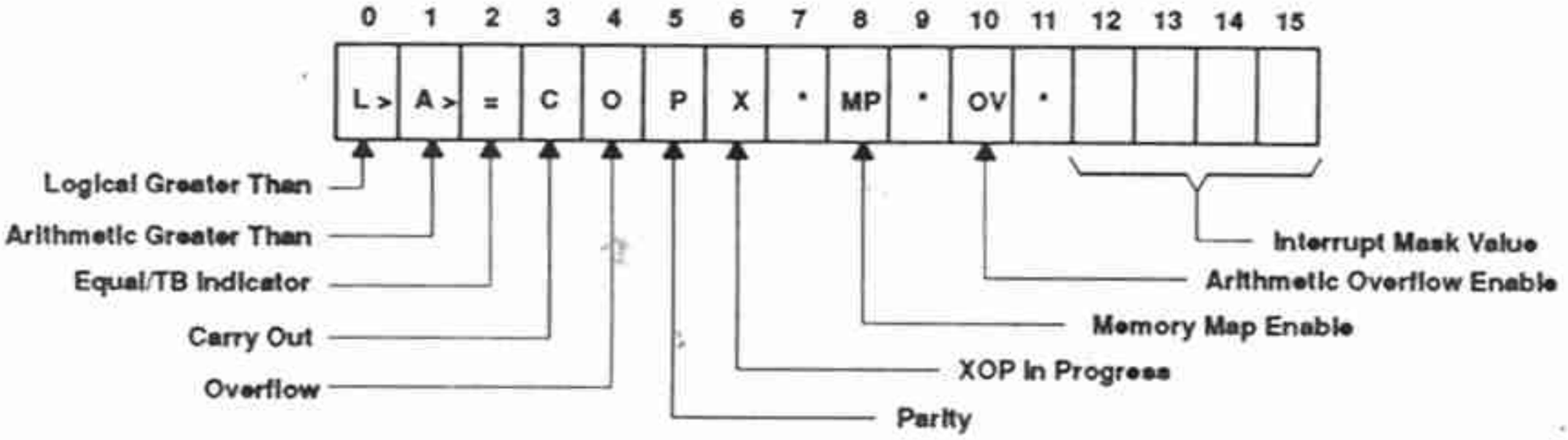
CRU-relative addressing

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



status register manipulation

Various SM/SMJ68689 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.



TEXAS INSTRUMENTS
POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

Table 3. Status Register Bit Definitions

BIT	NAME	INSTRUCTION	CONDITION TO SET BIT TO 1
		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
		C,CB	If MSB(SA) = 0 and MSB(DA) = 1, or if MSB(SA) = MSB(DA) and MSG 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 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
		C,CB	If (SA) = (DA)
		CI	If (wr) = IOP
		COC	If (SA) and (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
		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
		RTW	If bit (3) of WR15 is 1
		LST	If bit (3) of selected WR is 1
		A, AB	If MSB(SA) = MSB(DA) and MSB of result = MSB(DA)
		AI	If MSB(WR) = MSB or 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[(DA)–(SA)] = 0
		DVIS	If (SA) = 0000 or MSB(SA) = MSB(WR0) and $\lvert(2^{15} + 1)$ X (SA)$\rvert \leq$ 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

NOTE: Interrupt, $\overline{\text{LOAD}}$, XOPs, ILLOPs, and $\overline{\text{RESET}}$ operations set bits 7–11 to 0.

Table 3. Status Register Bit Definitions (continued)

BIT	NAME	INSTRUCTION	CONDITION TO SET BIT TO 1
ST5	Parity	CB, MOVB	If (SA) had odd number of 1s
		LDCR	If 1 ≤ C ≤ 8 and (SA) has odd number of 1s
		AB, SB, SOCB, SZCB	If result has odd number of 1s
		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 1s
ST6	XOP	XOP	If XOP instruction is executed
		RTWP	If bit (4) of WR15 is 1
		LST	If bit (6) of selected WR is 1
ST7, ST9 or ST11	User-defined	RTWP, LST	If corresponding bit of WR15 is 1 or 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 overflow enable	RTWP	If bit (10) of WR15 is 1
		LST	If bit (10) of selected WR is 1
ST2 thru ST15	Interrupt mask	LIMI	If corresponding bit of IOP is 1
		RTWP	If corresponding bit of WR15 is 1
		LST	If corresponding bit of selected WR is 1

NOTE: Interrupt, LOAD, XOPs, ILLOPs, and RESET operations set bits 7–11 to 0.

Instructions

Dual-operand instructions with multiple addressing for source operand and destination operands:

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

NOTES: 1. When 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	OPCODE	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)
BC	Compare bytes	100	1	No	0–2, 5	Compare (SA) to (DA) and set appropriate status bits
C	Compare	100	0	No	0–2	Compare (SA) to (DA) and set appropriate status bits
MOV	Move	110	0	Yes	0–2	(SA) → (DA)
MOVB	Move bytes	110	1	Yes	0–2, 5	(SA) → (DA)
S	Subtract	011	0	Yes	0–4	(SA) – (DA) → (DA)
SB	Subtract bytes	011	1	Yes	0–5	(SA) – (DA) → (DA)
SCZ	Set zeros corresponding	010	0	Yes	0–2	(DA) – ($\overline{SA}$) → (DA)
SCZB	Set zeros corresponding bytes	010	1	Yes	0–2, 5	(DA) – ($\overline{SA}$) → (DA)
SOC	Set ones corresponding	111	0	Yes	0–2	(DA) – (SA) → (DA)
SOCB	Set ones corresponding bytes	111	1	Yes	0–2, 5	(DA) – (SA) → (DA)

Instructions (continued)

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
Opcode						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 autoincrement	2

NOTES: 1. Workspace register 0 may not be used for indexing.
2. The workspace register is incremented by 2.

MNEMONIC	MEANING	OPCODE	RESULT COMPARED TO 0	STATUS BITS AFFECTED	DESCRIPTION
COC	Compare ones corresponding	001000	No	2	Test (D) to determine if 1s are in each bit position where 1s are in (SA). If so, set ST2.
CZC	Compare zeros corresponding	001001	No	0–2	Test (D) to determine if 0s are in each bit position where 1s are in (SA). If so, set ST2.
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.
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.
XOR	Exclusive OR	001010	Yes	0–2	(D) ⊕ (SA) → (D)

signed multiply and divide instructions

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Opcode										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 autoincrement	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	OPCODE	RESULT COMPARED TO 0	STATUS BITS AFFECTED	DESCRIPTION
DIVS	Signed divide	0000000110	Yes	0–2, 4	If (SA) = 0000 or if MSB (SA) = MSB (WR0) and |215 X (SA)| ≤ |WR (0,1)| or if MSB (SA) = MSB (WR0) and |(215+1) X (SA)| ≤ |WR (0,1)|, set ST4. Otherwise, divide signed 2s complement integer in WR0 and WR1 by the signed 2s 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.
MPYS	Signed multiply	0000000111	Yes	0–2	Multiply signed 2s-complement integer in WR0 by signed 2s complement integer (SA) and place signed 32-bit product in WR0 (most significant) and WR1 (least significant).

single-operand instruction

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

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

MNEMONIC	MEANING	OPCODE	RESULT COMPARED TO 0	STATUS BITS AFFECTED	DESCRIPTION
ABS	Absolute value†	0000011101	No	0–4	|(SA)| → (SA)
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)
DEC	Decerment	0000011000	Yes	0–4	(SA) – 1 → (SA)
DECT	Decerment by 2	0000011001	Yes	0–4	(SA) – 2 → (SA)
INC	Increment	0000010110	Yes	0–4	(SA) + 1 → (SA)
INCT	Increment by 2	0000010111	Yes	0–4	(SA) + 2 → (SA)
INV	Invert	0000010101	Yes	0–2	$(\overline{SA})$ → (SA)
NEG	Negate	0000010100	Yes	0–4	-(SA) → (SA)
SETO	Set to ones	0000011100	No		$FFFF_{16}$ → (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.
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 SM/SMJ68689 accesses the instruction at SA. Status bits are affected in the normal manner for the instruction executed.

CRU multiple-bit instruction

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

The C field specifies the number of bits to be transferred. If C = 0, 16 bits are 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 ($1 \leq C \leq 8$), the source address is a byte address. If 9 or more bits are transferred ($C \geq 9$), the source address is a word address. If the source is addressed in the workspace register indirect autoincrement mode, the workspace register is incremented by $1 \leq C \leq 8$; otherwise, the workspace register is incremented by 2.

MNEMONIC	MEANING	OPCODE	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 $1 \leq C \leq 8$

CRU single-bit instruction

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

CRU relative addressing is used to address the selected CRU bit.

MNEMONIC	MEANING	OPCODE	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
Opcode								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 2s complement form is a word count to be added to the 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	OPCODE	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 = 13

shift Instructions

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

If ≠ 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	OPCODE	RESULT COMPARED TO 0	STATUS BITS AFFECTED	DESCRIPTION
SAL	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 0s.

Immediate register instructions

0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Opcode											//////	WR			
IOP															

MNEMONIC	MEANING	OPCODE	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
Opcode												//////	//////	//////	//////
IOP															

MNEMONIC	MEANING	OPCODE	DESCRIPTION
LIMI	Load interrupt mask immediate	00000011000	Bits (12–15) of IOP → ST (12–15)
LWIP	Load workspace pointer immediate	00000010111	IOP → (WP), no ST bits affected.

Internal register load and store instructions

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

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

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 XOP is executed, ST6 is set, and the following transfers occur:

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

The SM/SMJ68689 does not test interrupt requests ($\overline{INTREQ}$) upon completion of the XOP instruction.

return workspace pointer (RTWP) instruction

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
Opcode											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 asserted, and the CRUCLK line to be pulsed, allowing external control functions to be initiated.

MNEMONIC†	MEANING	OPCODE	STATUS BITS AFFECTED	DESCRIPTION	ADDRESS BUS		
					A0	A1	A2
CKOF	User-defined	00000011110			H	H	L
CKON	User-defined	00000011101			H	L	H
IDLE	Idle	00000011010		Suspend SM/SMJ68689 instruction execution until an interrupt, LOAD or RESET occurs.	L	H	L
LREX	User-defined	00000011111			H	H	H
RSET	Reset	00000011011	7–15	0 → ST(7–15)	L	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 SM/SMJ68689 includes circuitry that indicates the completion of a microinstruction cycle. Designated as the $\overline{CYCEND}$ function, it provides CPU status that can simplify system design. The $\overline{CYCEND}$ output goes to a low logic level as a result of the low-to-high transition of each clock pulse that initiates the last clock cycle of a microinstruction.

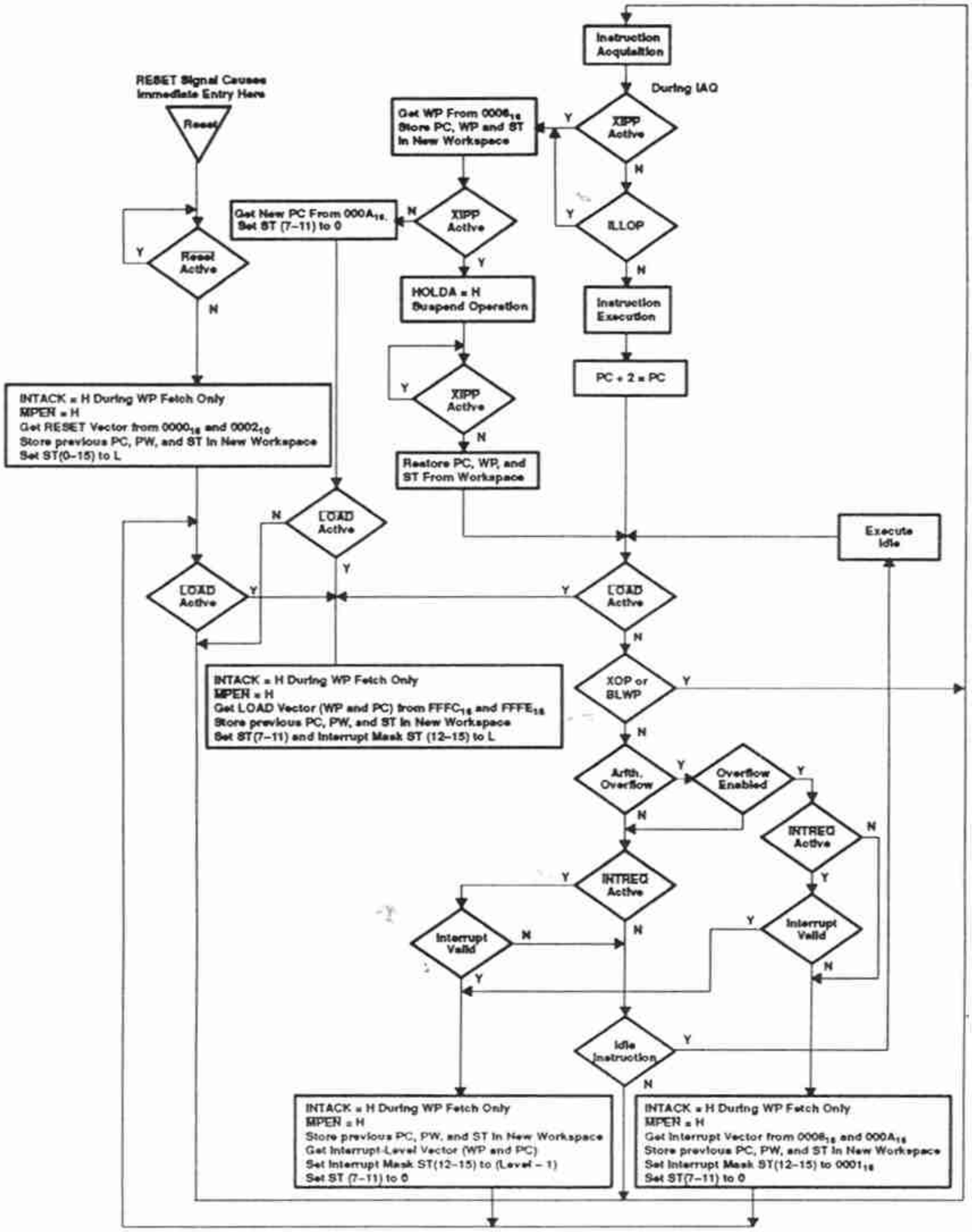


Figure 11. MicroInstruction Flow Chart

Instruction execution times

Instruction execution times for the SM/SMJ68689 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. Number 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 SM/SMJ68689 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 value 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 plus clock cycles for address modification)
W_M = number of required wait states per memory access
M_T = 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 $t_C = 0.06$ µs and no wait states are required to access memory. Both operands are addressed in workspace register mode. The instruction execution time is given by:

$$T = t_C[C_T + (W_M \times M_T)] + t_C (W_C \times P) = 0.06\ [12 + (0 \times 4)] + 0.06\ (0) = .72\ \mu s$$

If two wait states per memory access are required, the execution time will become:

$$= 0.06\ [12 + (2 \times 4)] + 0.06\ (0) = 1.2\ \mu s$$

If the source operands are addressed in the symbolic mode and two wait states are required:

$T = t_C[C_T + (W_M \times M_T)] + t_C (W_C \times P)$
$C_T = 2 + 6 = 18$
$M_T = 4 + 1 = 5$
$T = 0.06\ [\ 18 + (2 \times 5)] + 0.06\ (0) = 1.7\ \mu s$

TEXAS INSTRUMENTS
POST OFFICE BOX 655303 • DALLAS, TEXAS 75265

INSTRUCTION	CLOCK CYCLE		MEMORY CYCLES		ADDRESS MODIFICATION		CRU OPERATIONS
	SMJ68689	Δ FROM SBP9900A	SMJ68689	Δ FROM SBP9900A	SOURCE	DEST	
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		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	+ 4	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				

INSTRUCTIONS	CLOCK CYCLES		MEMORY CYCLES		ADDRESS MODIFICATION		CRU OPERATIONS
	SMJ68689	Δ FROM SMJ68689	SMJ68689	Δ FROM SMJ68689	SOURCE	DEST	
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				
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						
(C = 0, Bits (12–15) of WR) = 0)	52						
(C = 0, Bits (12–15) of WR) ≠ 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	
SCZB	12	– 2	4		Table 4	Table 4	
TB	12		2				1

INSTRUCTION	CLOCK CYCLES		MEMORY CYCLES		ADDRESS MODIFICATION		CRU OPERATIONS
	SMJ68689	Δ FROM SMJ68689	SMJ68689	Δ FROM SMJ68689	SOURCE	DEST	
XT	4		1		Table 4		
XOP	28	– 8	7	–1			
XOR	12	– 2	4		Table 4		
Reset function	12	– 4	5				
Load function	20	– 2	5				
Interrupt							
Context switch	20	– 2	5				
Undefined Opcodes‡							
$0000_{16} – 007F_{16}$	24	+ 18	6	+ 5			
$00A0_{16} – 017F_{16}$							
$0320_{16} – 033F_{16}$							
$0780_{16} – 07FF_{16}$							
$0C00_{16} – 0FFF_{16}$							

† 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 $\overline{XIPP}$ being released
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
	SMJ68689	Δ FROM SMJ68689	
WR(T_S or T_D = 00)	0		0
WR indirect (T_S or T_D = 01)	4		1
WR indirect autoincrement (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 SM/SMJ68689 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 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 an I/O device. 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 at the end of the second 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.

CRU cycle (continued)

A special feature of SM/SMJ68689 operation not available on its forerunner, the SBP9900A, is the capability to insert wait states. The SM/SMJ68689 samples the READY line at the beginning of the second clock cycle.

SM/SMJ68689 machine cycle sequences

Most SM/SMJ68689 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 description.

terms and definitions

The following terms and definitions are used in describing the SM/SMJ68689 instruction set:

TERM	DEFINITION
C	Bit count
Nd	Number of machine cycles to derive the destination operand
Ns	Number of machine cycles to derive the source operand
Nc	Number of machine cycles to transfer/shift C bits

data derivation sequence

WORKSPACE REGISTER

CYCLE	TYPE
0	Memory read

NOTE
Fastest addressing mode; no additional clock cycles for source or destination aquisition. Read will already have occurred in the opcode instruction fetch. Therefore, Ns = Nd = 0 when using workspace register addressing mode.

WORKSPACE REGISTER

CYCLE	TYPE
1	Memory read
2	ALU

NOTE
Ns = Nd = 2

WORKSPACE REGISTER AUTO-INCREMENT

CYCLE	TYPE
1	Memory read
2	ALU
3	Memory write

NOTE
Ns = Nd = 3

SYMBOLIC

CYCLE	TYPE
1	ALU
2	Memory read
3	ALU

NOTE
Ns = Nd = 3

INDEXED

CYCLE	TYPE
1	Memory read
2	Memory read
3	ALU

NOTE
Ns = Nd = 3

Instruction execution sequence

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

CYCLE	TYPE
0	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 = 0)

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

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

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

Instruction execution sequence (continued)

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
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 = 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)

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

Instruction execution sequence (continued)

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	ALU

LI

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

LIMI

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	ALU
10	Memory write

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

Instruction execution sequence (continued)

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
6	Memory read
5	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	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

Instruction execution sequence (continued)

X

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

XOP

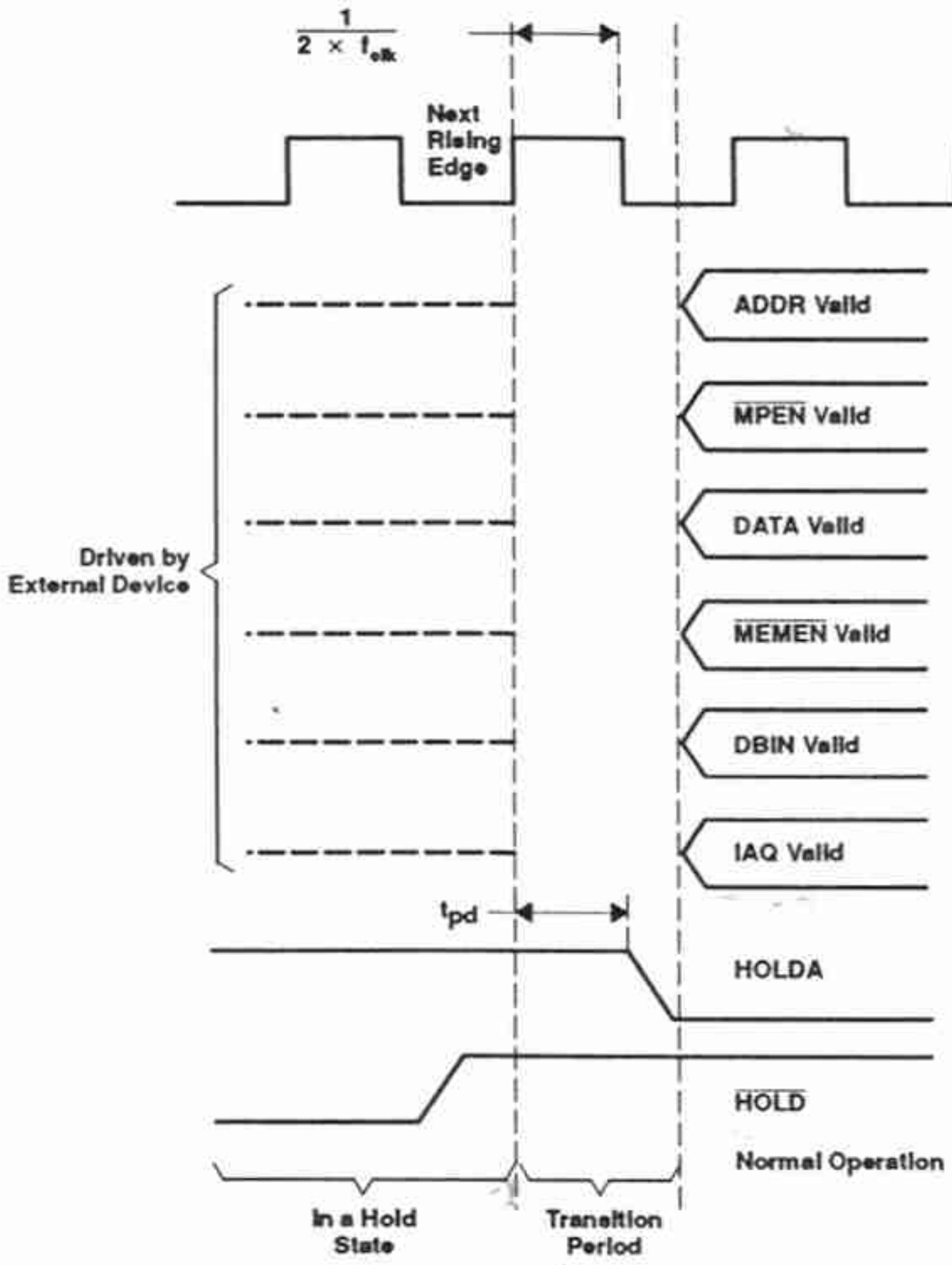
CYCLE	TYPE
1	Memory read
2	ALU
Ns	Source acquisition
3+Ns	ALU
4+Ns	Memory read
5+Ns	ALU
6+Ns	Memory write
7+Ns	ALU
8+Ns	Memory write
9+Ns	ALU
10+Ns	Memory write
11+Ns	ALU
12+Ns	Memory write
13+Ns	Memory read
14+Ns	ALU

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 SM/SMJ68689 leaves a hold state, the time required for the signals to return to their proper levels is not much different from the other delay times. If your system does use hold states, there will be no problems.



absolute maximum ratings over operating free-air temperature range (unless otherwise noted)

Supply voltage range, V_{CC} ..... – 0.5 V to 7.0 V
Output voltage range, V_O ..... – 0.5 V to 7.0 V
Input voltage range, V_I ..... – 0.5 V to 7.0 V
Storage temperature range ..... – 65°C to 150°C

† Stresses beyond those listed under "absolute maximum ratings" may cause permanent damage to the device. This is a stress rating only, and functional operation of the device at these or any other conditions beyond those indicated in the "recommended operating conditions" section of this specification is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

recommended operating conditions

			MIN	NOM	MAX	UNIT
V_{CC}	Supply voltage		4.5	5	5.5	V
V_I†	Input voltage				V_{CC}	V
V_O‡	Output voltage				V_{CC}	V
	Clock frequency		1		16	MHz
	Clock pulse width		31			ns
t_r	Clock rise time				6	ns
t_f	Clock fall time				6	ns
t_{su}	Setup time	$\overline{HOLD}$	0			ns
		READY	5			
		D0 thru D15	8			
		CRUIN	10			
		$\overline{INTREQ}$	5			
		IC0 thru IC3	5			
		$\overline{XIPP}$	5			
		$\overline{LOAD}$	10			
		$\overline{RESET}$	0			
t_h	Hold time	$\overline{HOLD}$	10			ns
		READY	5			
		D0 thru D15	4			
		CRUIN	5			
		$\overline{INTREQ}$	5			
		IC0 thru IC3	5			
		$\overline{XIPP}$	5			
		$\overline{LOAD}$	5			
		$\overline{RESET}$	10			
T_A	Operating temperature		–55		125	°C

† Applies for external input and bidirectional buffers
‡ Applies for external output and bidirectional buffers

electrical characteristics over recommended operating free-air temperature range (unless otherwise noted)

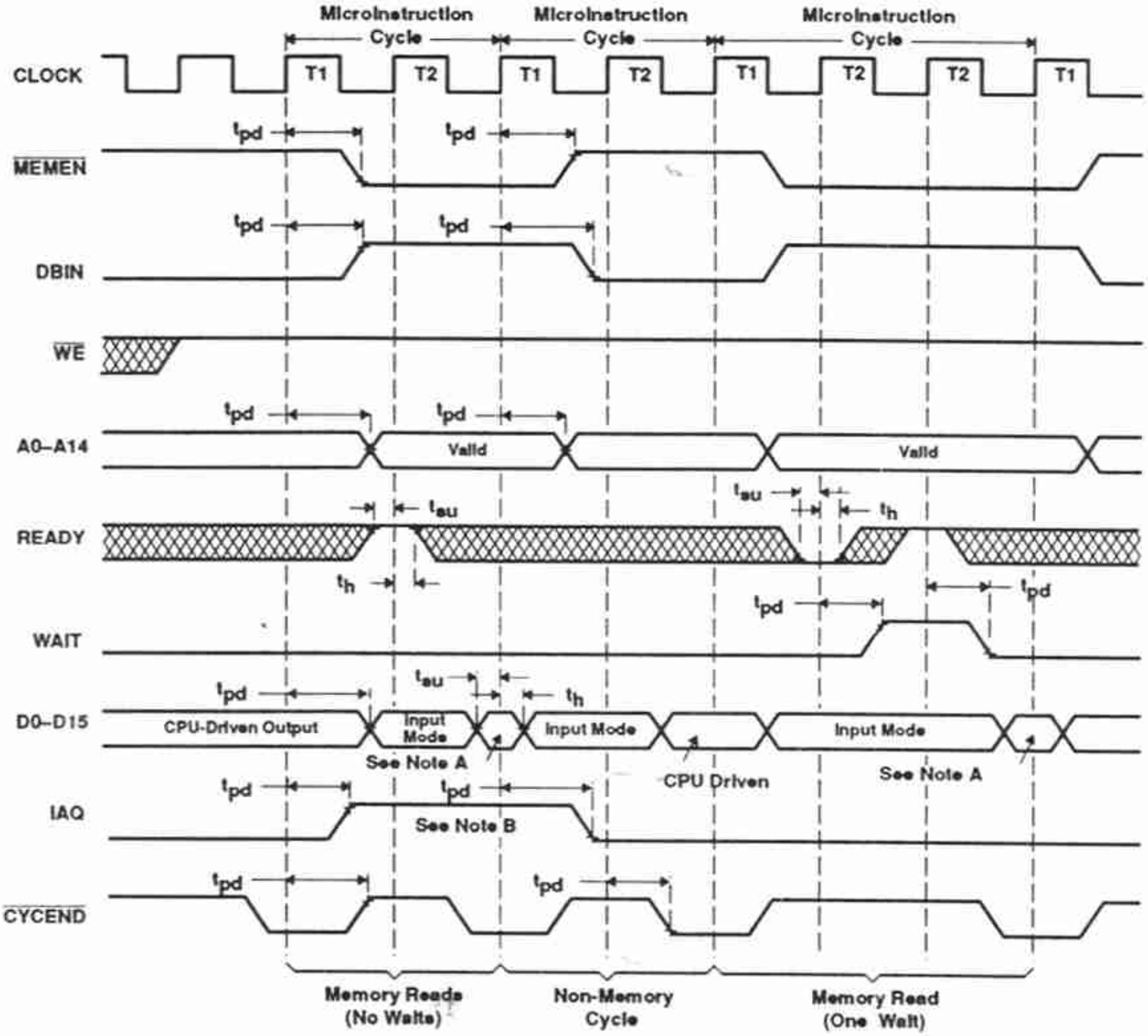
PARAMETER			TEST CONDITIONS		MIN	TYP	MAX	UNIT
V_{OHL}	High-level output voltage w/ loads		V_{CC} = 4.5 V, V_{IL} = 0,	V_{IH} = 3 V, I_{OH} = –13.6 mA	3.8			V
V_{OLL}	Low-level output voltage w/ loads		V_{CC} = 4.5 V, V_{IL} = 0,	V_{IH} = 3 V, I_{OL} = 20.4 mA			.5	V
V_{OHN}	High-level output voltage w/o loads		V_{CC} = 5.5 V, V_{IL} = 0,	V_{IH} = 5.5 V I_{OH} = –20 µA	5.4			V
V_{OLN}	Low-level output voltage w/o loads		V_{CC} = 5.5 V, V_{IL} = 0,	V_{IH} = 5.5 V, I_{OL} = 20 µA			.1	V
I_I	Input current	Clock	V_{CC} = 5.5 V,	V_I = V_{CC} or 0			± 1	µA
		Other inputs and I/Os					± 10	
I_{OZ}	Off-state output current		V_{CC} = 5.5 V,	V_O = V_{CC} or 0			± 10	µA
I_{CCQ}	Quiescent supply current		V_{CC} = 5.5 V, V_{IL} = 0	V_{IH} = 5.5 V,	– 5		10	µA
I_{CCD}	Dynamic supply current		V_{CC} = 5.5 V			25	50*	mA
C_i	Input capacitance					7		pF
C_o	Output capacitance	Includes I/Os				11		pF

*This parameter is not production tested.

switching characteristics over recommended operating free-air temperature range (unless otherwise noted)

PARAMETER	FROM	TO	TEST CONDITIONS	MIN	TYP	MAX	UNIT
t_{PD}	CLK	Outputs and I/Os	C_L for outputs = 55 pF, for I/Os = 75 pF R_L = 2 kΩ			50	ns
t_{PD}	CLK	Address bus, $\overline{MPEN}$, data bus, $\overline{MEMEN}$, DBIN, IAQ	C_L for outputs = 55 pF, for I/Os = 75 pF R_L = 2 kΩ When leaving a Hold state			50	ns

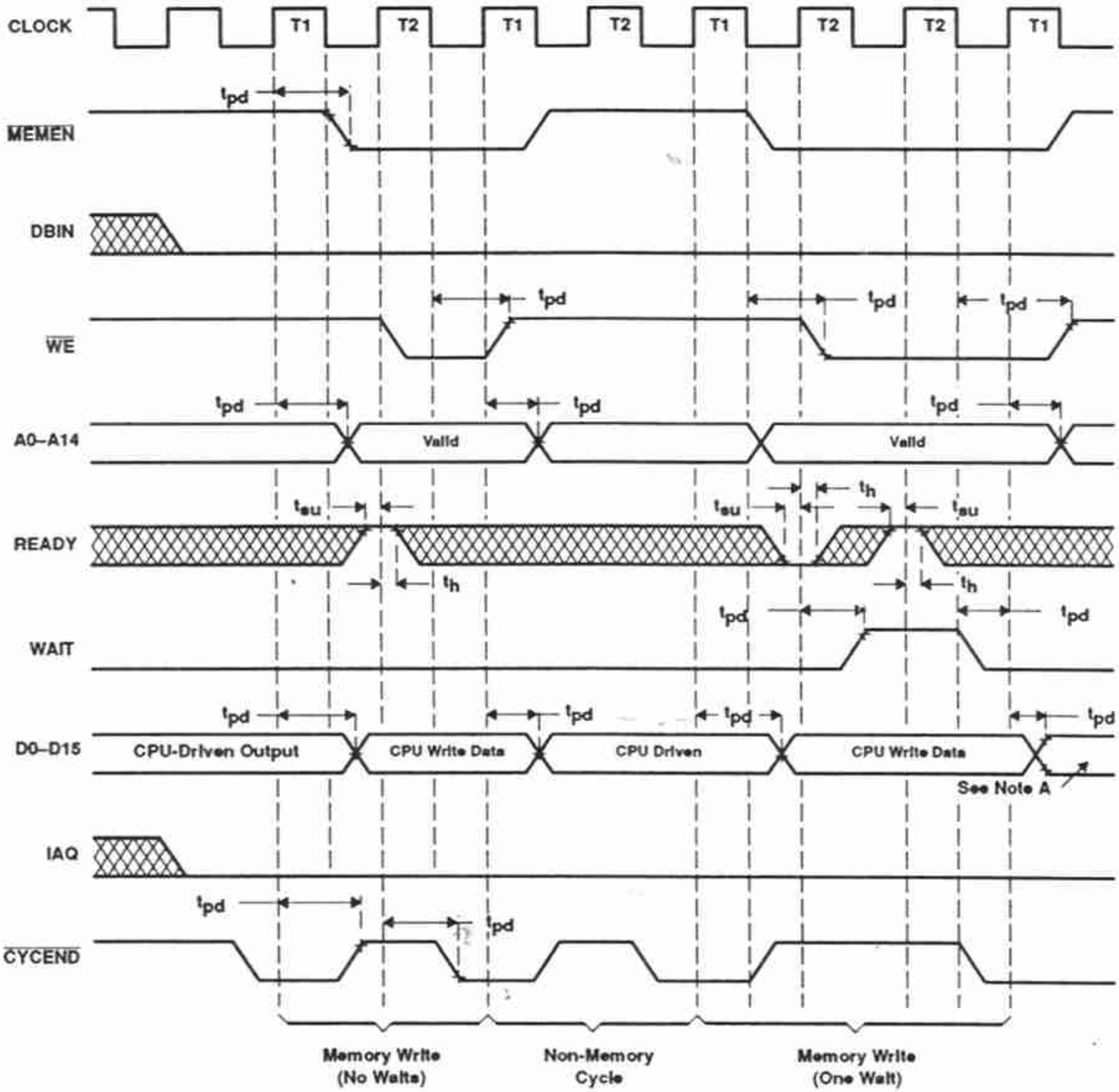
PARAMETER MEASUREMENT INFORMATION



NOTES: A. Valid memory read data
B. Show assuming IAQ cycle

Figure 12. Memory Bus Timing for Read Operation

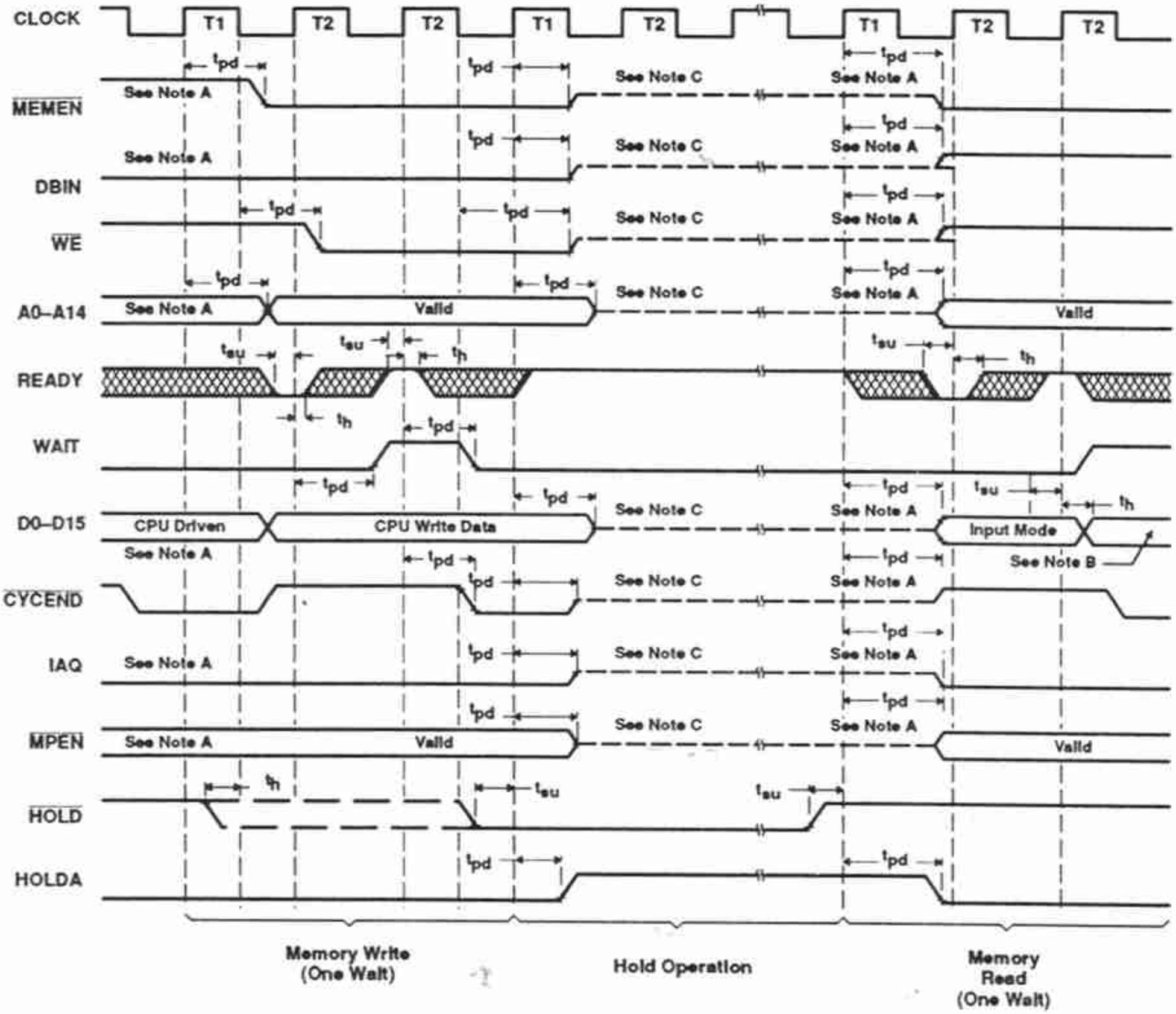
PARAMETER MEASUREMENT INFORMATION



NOTE A. CPU driven

Figure 13. Memory Bus Timing for Write Operation

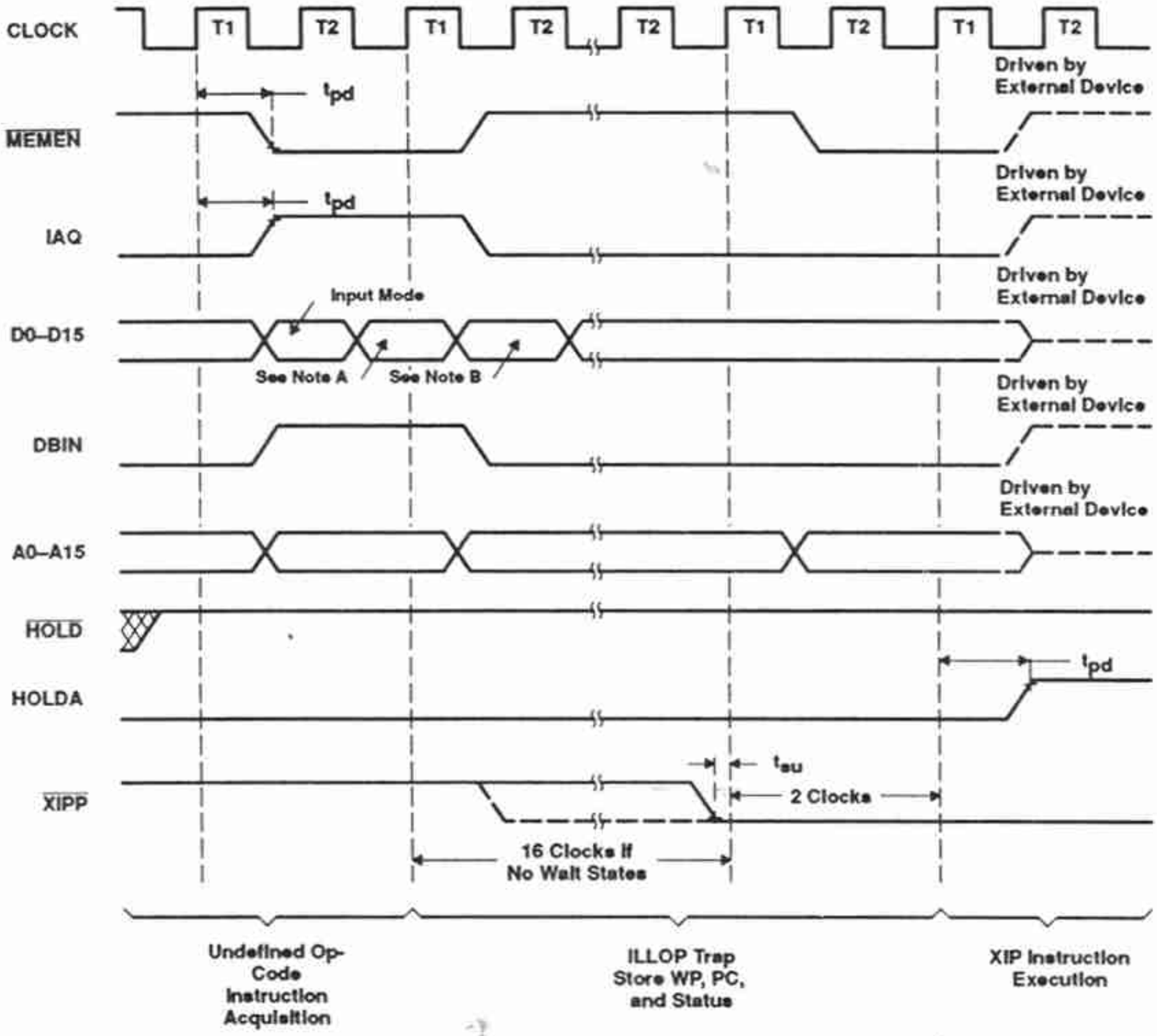
PARAMETER MEASUREMENT INFORMATION



NOTES: A. There is a maximum delay of 200 ns for these signals to return to valid level on leaving a hold state.
B. Valid memory read data
C. Driven by external device

Figure 14. Memory Bus Timing for Hold Operation

PARAMETER MEASUREMENT INFORMATION



NOTES: A. Valid undefined opcode from memory
B. Input mode

Figure 15. Extended-Instruction Processor Interface Timing (ILLOP)

PARAMETER MEASUREMENT INFORMATION

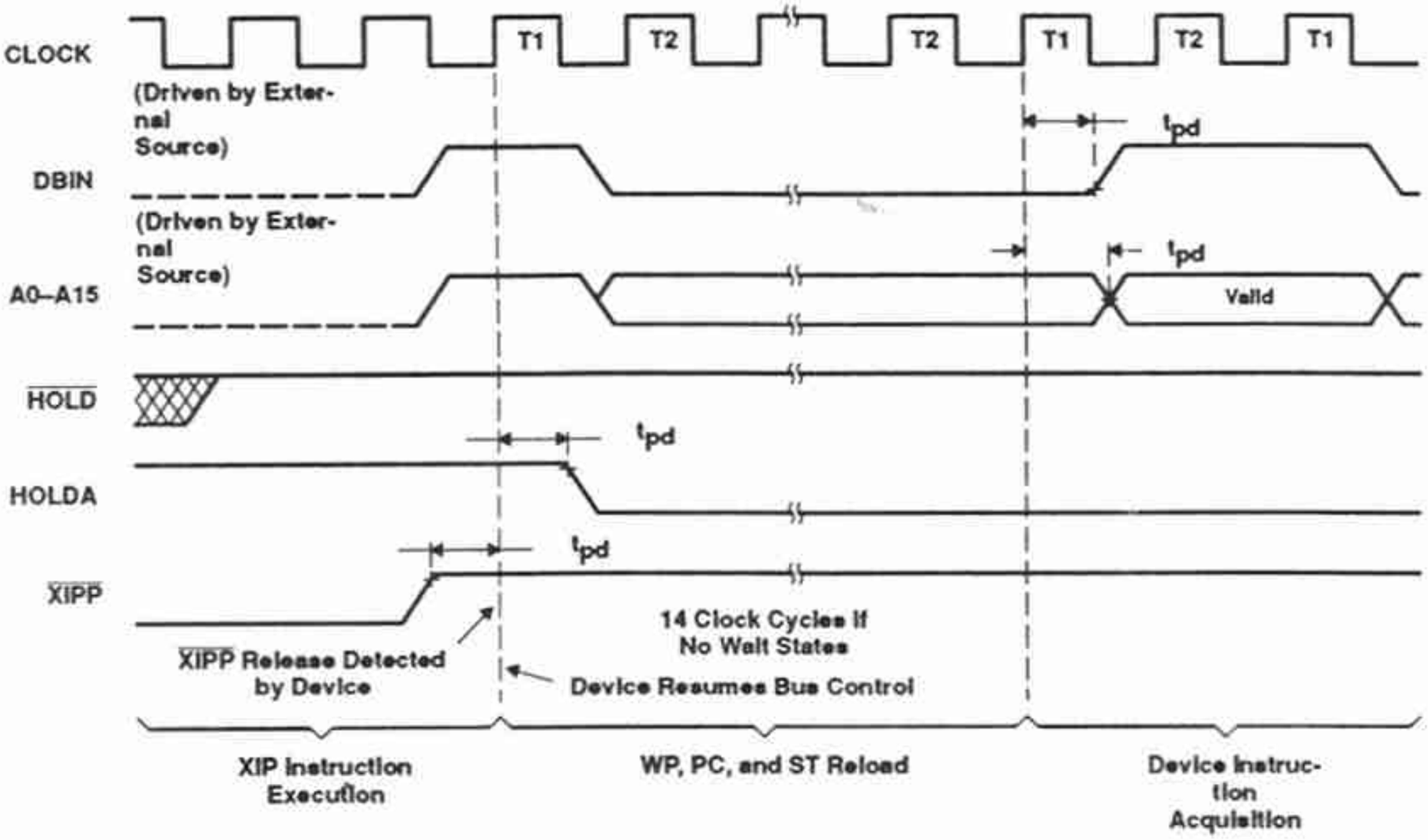


Figure 16. Extended-Instruction Processor Interface Timing (WP, PC, ST, RELOAD)

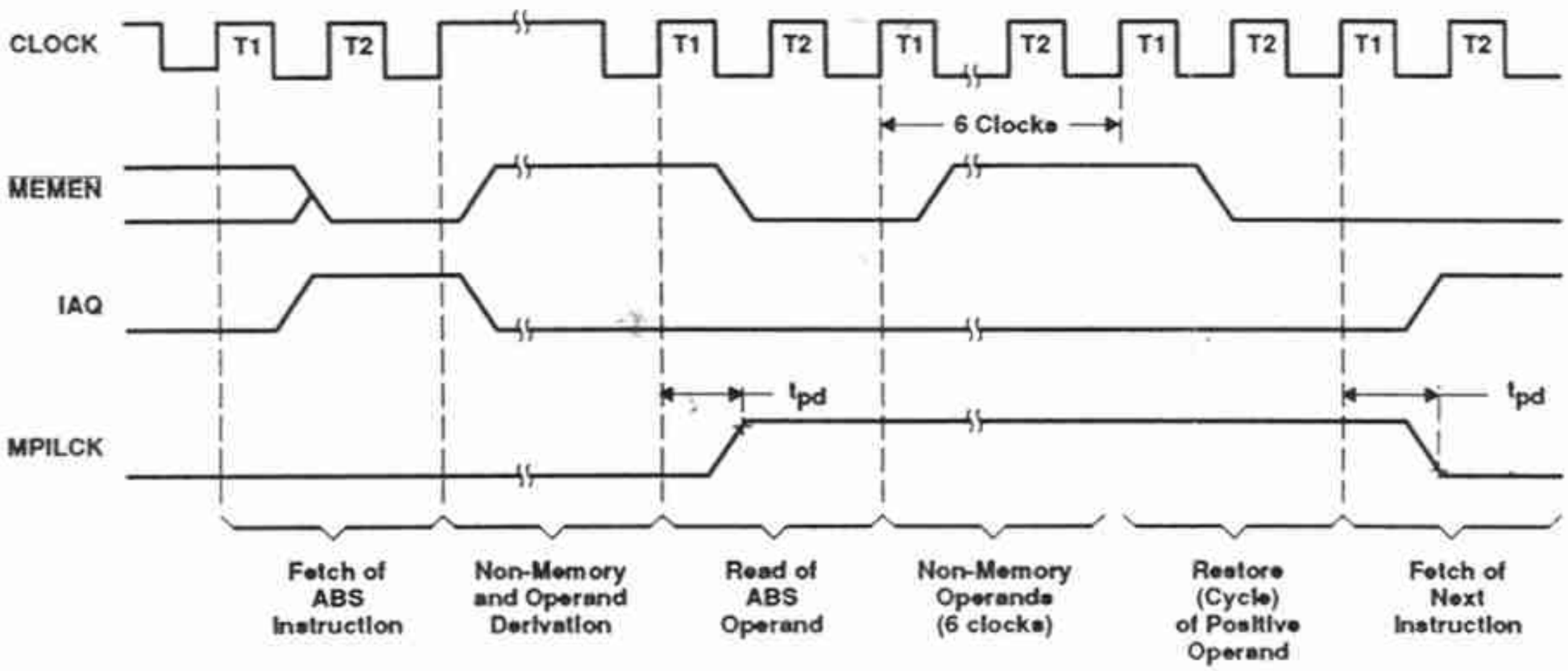


Figure 17. Multiprocessor Interlock Timing (Negative Operand)

PARAMETER MEASUREMENT INFORMATION

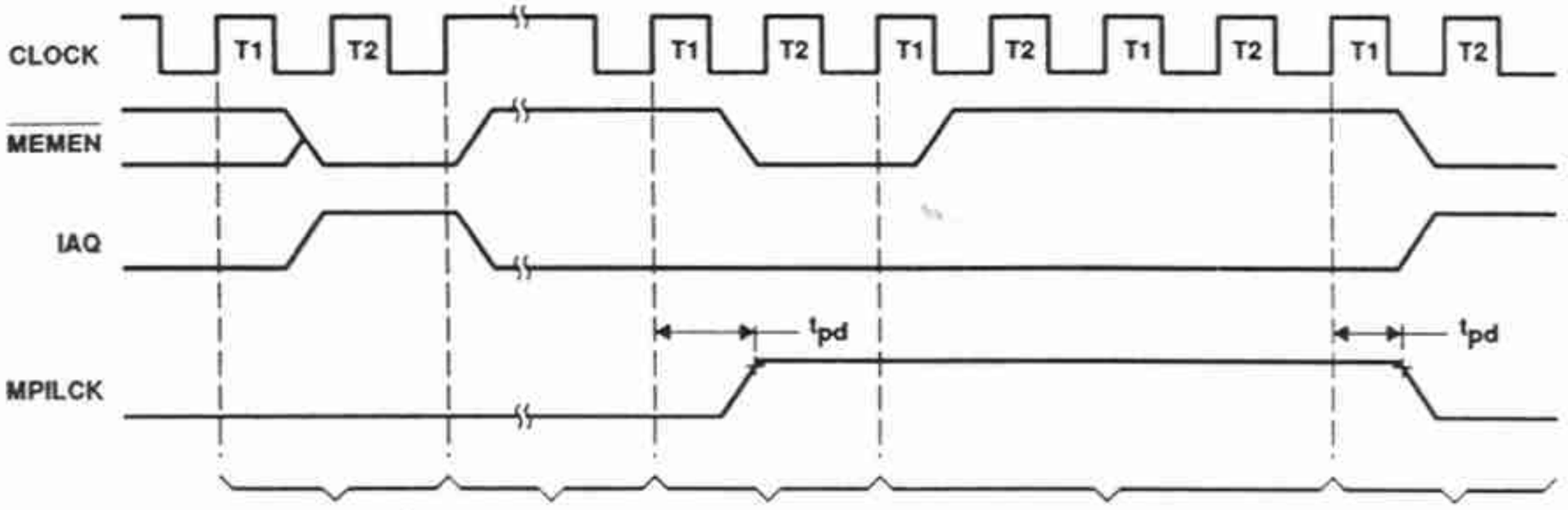
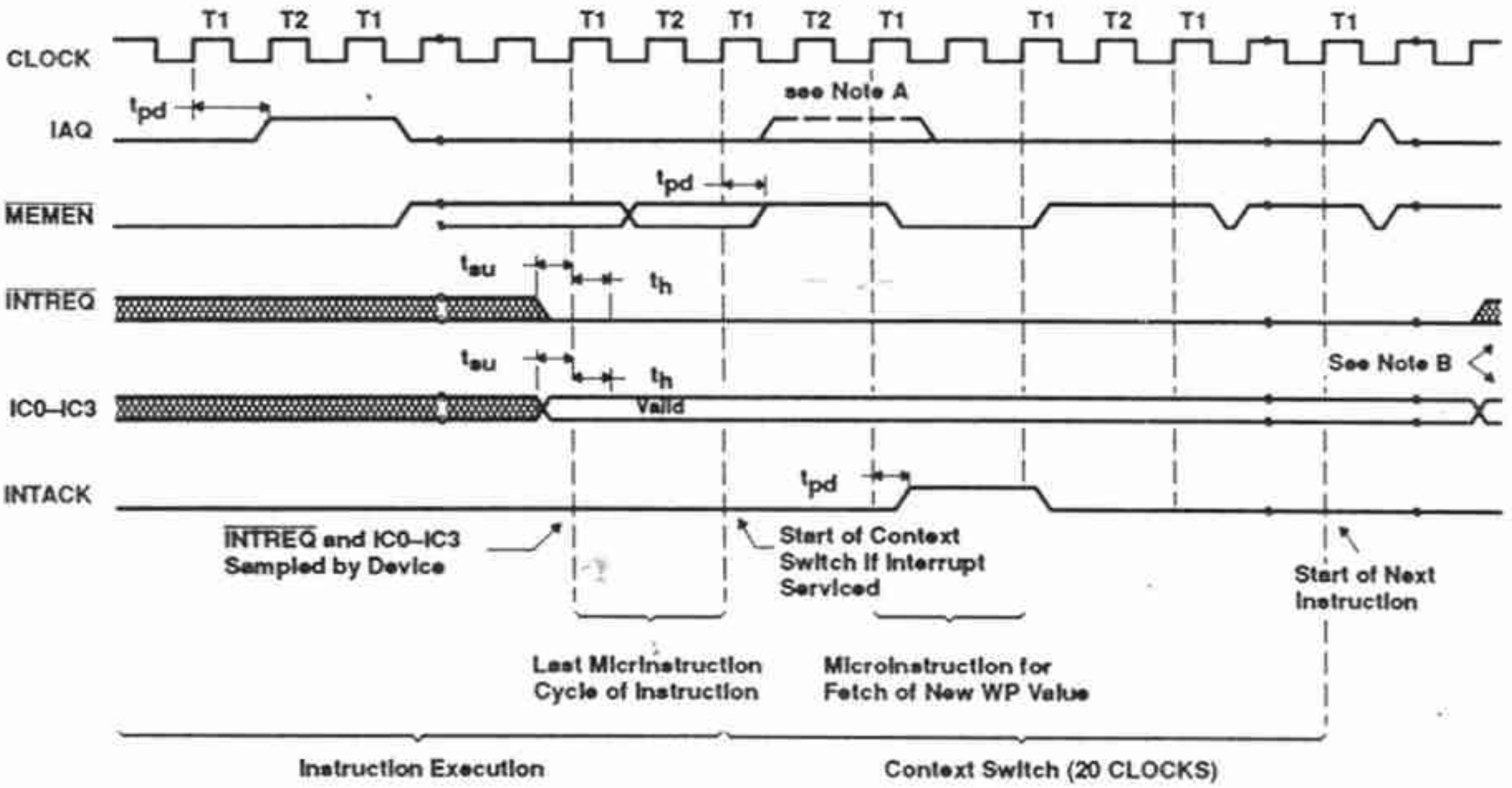


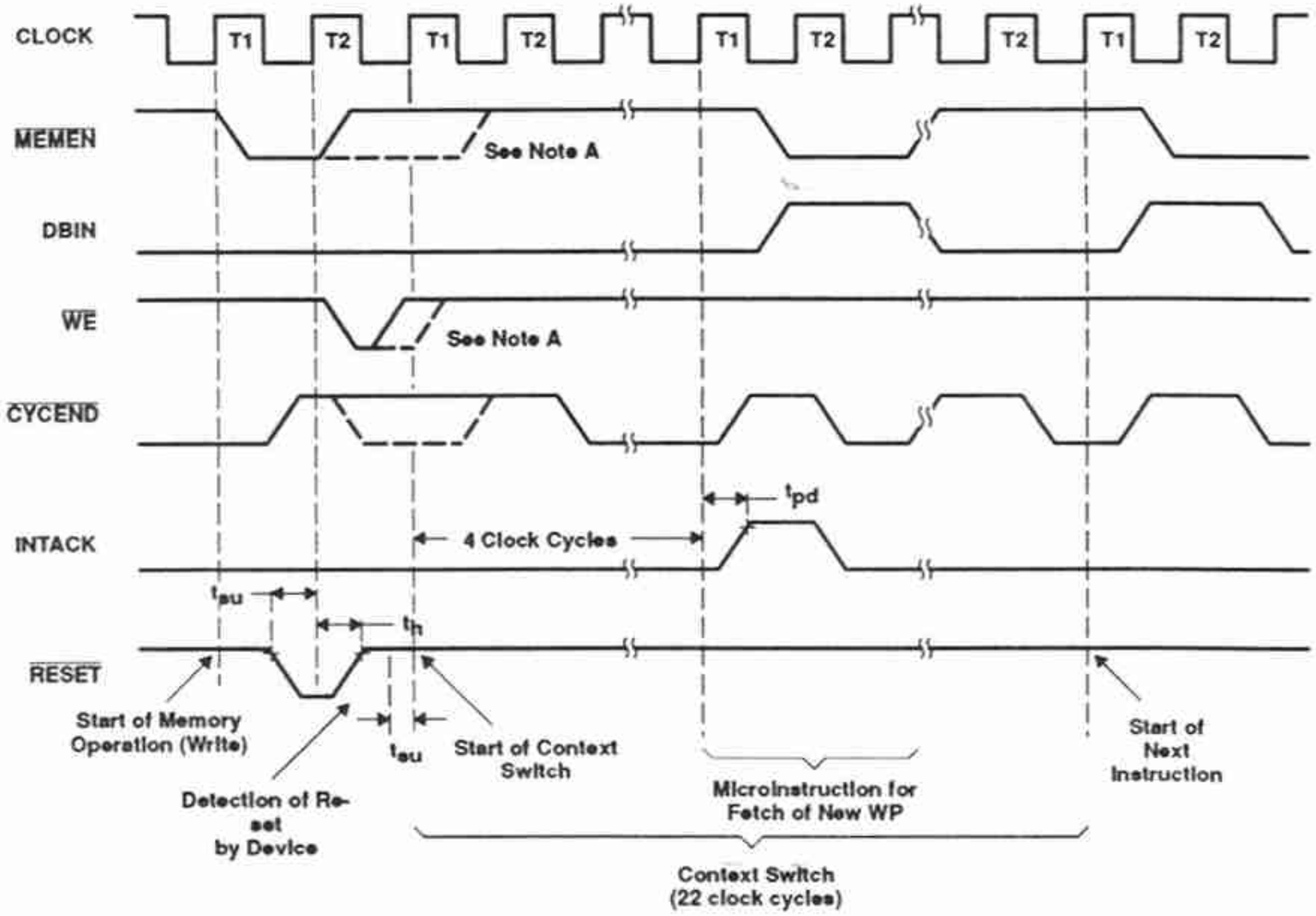
Figure 18. Multiprocessor Interlock Timing (Positive Operand)



NOTES: A. Next IAQ if interrupt not serviced
B. $\overline{\text{INTREQ}}$ and IC0–IC3 changed by interrupt service routine
C. This diagram assumes valid IC is of higher priority than code in status register.

Figure 19. Interrupt Timing

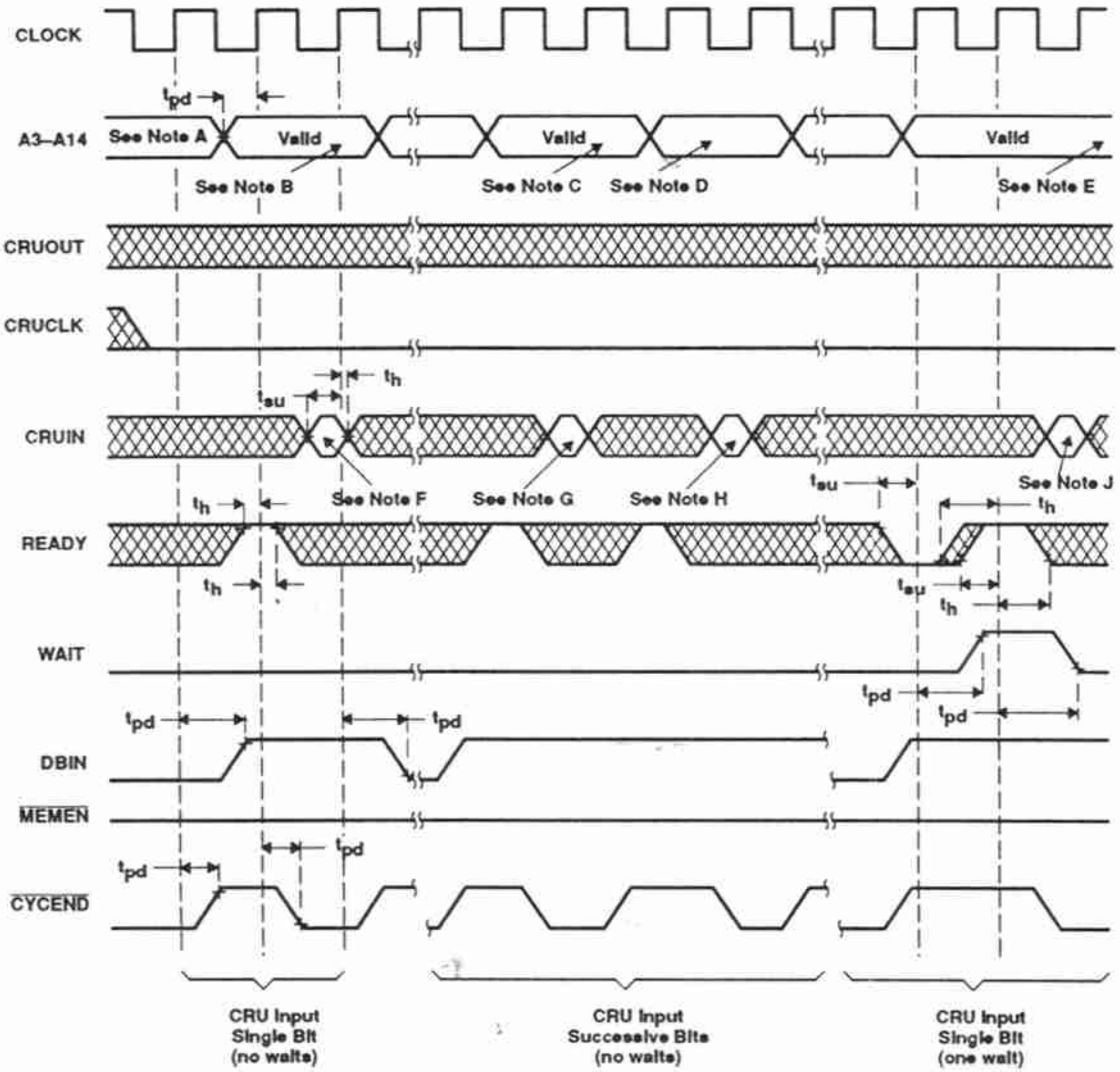
PARAMETER MEASUREMENT INFORMATION



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 20. Reset Timing

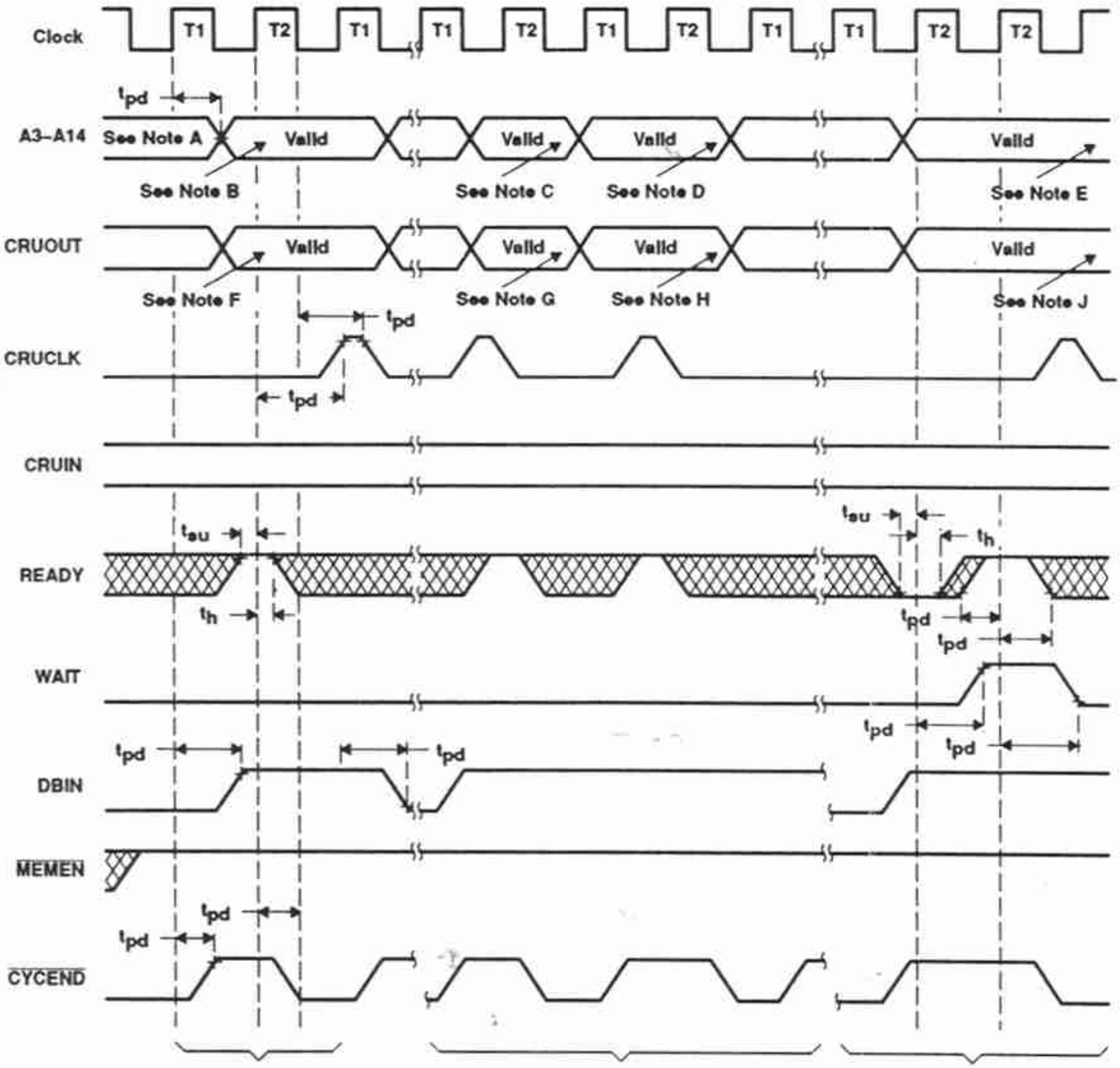
PARAMETER MEASUREMENT INFORMATION



NOTES: 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+1
I. CRU data out address R

Figure 21. CRU Timing (Input Operation)

PARAMETER MEASUREMENT INFORMATION



NOTES: 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+1
I. CRU data out address R

Figure 22. CRU Timing (Output Operation)