National Semiconductor

November 1991

DP8344B Biphase Communications Processor—BCP®

General Description

The DP8344B BCP is a communications processor designed to efficiently process IBM® 3270, 3299 and 5250 communications protocols. A general purpose 8-bit protocol is also supported.

The BCP integrates a 20 MHz 8-bit Harvard architecture RISC processor, and an intelligent, software-configurable transceiver on the same low power microCMOS chip. The transceiver is capable of operating without significant processor interaction, releasing processor power for other tasks. Fast and flexible interrupt and subroutine capabilities with on-chip stacks make this power readily available.

The transceiver is mapped into the processor's register space, communicating with the processor via an asynchronous interface which enables both sections of the chip to run from different clock sources. The transmitter and receiver run at the same basic clock frequency although the receiver extracts a clock from the incoming data stream to ensure timing accuracy.

The BCP is designed to stand alone and is capable of implementing a complete communications interface, using the processor's spare power to control the complete system. Alternatively, the BCP can be interfaced to another processor with an on-chip interface controller arbitrating access to data memory. Access to program memory is also possible, providing the ability to download BCP code.

A simple line interface connects the BCP to the communications line. The receiver includes an on-chip analog comparator, suitable for use in a transformer-coupled environment, although a TTL-level serial input is also provided for applications where an external comparator is preferred.

A typical system is shown below. Both coax and twinax line interfaces are shown, as well as an example of the (optional) remote processor interface.

Features

Transceiver

- Software configurable for 3270, 3299, 5250 and general 8-bit protocols
- Fully registered status and control
- On-chip analog line receiver

Processor

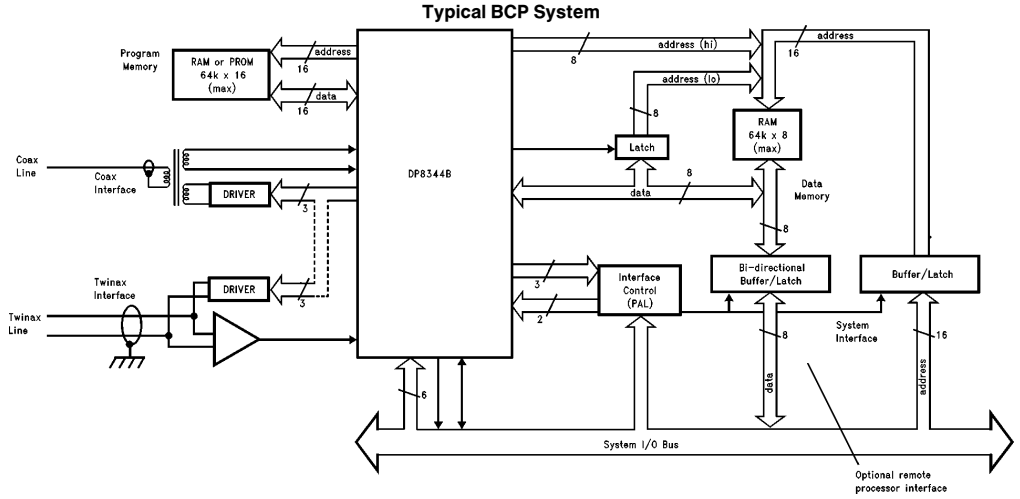
- 20 MHz clock (50 ns T-states)
- Max. instruction cycle: 200 ns
- 33 instruction types (50 total opcodes)
- ALU and barrel shifter
- 64k x 8 data memory address range
- 64k x 16 program memory address range (note: typical system requires <2k program memory)
- Programmable wait states
- Soft-loadable program memory
- Interrupt and subroutine capability
- Stand alone or host operation
- Flexible bus interface with on-chip arbitration logic

General

- Low power microCMOS; typ. I_{CC} = 25 mA at 20 MHz
- 84-pin plastic leaded chip carrier (PLCC) package

Block Diagram

Typical BCP System



TL/F/9336–51

FIGURE 1

BCP® and TRI-STATE® are registered trademarks of National Semiconductor Corporation.
IBM® is a registered trademark of International Business Machines Corporation.

RRD-B30M105/Printed in U. S. A.

The DP8344B is an enhanced version of the DP8344A, exhibiting improved switching performance and additional functionality. The device has been been characterized in a number of applications and found to be a compatible replacement for the DP8344A. Differences between the DP8344A and DP8344B are noted by shading of the text on the pages of this data sheet. For more information, refer to Section 6.6.

Note: In this document [XXX] denotes a control or status bit in a register, {YYY} denotes a register.

Table of Contents

Table of Contents (Continued)

List of Illustrations

List of Illustrations (Continued)

List of Illustrations (Continued)

List of Tables

List of Tables (Continued)

Note: To match Timing table number with appropriate Timing illustration, Tables 5-1 and 5-2 are purposely omitted.

1.0 Communications Processor Introduction

The increased demand for computer connectivity has driven National Semiconductor to develop the next generation of special purpose microprocessors. The DP8344B is the first example of a ''Communications Processor'' for the IBM environment. It integrates a very fast, full function microprocessor with highly specialized transceiver circuitry. The combination of speed, power, and features allows the designer to easily implement a state-of-the-art communications interface. Typical applications for a communications processor are terminal emulation boards for PCs, stand-alone terminals, printer interfaces, and cluster controllers.

The transceiver is designed to simplify the handling of specific communication protocols. This feature makes it possible to quickly develop interfaces and software with little concern for the ''housekeeping'' details of the protocol being used.

1.1 COMMUNICATIONS PROTOCOLS

A communication protocol is a set of rules which defines the physical, electrical, and software specifications required to successfully transfer data between two systems.

The physical specification includes the network architecture, as well as the type of connecting medium, the connectors used, and the maximum distance between connections. Networks may be configured in ''loops,'' ''stars,'' or ''daisy chains,'' and they often use standard coaxial or twisted-pair cable.

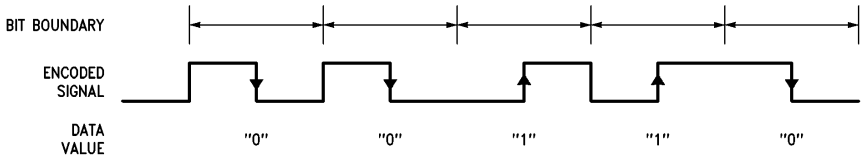
The electrical specification includes the polarity and amplitude of the signal, the frequency (bit rate), and encoding technique. One common method of encoding is called ''biphase'' or ''Manchester II.'' This technique combines the clock and data information into one transmission by encoding data as a ''mid-bit'' transition. *Figure 1-1* shows how the data transition is related to the bit boundary in a typical transmission. The polarity of the ''mid-bit'' transition encodes the data value, other transitions lie on bit boundaries. Bit boundaries are not always indicated by transitions, so techniques employing start sequences and sync bits are used with bi-phase transmissions to ensure proper frame alignment and synchronization.

The software specification covers the use of start sequences and sync bits, as well as defining the message format. Parity bits may be used to ensure data integrity. The message format is the ''language'' that is used to exchange information across the connecting medium. It defines command and control words, response times, and expected responses.

The DP8344B Bi-phase Communications Processor supports both the IBM 3270 and 5250 communication protocols, as well as IBM 3299 and a general purpose 8-bit protocol. The specialized transceiver is combined with a microprocessor whose instruction set is optimized for use in a communications environment. This makes the DP8344 a powerful single-chip solution to a wide range of communication applications.

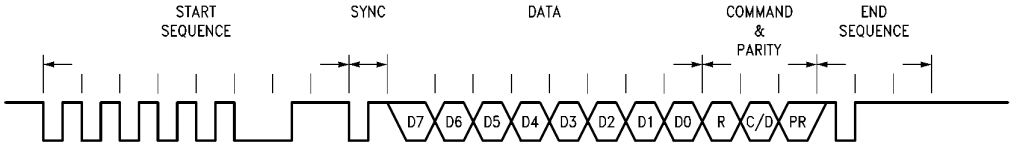
An example of an IBM 3270 message is shown in *Figure 1-2*. The transmission begins with a very specific start sequence and sync pulse for synchronization. This is followed by the data, command, and parity bits. Finally, the end sequence defines the end of the transmission.

The IBM 3270 and 5250 are two widely used protocols. The 3270 protocol was developed for the 370 class mainframe, and it employs coaxial cable in a ''star'' configuration. The 5250 protocol was developed for the System/3x machines, and it uses a ''daisy-chain'' of twin-ax cable. A good overview of both of these environments may be found in the ''Multi-Protocol Adapter System User Guide'' from National Semiconductor, and in the Transceiver section of this document.



TL/F/9336–B7

FIGURE 1-1. Biphase Encoding



TL/F/9336–B8

FIGURE 1-2. IBM 3270 Message Format

1.0 Communications Processor Introduction (Continued)

1.2 INTERNAL ARCHITECTURE INTRODUCTION

The DP8344B Biphase Communications Processor (BCP) is divided into three major functional blocks: the Transceiver, the Central Processing Unit (CPU), and the Remote Interface and Arbitration System, RIAS. *Figure 1-3* shows how these blocks are related to each other and to other system components.

The transceiver consists of an asynchronous transmitter and receiver which can communicate across a serial data path. The transmitter takes parallel data from the CPU and appends to it the appropriate framing information. The resulting message is shifted out and is available as a serial data stream on two output pins. The receiver shifts in serial messages, strips off the framing information, and makes the data available in parallel form to the CPU. The framing information supplied by the BCP provides the proper message format for several popular communication protocols. These include IBM 3270, 3299, and 5250, as well as a general purpose 8-bit mode.

The transceiver clock may be derived from the internal oscillator, either directly or through internal divide-down circuitry. There is also an input for an external transceiver clock, thus allowing complete flexibility in the choice of data rates.

The receiver input can come from three possible sources. There is a built-in differential amplifier which is suitable for most line interfaces, a single-ended digital input for use with an external comparator, and an internal loopback path for self testing. Refer to the Transceiver section for a detailed description of all transmitter and receiver functions, and to the application note on coax interfaces for the proper use of the differential amplifier.

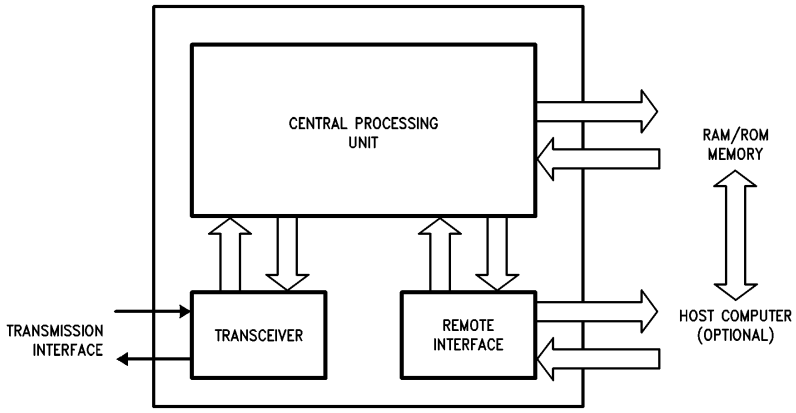
The CPU is a general purpose, 8-bit microprocessor capable of 20 MHz operation. It has a reduced instruction set which is optimized for transceiver and data handling performance. It also has a full function arithmetic/logic unit (ALU) which performs addition, subtraction, Boolean operations, rotations and shifts. Separate instruction and data memory systems are supported, each with 16-bit address buses, for a total of 64k address space in each.

There are 44 internal registers accessible to the CPU. These include special configuration and control registers for the transceiver and processor, four 16-bit indices to data memory, and 20 8-bit general purpose registers. There is also a 16-bit timer and a 16-byte deep LIFO data stack which are accessible in the register address space. For more detailed information, see the specific sections on the Register set, the Timer, and the ALU.

The BCP can operate independently or with another processor as the host system. If such a system is required, communication with the BCP is possible by sharing data memory. The Remote Interface controls bus arbitration and access to data memory, as well as program up-loading and execution. For example, it is possible for a host system to load the BCP's instruction memory and begin program execution, then pass data back and forth through data memory accesses. The section on the Remote Interface and Arbitration System provides all of the necessary timing and control information to implement an interface between a BCP and a remote system.

As shown in *Figure 1-4,* the BCP uses two entirely separate memory systems, one for program storage and the other for data storage. This type of memory arrangement is referred to as Harvard architecture. Each system has 16 address lines, for a maximum of 64k words in each, and its own set of data lines. The instruction (program) memory is two bytes (16 bits) wide, and the data memory is one byte (8 bits) wide.

In order to reduce the number of pins required for these signals, the address and data lines for data memory are multiplexed together. This requires an external latch and the Address Latch Enable signal (ALE) for de-multiplexing.



TL/F/9336–B9

FIGURE 1-3. Simplified Block Diagram

1.0 Communications Processor Introduction (Continued)

Simultaneous access to both data and program memory, and instruction pipelining greatly enhance the speed performance of the BCP, making it well suited for real-time processing. The pipeline allows the next instruction to be retrieved from program memory while the current instruction is being executed.

1.3 TIMING INTRODUCTION

The timing of all CPU operations, instruction execution and memory access is related to the CPU clock. This clock is usually generated by a crystal and the internal oscillator, with optional divide by two circuitry. The period of the resulting CPU clock is referred to as a T-state; for example, a 20 MHz CPU clock yields a 50 ns T-state. Most CPU functions, such as arithmetic and logical operations, shifts and rotates, and register moves, require only two T-states. Branching instructions and data memory accesses require three to four T-states.

Each memory system has a separate, programmable number of wait states to allow the use of slower memory devices. Instruction memory wait states are inserted into all instructions, as shown in *Figure 1-5,* thus they affect the overall speed of program execution. Instruction memory wait states can also apply when the Remote Interface is loading a program into instruction memory. Data memory wait states are only inserted into data memory access instructions, hence there is less degradation in overall program execution. Refer to the Timing section for detailed examples of all BCP instruction and data memory timing.

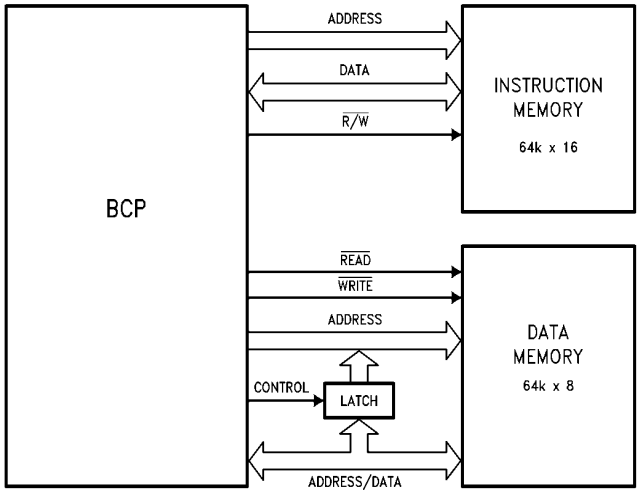


FIGURE 1-4. Memory Configuration

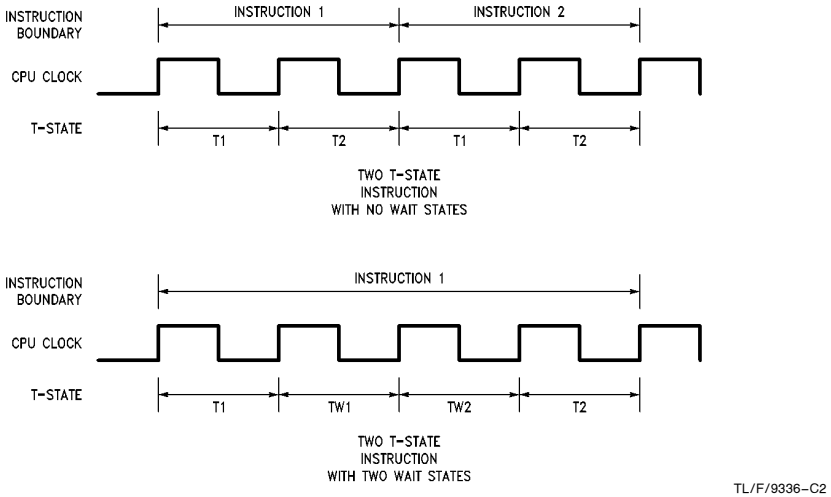


FIGURE 1-5. Effect of Memory Wait States on Timing

1.0 Communications Processor Introduction (Continued)

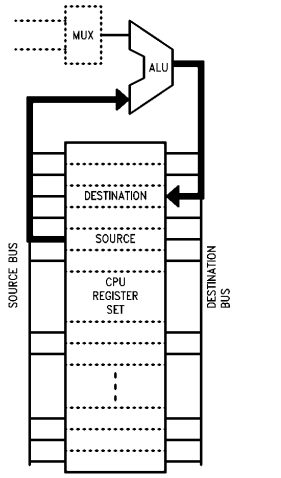
1.4 DATA FLOW

The CPU registers are all dual port, that is, they have separate input and output paths. This arrangement allows a single register to function as both a source and a destination within the same instruction.

Figures 1-6a through *1-6f* show the internal data flow path for the BCP. The CPU registers are a central element to this path. When a register functions as an output, its contents are placed on the Source bus. When a register is an input, data from the Destination bus is written into that register.

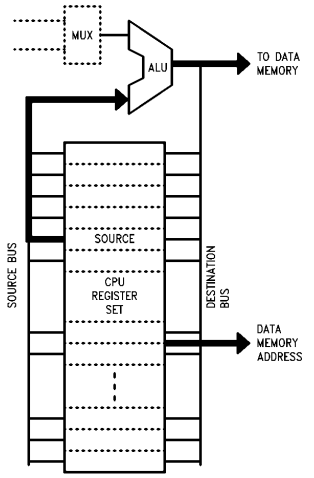
The other key element in the data path is the ALU. This unit does all of the arithmetic and data manipulation operations, but it also has bus multiplexing capabilities. Both the Data Memory bus and a portion of the Instruction Memory bus are routed to this unit and serve as alternative sources of data. Since the data flow is always through this unit, most data moves may include arithmetic manipulations with no penalty in execution time.

Figure 1-6a shows the data path for all arithmetic instructions and register to register moves. The source register contents are placed on the Source bus, routed through the



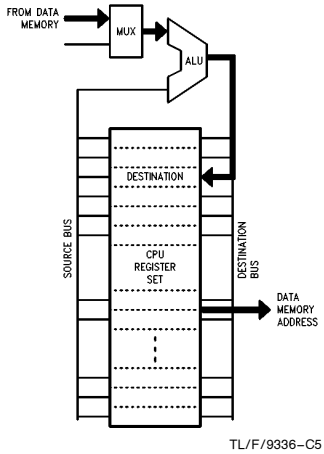
TL/F/9336–C3

FIGURE 1-6a. Register to Register



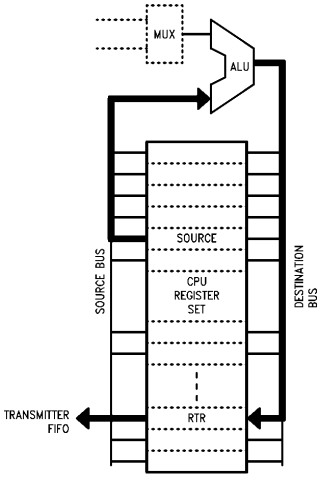
TL/F/9336–C4

FIGURE 1-6b. Data Memory WRITE



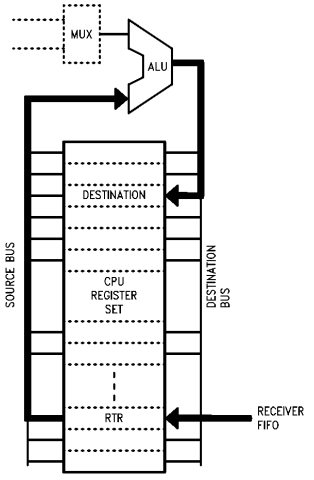
TL/F/9336–C5

FIGURE 1-6c. Data Memory READ



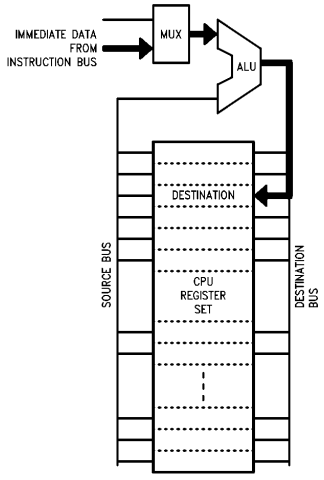
TL/F/9336–C6

FIGURE 1-6d. WRITE to Transmitter



TL/F/9336–C7

FIGURE 1-6e. READ from Receiver



TL/F/9336–C8

FIGURE 1-6f. Load Immediate Data

ALU/MUX, and then placed on the destination bus. This data is then stored into the appropriate destination register.

Figures 1-6b and *1-6c* show the data path for data memory accesses. For a WRITE operation, the source register contents follow the same path through the ALU/MUX, but the Destination bus is routed to output pins and on to data memory. For a READ operation, incoming data is routed onto the Destination bus by the ALU/MUX, and then stored in a register. The address for all data memory accesses is provided by one of four 16-bit index registers which can operate in a variety of automatic increment and decrement modes.

Transfer of the data byte between the CPU and the Transceiver is accomplished through a register location. This register, {RTR}, appears as a normal CPU register, but writing to it automatically transfers data to the transmitter FIFO, and reading from it retrieves data from the receiver FIFO. These paths are illustrated in *Figures 1-6d* and *1-6e.*

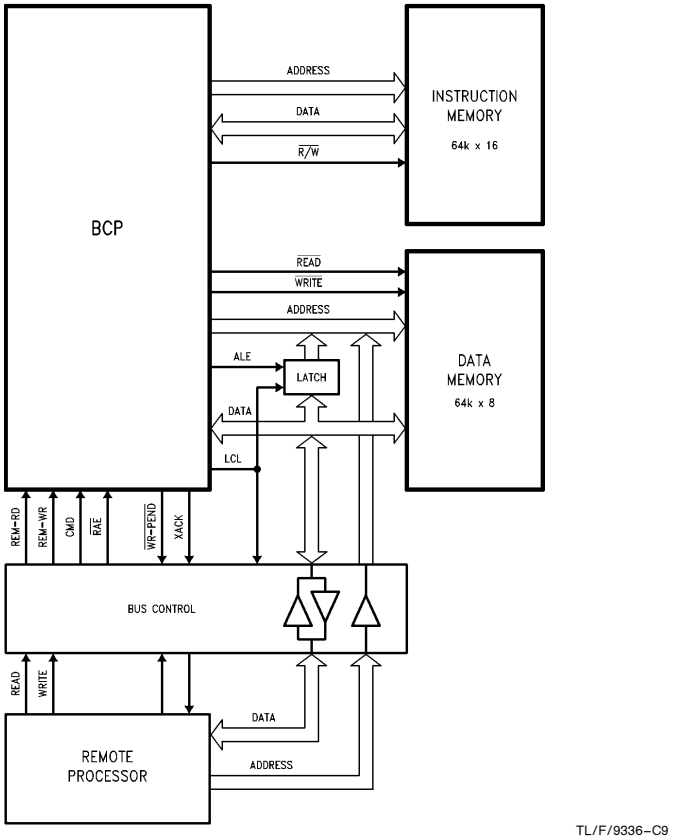
It is also possible to load immediate data into a CPU register. This data is supplied by the program and is usually a constant such as a pointer or character. As shown in *Figure 1-6f,* a portion of the Instruction bus is routed through the ALU/MUX for this purpose.

1.5 REMOTE INTERFACE AND ARBITRATION SYSTEM INTRODUCTION

The BCP is designed to serve as a complete, stand alone communications interface. Alternately, it can be interfaced with another processor by means of the Remote Interface and Arbitration System. Communication between the BCP and the remote processor is possible by sharing data memory. Harvard architecture allows the remote system to access any BCP data memory location while the BCP continues to fetch and execute instructions, thereby minimizing performance degradation.

Figure 1-7 shows a simplified remote processor interface. This includes tri-state buffers on the address and data buses of the BCP's Data Memory, and all of the control and handshaking signals required to communicate between the BCP and the host system.

There is an 8-bit control register, Remote Interface Control {RIC}, accessible only to the remote system, which is used to control a variety of features, including the types of memory accesses, interface speeds, single step program execution, CPU start/stop, instruction memory loads, and so forth. Detailed information on all interface options is provided in the section on Remote Interface and Arbitration System, and in the related Reference section.



TL/F/9336–C9

FIGURE 1-7. Basic Remote Interface

2.0 CPU Description

The CPU is a general purpose, 8-bit microprocessor capable of 20 MHz operation. It contains a large register set for standard CPU operations and control of the transceiver. The reduced instruction set is optimized for the communications environment. The following sections are an architectural and functional description of the DP8344B CPU.

2.1 CPU ARCHITECTURAL DESCRIPTION

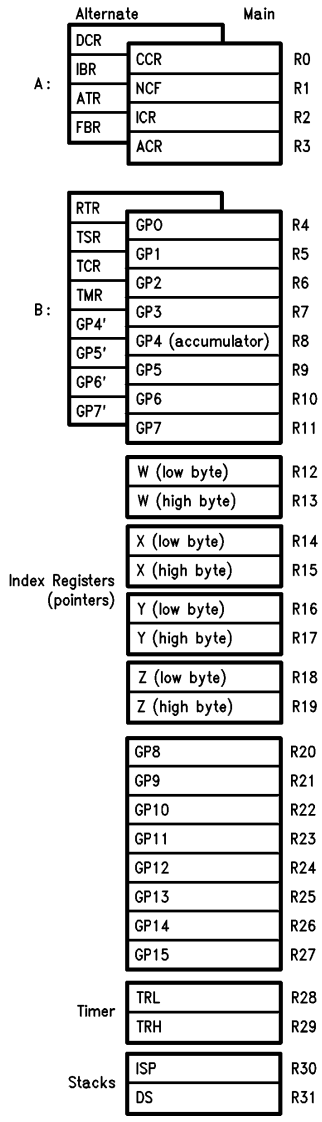
2.1.1 Register Set

This section describes the BCP's internal CPU registers. It is a general overview of the register structure and the functions mapped into the CPU register space. It is not a detailed or exhaustive description of every bit. For such a description, please refer to Section 6.2, Register Set Reference. Also, the Remote Interface Configuration register, {RIC}, is not accessible to the BCP (being accessible only by the remote system) and is described in Section 6.3, Remote Interface Reference.

The register set of the BCP provides for a compliment of both special function and general purpose registers. The special function registers provide access to on-chip peripherals (transceiver, timer, interrupt control, etc.) while the general purpose registers maximize CPU throughput by minimizing accesses to external data memory. The CPU can address a total of 44 8-bit registers, providing access to:

- 20 general purpose registers
- 8 configuration and control registers
- 4 transceiver access registers
- 2 8-bit accumulators
- 4 16-bit pointers
- 16-bit timer
- 16 byte data stack
- address and data stack pointers

The CPU addresses internal registers with a 5-bit field, addressing 32 locations generically named R0 through R31. The first twelve locations (R0–R11) are further organized by function as two groups of banked registers (A and B) as shown in *Figure 2-1*. Each group contains both a main and an alternate bank. Only one bank is active for group A and one for bank B and thus accessible during program execution. Switching between the banks is performed by the exchange instruction EXX which selects whether Main A or Alternate A occupies R0–R3 and whether Main B or Alternate B occupies R4–R11.



TL/F/9336–32

FIGURE 2-1. Register Map

2.0 CPU Description (Continued)

Registers in the R0–R11 address space are allocated in a manner that minimizes the need to switch banks:

Main A: CPU control and transceiver status

Alternate A: CPU and transceiver configuration

Main B: 8 general purpose

Alternate B: 4 transceiver access, 4 general purpose

Most of the BCP's instructions with register operand(s) can access all 32 register locations. Only instructions with an immediate operand are limited to the first sixteen register locations (R0–R15). These instructions, however, still have access to all registers required for transceiver operation, CPU status and control registers, 12 general purpose registers, and two of the index registers.

The general purpose registers are used for the majority of BCP operations. There are 8 general purpose registers in Main Bank B (R4–R11), 4 in Alternate Bank B (R8–R11), and 8 more (R20–R27) that are always accessible but are outside the limited register range. Since these registers are internal to the BCP, they can be accessed without data memory wait states, speeding up processing time. The index registers may also be used as general purpose registers if required.

For those instructions that require two operands, an accumulator (R8, one in each bank) serves as the second operand. The result of such an operation is stored back in the accumulator only if it is specified as the destination, thus allowing three operand operations such as R5 + R8 ⟶ R20. See Section 2.1.3 Instruction Set for further explanation.

Most registers have a predetermined state following a reset to the BCP. Refer to Section 6.2, Register Set Reference for a detailed summary.

2.1.1.1 Banked Registers

The CPU register set was designed to optimize CPU performance in an environment which supports multiple tasks. Generally the most important and time critical of these tasks will be maintaining the serial link (servicing the transceiver section) which often requires real time processing of commands and data. Therefore, all transceiver functions have been mapped into special function registers which the CPU can access quickly and easily. Switching between this task and other tasks has been facilitated by dedicating a register bank (Alternate B) to transceiver functions. Alternate Bank B provides access to all transceiver status, control, and data, in addition to four general purpose registers for protocol related storage. Main Bank B contains eight general purpose registers for use by other tasks. Having general purpose registers in both B banks allows for quick context switching and also helps eliminate some of the overhead of saving general purpose registers. The main objective of this banked register structure is to expedite servicing of the transceiver as a background (interrupt driven) task allowing the CPU to efficiently interleave that function with other background and foreground operations.

To facilitate using the transceiver in a polled fashion (instead of using interrupts), many of the status flags necessary to handshake with the transceiver are built into the conditional jump instructions, with others available in the Main A bank (normally active) so that Alternate Bank B does not have to be switched in to poll the transceiver. Timer and BIRQ tasks may also be run using polling techniques to Main A bank.

In general, the registers have been arranged within the banks so as to minimize the need to switch banks. The power-up state is Alternate bank A, Alternate bank B allowing access to configuration registers. Again, the banks switch by using the EXX instruction which explicitly specifies which bank is active (Main or Alternate) for each register group (A and B). The EXX instruction allows selecting any of four possible bank settings with a single two T-state instruction. This instruction also has the option of enabling or disabling the maskable interrupts.

The contents of the special function registers can be divided into several groups for general discussion—timing/control, interrupt control, the transceiver, the condition codes, the index registers, the timer, the stacks, and remote interface.

2.1.1.2 Timing/Control Registers

The BCP provides a means to configure its external timing through setting bits in the Device Control Register, {DCR}, and the Auxiliary Control Register, {ACR}. One of the first configuration registers to be initialized on power-up/reset is {DCR} which defines the hardware environment in which the BCP is functioning. Specifically, {DCR} controls the clock select logic for both the CPU and transceiver, in addition to the number of wait states to be used for instruction and data memory accesses.

The BCP allows either one clock source operation for the CPU and the transceiver from the on-chip oscillator, or an independent clock source can run the transceiver from the eXternal Transceiver CLocK input, X-TCLK. The Transceiver Clock Select bits, [TCS1,0], select the clock source for the transceiver which is either the on-chip Oscillator CLocK, OCLK, or X-TCLK. Options for selecting divisions of the on-chip oscillator frequency are also provided (see the description of {DCR} in Section 6.2, Register Set Reference. The CPU Clock Select bit, [CCS], allows the CPU to run at the OCLK frequency or at half that speed. The clock output at the pin CLK-OUT, however, is never divided and always reflects the crystal frequency OCLK. The frequency selected for the transceiver (referred to as TCLK) should always be eight times the desired serial data rate. The frequency selected for the CPU defines the length of each T-state (e.g., 20 MHz implies 50 ns T-states).

There are two independent fields for defining wait states, one for instruction memory access (n_{IW}) and one for data memory access (n_{DW}). These fields specify to the BCP how many wait states to insert to meet the access time requirements of both memory systems. The Instruction memory Wait-state select bits, [IW1,0], and the Data memory Wait-state select bits, [DW2–0], control the number of inserted wait states for instruction and data memory, respectively.

After a reset, the maximum number of wait states are set in {DCR}, $n_{IW} = 3$ T-states and $n_{DW} = 7$ T-states. Wait-states are discussed in more detail in Section 2.2.2, Timing. For a complete discussion on choosing your memory and determining the number of wait states required, please refer to the application note *Choosing Your RAM for the Biphase Communication Processor.*

2.0 CPU Description (Continued)

Another control bit in the {ACR} register is the Clock Out Disable bit, [COD]. When [COD] is asserted, the buffered clock output at pin CLK-OUT is tri-stated.

2.1.1.3 Interrupt Control Registers

The configuration bank (Alternate Bank A) includes an Interrupt Base Register, {IBR}, which defines the high byte of all interrupt and trap vector addresses. Thus, the interrupt vector table can be located in any 256 byte page of the 64k range of instruction addresses. The interrupt base is normally initialized once on reset before interrupts are enabled or any traps are executed. Since $\overline{\text{NMI}}$ is nonmaskable and may occur before {IBR} is initialized, the power-up/reset value of {IBR} (00h) should be used to accommodate $\overline{\text{NMI}}$ during initialization. In other words, if $\overline{\text{NMI}}$ is used in the system, the absolute address 001Ch (the $\overline{\text{NMI}}$ vector) should contain a jump to an $\overline{\text{NMI}}$ service routine.

The Interrupt Control Register, {ICR}, provides individual masks [IM4–0] for each of the maskable interrupts. The Global Interrupt Enable bit, [GIE], located in {ACR} works in conjunction with these individual masks to control each of the maskable interrupts.

The external pin called $\overline{\text{BIRQ}}$ is a Bidirectional Interrupt ReQuest. $\overline{\text{BIRQ}}$ is defined as an input or an output by the Bidirectional Interrupt Control bit, [BIC], in {ACR}. [IM3] functions as BIRQ's interrupt mask if $\overline{\text{BIRQ}}$ is an input as defines by [BIC]. When [BIC] defines BIRQ as an output, [IM3] controls the output state of $\overline{\text{BIRQ}}$.

Section 2.2.3, Interrupts provides a further description of these registers.

2.1.1.4 Timer Registers

The timer block interfaces with the CPU via two registers, TimeR Low byte, {TRL}, and TimeR High byte, {TRH}, which form the input/output ports to the timer. Writing to {TRL} and {TRH} stores the low and high byte, respectively, of a 16-bit time-out value into two holding registers. The word stored in the holding registers is the value that the timer will be loaded with via [TLD]. Also, the timer will automatically reload this word upon timing out. Reading {TRL} and {TRH} provides access to the count down status of the timer.

Control of timer operation is maintained via three bits in the Auxiliary Control Register {ACR}. Timer STart [TST], bit 7 in {ACR}, is the start/stop control bit. Writing a one to [TST] allows the timer to start counting down from its current value. When low, the timer stops and the timer interrupt is cleared. Timer Load [TLD], bit 6 in {ACR}, is the load control of the timer. After writing the desired values into {TRL} and {TRH}, writing a one to [TLD] will load the 16-bit word in the holding registers into the timer and initialize the timer clock to zero in preparation to start counting. Upon completing the load operation, [TLD] is automatically cleared. Timer Clock Selection [TCS], bit 5 in {ACR}, determines the clock frequency of the timer count down. When low, the timer divides the CPU clock by sixteen to form the clock for the down counter. When [TCS] is high, the timer divides the CPU clock by two. The input clock to the timer is the CPU clock and should not be confused with the oscillator clock, OCLK. The rate of the CPU clock will be either equal to OCLK or one-half of OCLK depending on the value of bit 7 in the Device Control Register, {DCR}.

When the timer reaches a count of zero, the timer interrupt is generated, the Time Out flag, [TO], (bit 7 in the Condition Code Register {CCR}), goes high, and the timer reloads the 16-bit word stored in the holding registers to recycle through a count down. The timer interrupt and [TO] can be cleared by either writing a one to [TO] in {CCR} or stopping the timer by writing a zero to [TST] in {ACR}. Refer to Section 2.1.2, Timer for more information on the timer operation.

2.1.1.5 Transceiver Registers

Two registers in the Alternate A bank initialize transceiver functions. The Auxiliary Transceiver Register, {ATR}, specifies a station address used by the address recognition logic within the transceiver when using the non-promiscuous 5250 and 8-bit protocol modes. In 5250 modes, {ATR} also defines how long the TX-ACT pin stays asserted after the end of a transmitted message. The Fill Bit Register, {FBR}, specifies the number of optional fill bits inserted between frames in a multiframe 5250 message.

{ICR} contains the Receiver Interrupt Select bits, [RIS1,0]. These bits determine the receiver interrupt source selection. The source may be either Receiver FIFO Full, Data Available, or Receiver Active.

The Receive/Transmit Register, {RTR}, is the input/output port to both the transmitter and receiver FIFO's. It appears to the BCP CPU like any other register. The {RTR} register provides the least significant eight bits of data in both received and transmitted messages.

The Transceiver Mode Register, {TMR}, contains bits used to set the configuration of the transceiver. As long as the Transceiver RESet bit, [TRES], is high, the transceiver remains in reset. Internal LOOP-back operation of the transceiver can be selected by asserting [LOOP]. The RePeat ENable bit, [RPEN], allows the receiver to be active at the same time as the transmitter. When the Receiver INvert bit, [RIN], is set, all data sent to the receiver is inverted. The Transmitter INvert bit, [TIN], is analogous to [RIN] except it is for the transmitter. The protocol that the transceiver is using is selected with the Protocol Select bits, [PS2–0].

The Transceiver Command Register, {TCR}, controls the workings of the transmitter. To generate 5.5 line quiesce pulses at the start of a transmission rather than 5, the Advance Transmitter Active bit, [ATA], must be set high. Parity is automatically generated on a transmission and the Odd Word Parity bit, [OWP], determines whether that parity is even or odd. Bits 2–0 of {TCR} make up part of the Transmitter FIFO [TF10–8] along with {RTR}. Whenever a write is made to {RTR}, [TF10–8] are automatically pushed on the FIFO with the 8 bits written to {RTR}.

Other bits in {TCR} control the operation of the on-chip receiver. The number of line quiesce bits the receiver must detect to recognize a valid message is determined by the Receive Line Quiesce bit, [RLQ]. The BCP has its own internal analog comparator, but an off-chip one may be connected to DATA-IN. The receiver source is determined by the Select Line Receiver bit, [SLR]. To view transceiver errors in the Error Code Register, {ECR}, the Select Error Codes, [SEC], bit in {TCR} must be set high. When [SEC] is high, Alternate Bank B R4 is remapped from {RTR} to {ECR} so that {ECR} can be read.

2.0 CPU Description (Continued)

Just as [TF10–8] bits get pushed onto the transmitter FIFO when a write to {RTR} occurs, the Receiver FIFO bits, [RF10–8], in the Transceiver Status Register, {TSR}, reflect the state of the top word of the receive FIFO. {TSR} also contains flags that show Transmit FIFO Full, [TFF], Transmitter Active, [TA], Receiver Error, [RE], Receiver Active, [RA], and Data AVailable, [DAV]. These flags may be polled to determine the state of the transceiver. For instance, during a Receiver Active interrupt, the BCP can query the [DAV] bit to determine whether data is ready in the receiver FIFO yet.

The Error Code Register, {ECR}, contains flags for receiver errors. As previously stated, the [SEC] bit in {TRC} must be set high to read this register. Reading {ECR} or resetting the transceiver with [TRES] will clear all the errors that are present. The receiver OVerFlow flag, [OVF], is set when the receiver attempts to add another word to the FIFO when it is full. If internally checked parity and parity transmitted with a 3270 message conflict, then the PARity error bit, [PAR], is set high. The Invalid Ending Sequence bit, [IES], is set when the ending sequence in a 3270, 3299, or 8-bit message is incorrect. When the expected mid-bit transition in the Manchester waveform does not occur, a Loss of Mid-Bit Transition occurs ([LMBT]). Finally, if the transmitter is activated while the receiver is active, the Receiver DISabled while active flag, [RDIS], will be set unless [RPEN] is asserted.

The second register in Main A bank is called the Network Command Flag register, {NCF}, and contains information about the transceiver which is useful for polling the transceiver (during other tasks for example) to see if it needs servicing. These flags include bits to indicate Transmit FIFO Empty [TFE], Receive FIFO Full [RFF], Line Active [LA], and a Line Turn Around [LTA]. [LTA] indicates that a message has been received without error and a valid ending sequence has occurred. These flags facilitate polling of the transceiver section when transceiver interrupts are not used. Also included in this register is a bit called [DEME] (Data Error/Message End). In 3270/3299 modes, this bit indicates a mismatch between received and locally generated byte parity. In 5250 modes, [DEME] decodes an end of message indicator (111 in the address field). Three other bits: Received Auto Response [RAR], Acknowledge [ACK] and Poll [POLL] are decoded from a received message (at the output of the receive FIFO) and are valid only in 3270/3299 modes where response time is critical.

Section 3.0 Transceiver provides comprehensive coverage of this on-chip peripheral.

2.1.1.6 Condition Codes/Remote Handshaking Register

The ALU condition codes are available in the Condition Code Register {CCR}. The [Z] bit is set when a zero result is generated by an arithmetic, logical, or shift instruction. Similarly, [N] indicates the Negative result of the same operations. An oVerflow condition from an arithmetic instruction sets the [V] bit in {CCR}. The Carry bit [C] indicates a carry or borrow result from an arithmetic instruction. See Section 2.2.2, ALU for more information.

The Condition Code Register, {CCR}, also contains [BIRQ], a status bit which reflects the logic level of the bidirectional interrupt input pin BIRQ. Hence, this pin can be used as a general purpose input/output port as well as a bidirectional interrupt request as defined by bits in {ACR} and {ICR}. If a remote CPU is present and shares data memory (dual port memory) with the BCP, handshaking can be accomplished by using the two status bits in {CCR} called [RR] and [RW], which indicate Remote Read and Remote Write accesses, respectively.

In {ACR}, a lock bit, [LOR], is available to lock out all host accesses. When this bit is set, all host accesses are disabled. Locking out remote accesses is often done during interrupts to ensure quick response times.

The Remote Interface Configuration register, {RIC}, is not available to the BCP internally. The Remote Interface Reference section provides further detail on {RIC} and interfacing a remote processor.

2.1.1.7 Index Registers

Four index registers called IW, IX, IY, and IZ provide 16-bit addressing for both data memory and instruction memory. Each of these index registers is actually a pair of 8-bit registers which are individually addressable just like any other CPU register. They occupy register addresses R12 through R19. Thus, the first two pointers IW and IX (comprising R12–R15) can be accessed with immediate mode instructions (which can access only R0 to R15). Refer to Section 2.1.3.2, Addressing Modes to see how the index registers are formed from R12–R19.

Accessing data memory requires the use of one of the four index registers. All such instructions allow you to specify which pointer is to be used, except the immediate-relative moves: MOVE rs,[IZ+n] and MOVE [IZ+n],rd. These instructions always use the IZ pointer. Register indirect operations have options to alter the value of the index register; the options include pre-increment, post-increment, and post-decrement. These options facilitate block moves, searches, etc. Refer to Section 2.1.3, Instruction Set for more information about data moves.

Since the BCP's ALU is 8 bits wide, all code that manipulates the index registers must act on them eight bits at a time.

The index registers can also be used in register indirect jumps (LJMP [Ir]), useful in implementing relocatable code. Any one of the index registers can be specified to provide the 16-bit instruction address for the indirect jump.

2.1.1.8 Stack Registers

The last two register addresses (R30,R31) are dedicated to provide access to the two on-chip stacks—the data stack and the address stack. The data stack is 8 bits wide and 16 words deep. It is a Last In First Out (LIFO) type and provides high speed storage for variables, pointers, etc. The address stack is 23 bits wide and 12 words deep, providing twelve levels of nesting of subroutines and interrupts. It is also a LIFO structure and stores processor status as well as return addresses from CALL instructions, TRAP instructions, and interrupts. The seven bits of processor status consist of the four ALU flags, ([C], [N], [V], and [Z]), the current bank setting (two bits), and [GIE].

Stack pointers for both the on-chip stacks are provided in R30, the Internal Stack Pointer register, {ISP}. The lower four bits are the pointer for the data stack and the upper four bits are the pointer for the address stack. Both internal stacks are circular. For example if 16 bytes are written to

2.0 CPU Description (Continued)

the data stack, the next byte pushed will overwrite the first. {ISP} can be read and written to like any other register, but after a write, the BCP must execute one instruction before reading the stack whose pointer was modified.

The Data Stack register, {DS}, is the input/output port for the data stack. This port is accessed like any other register, but a write to it will "push" a byte onto the stack and a read from it will "pop" a byte from the stack. The data stack pointer is updated when a read or write of {DS} occurs.

Information bits in the instruction address stack are not mapped into the CPU's register space and, therefore, are not directly accessible. A remote system running a monitor program can access this information by forcing the BCP to single-step through a return instruction and then reading the program counter. Since the stack pointers are writeable, the remote system can access any location (return address) in the address stack to trace program flow and then restore the stack pointer to its original position.

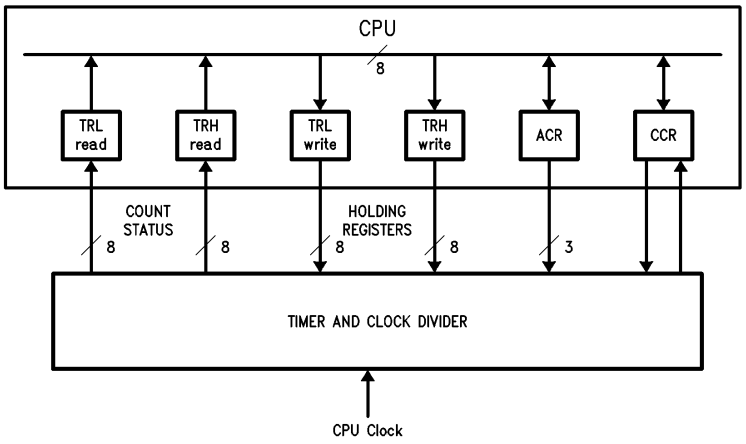
2.1.2 Timer

The BCP has an internal 16-bit timer that can be used in a variety of ways. The timer counts independently of the CPU, eliminating the waste of valuable processor bandwidth. The timer can be used in a polled or interrupt driven configuration for user software flexibility.

The timer interfaces with the CPU via two registers, TimeR Low byte, {TRL}, and TimeR High byte, {TRH}, which form the input/output ports to the timer. Writing to {TRL} and {TRH} stores the low and high byte, respectively, of a 16-bit time-out value into two holding registers. The word stored in the holding registers is the value that the timer will be loaded with via [TLD]. Also, the timer will automatically reload this word upon timing out. Reading {TRL} and {TRH} provides access to the count down status of the timer.

Control of timer operation is maintained via three bits in the Auxiliary Control Register {ACR}. Timer STart [TST], bit 7 in {ACR}, is the start/stop control bit. Writing a one to [TST] allows the timer to start counting down from its current value. When low, the timer stops and the timer interrupt is cleared. Timer Load [TLD], bit 6 in {ACR}, is the load control of the timer. After writing the desired values into {TRL} and {TRH}, writing a one to [TLD] will load the 16-bit word in the holding registers into the timer and initialize the timer clock to zero in preparation to start counting. Upon completing the load operation, [TLD] is automatically cleared. Timer Clock Selection [TCS], bit 5 in {ACR}, determines the clock frequency of the timer count down. When low, the timer divides the CPU clock by sixteen to form the clock for the down counter. When [TCS] is high, the timer divides the CPU clock by two. The input clock to the timer is the CPU clock and should not be confused with the oscillator clock, OCLK. The rate of the CPU clock will be either equal to OCLK or one-half of OCLK depending on the value of bit 7 in the Device Control Register, {DCR}.

When the timer reaches a count of zero, the timer interrupt is generated, the Time Out flag, [TO], (bit 7 in the Condition Code Register {CCR}), goes high, and the timer reloads the 16-bit word stored in the holding registers to recycle through a count down. The timer interrupt and [TO] can be cleared by either writing a one to [TO] in {CCR} or stopping the timer by writing a zero to [TST] in {ACR}. A block diagram of the timer is shown in *Figure 2-2*.



TL/F/9336–D1

FIGURE 2-2. Timer Block Diagram

2.0 CPU Description (Continued)

2.1.2.1 Timer Operation

After the desired 16-bit time-out value is written into {TRL} and {TRH}, the start, load, and clock selection can be achieved in a single write to {ACR}. A restriction exists on changing the timer clock frequency in that [TCS] should not be changed while the timer is running (i.e., [TST] is high). After a write to {ACR} to load and start the timer, the timer begins counting down at the selected frequency from the value in {TRL} and {TRH}. Upon reaching a count of zero, the timer interrupt is generated and, the timer reloads the current word from {TRL} and {TRH} to cycle through a countdown again. The timing waveforms shown in *Figure 2-3* show a write to {ACR} that loads, starts, selects the CPU clock rate/2 for the countdown rate, and asserts the Global Interrupt Enable [GIE]. Prior to the write to {ACR}, {TRL} and {TRH} were loaded with 00h and 01h respectively, the timer interrupt was unmasked in the Interrupt Control Register {ICR} by clearing bit 4, and zero instruction wait states were selected in {DCR}. Since the write to {ACR} asserted [GIE], the timer interrupt is enabled and the CPU will vector to the timer interrupt service routine address when the timer reaches a count of zero. The timer interrupt is the lowest priority interrupt and is latched and maintained until it is cleared in software. (See CPU Interrupts section). For very long time intervals, time-outs can be accumulated under software control by writing a one to [TO] in {CCR} allowing the timer to recycle its count down with no other intervention. For time-outs attainable with one count down, stopping the timer will clear the interrupt and [TO]. When the timer interrupt is enabled, the call to the interrupt service routine occurs at different instruction boundaries depending on when the timer interrupt occurs in the instruction cycle. If the timer times out prior to T2, where T2 is the last T-state of an instruction cycle, the call to the interrupt service routine will occur in the next instruction. When the time-out occurs in T2, the call to the interrupt service routine will not occur in the next instruction. It occurs in the second instruction following T2.

The count status of the timer can be monitored by reading {TRL} and/or {TRH}. When the registers are read, the output of the timer, not the value in the input holding registers, is presented to the ALU. Some applications might require monitoring the count status of the timer while it is counting down. Since the timer can time-out between reads of {TRL} and {TRH}, the software should take this fact into consideration. To read back what was written to {TRL} and {TRH}, the timer must first be loaded via [TLD] without starting the timer followed by a one instruction delay before reading {TRL} and {TRH} to allow the output registers to be updated from the load operation.

To determine the time-out delay for a given value in {TRL} and {TRH} other than 0000h, the following equation can be used:

$$TD = (\text{value in } \{TRH\}\{TRL\}) * T * k$$

where:

k = 2 when [TCS] = 1 or 16 when [TCS] = 0

T = The period of the CPU clock

TD = The amount of time delay after the end of the instruction that asserts [TST] in {ACR}

When the value of 0000h is loaded in the timer, the maximum time-out is obtained and is calculated as follows:

$$TD = 65536 * T * k$$

With the CPU running full speed with an 18.8 MHz crystal, the maximum single loop time delay attainable would be 55.6 ms ([TCS] = 0). The minimum time delay with the same constraints is 106 ns ([TCS] = 1). For accumulating time-out intervals, the total time delay is simply the number of loops accumulated multiplied by the calculated time delay. The equations above do not account for any overhead for processing the timer interrupt. The added overhead of processing the interrupt may need to be included for precision timing.

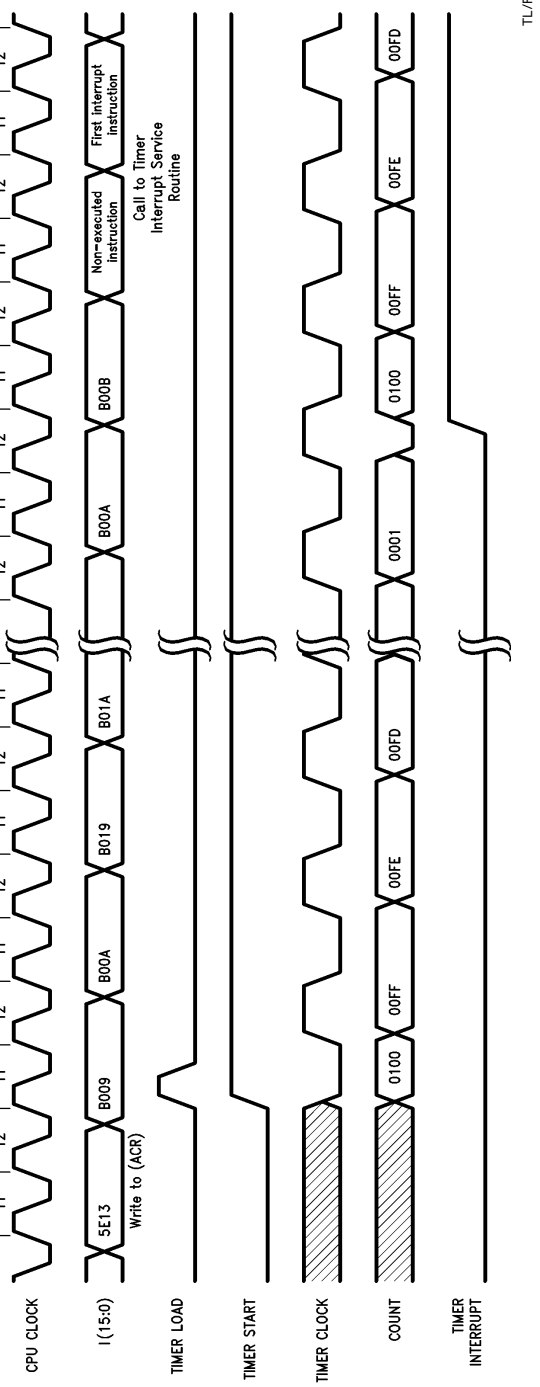


FIGURE 2-3. Timer Interrupt Diagram

2.0 CPU Description (Continued)

2.1.3 Instruction Set

The followng paragraphs introduce the BCP's architecture by discussing addressing modes and briefly discussing the Instruction Set. For detailed explanations and examples of each instruction, refer to the Instruction Set Reference Section.

2.1.3.1 Harvard Architecture Implications

The BCP utilizes a true Harvard Architecture, where the instruction and data memory are organized into two independent memory banks, each with their own address and data buses. Both the Instruction Address Bus and the Instruction Bus are 16 bits wide with the Instruction Address Bus addressing memory by words. (A word of memory is 16 bits long; i.e., 1 word = 2 bytes.) Most of the instructions are one word long. The exceptions are two words long, containing a word of instruction followed by a word of immediate data. The combination of word sized instructions and a word based instruction address bus eliminates the typical instruction alignment problems faced by many CPU's.

The Data Address Bus is 16 bits wide (with the low order 8 bits multiplexed on the Data Bus), and the Data Bus is 8 bits wide (i.e., one byte wide). The Data Address Bus addresses memory by bytes. Most of the BCP's instructions operate on byte-sized operands.

Note that although both instruction addresses and data addresses are 16 bits long, these addresses are for two different buses and, therefore, have two different numerical meanings, (i.e., byte address or word address.) Each instruction determines whether the meaning of a 16-bit address is that of an instruction word address or a data byte address. Little confusion exists though because only the program flow instructions interpret 16-bit addresses as instruction addresses.

2.1.3.2 Addressing Modes

An addressing mode is the mechanism by which an instruction accesses its operand(s). The BCP's architecture supports five basic addressing modes: register, immediate, indexed, immediate-relative, and register-relative. The first two allow instructions to execute the fastest because they require no memory access beyond instruction fetch. The remaining three addressing modes point to data or instruction memory. Typical of a RISC processor, most of the instructions only support the first three addressing modes, with one of the operands always limited to the register addressing mode.

Register Addressing Modes

There are two terminologies for the register addressing modes: Register and Limited Register. Instructions that allow Register operands can access all the registers in the CPU. Note that only 32 of the 44 CPU registers are available at any given point in time because the lower 12 register locations (R0–R11) access one of two switchable register banks each. (See Section 2.1.1.1, Banked Registers for more information on the CPU register banks.) Instructions that allow the Limited Register operands can access just the first 28 registers of the CPU. Again, note that only 16 of these 28 registers are available at any given point in time. Table 2-1 shows the notations used for the Register and Limited Register operands. Some instructions also imply the use of certain registers, for example the accumulators. This is noted in the discussions of those instructions.

Immediate Addressing Modes

The two types of the immediate addressing modes available are: Immediate numbers and Absolute numbers. Immediate numbers are 8 bits of data, (one data byte), that code directly into the instruction word. Immediate numbers may represent data, data address displacements, or relative instruction addresses. Absolute numbers are 16-bit numbers. They code into the second word of two word instructions and they represent absolute instruction addresses. Table 2-2 shows the notations used for both of these addressing modes.

TABLE 2-1. Register Addressing Mode Notations

Notation	Type of Register Operand	Registers Allowed
Rs	Source Register	R0–R31
Rd	Destination Register	R0–R31
Rsd	Register is both a Source & Destination	R0–R31
rs	Limited Source Register	R0–R15
rd	Limited Destination Register	R0–R15
rsd	Limited Register is both a Source & Destination	R0–R15

TABLE 2-2. Immediate Addressing Mode Notations

Notation	Type of Immediate Operand	Size
n	Immediate Number	8 Bits
nn	Absolute Number	16 Bits

Indexed Addressing Modes

Indexed operands involve one of four possible CPU register pairs referred to as the index registers. *Figure 2-4* illustrates how the index registers map into the CPU Register Set. Note that the index registers are 16 bits wide.

Index registers allow for indirect memory addressing and usually contain data memory addresses, although, the LJMP instruction can use index registers to hold instruction memory addresses. Most of the instructions that allow memory indirect addressing, (i.e. the use of index registers), also allow pre-incrementing, post-incrementing, or post-decrementing of the index register contents during instruction execution, if desired. Table 2-3 lists the notations used for the index register modes.

The index registers are set to zero when the BCP's $\overline{\text{RESET}}$ pin is asserted.

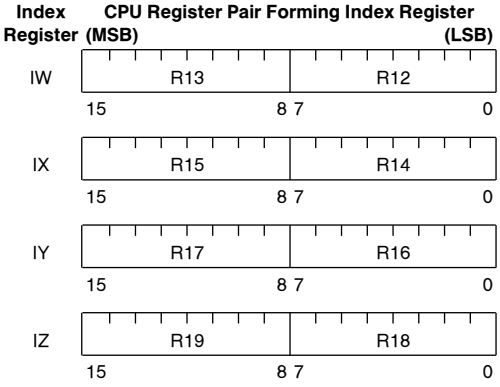


FIGURE 2-4. Index Register Map

Immediate-Relative and Register-Relative Address Modes

The Immediate-Relative mode adds an unsigned 8-bit immediate number to the index register IZ forming a data byte address. The Register-Relative mode adds the unsigned 8-bit value in the current accumulator, A, to any one of the index registers forming a data byte address. Both of these indirect memory addressing modes are available only on the MOVE instruction. Table 2-4 shows the notation used for these two addressing modes.

2.1.3.3 Instruction Set Overview

The BCP's RISC instruction set contains seven categories of instructions: Data Movement, Integer Arithmetic, Logic, Shift-Rotate, Comparison, Program Flow, and Miscellaneous.

Data Movement Instructions

The MOVE instruction is responsible for all the data transfer operations that the BCP can perform. Moving one byte at a time, five different types of transfer are allowed: register to register, data memory to register, register to data memory, instruction memory to register, and instruction memory to data memory. Table 2-5 lists all the variations of the MOVE instruction.

TABLE 2-3. Index Register Addressing Mode Notations

Notation	Meaning
[Ir]	Index Register, Contents Not Changed
[Ir−]	Index Register, Contents Post-Decremented
[Ir+]	Index Register, Contents Post-Incremented
[+Ir]	Index Register, Contents Pre-Incremented
[mIr]	General Notation Indicating that Any of the Above Modes Is Allowed

Note: [ ] denotes indirect memory addressing and is part of the instruction syntax.

TABLE 2-4. Relative Index Register Mode Notations

Notation	Type of Action Performed to Calculate a Data Memory Address
[IZ + n]	IZ + Immediate Number (unsigned) ⟶ Data Memory Address
[Ir + A]	Index Register + Current Accumulator (unsigned) ⟶ Data Memory Address

Note: [ ] denotes indirect memory addressing and is part of the instruction syntax.

TABLE 2-5. Data Movement Instructions

Syntax	Instruction Operation	Addressing Modes
MOVE Rs, Rd	register ⟶ register	Register, Register
MOVE Rs, [mIr]	register ⟶ data memory	Register, Indexed
MOVE [mIr], Rd	data memory ⟶ register	Indexed, Register
MOVE Rs, [Ir + A]	register ⟶ data memory	Register, Register-Relative
MOVE [Ir + A], Rd	data memory ⟶ register	Register-Relative, Register
MOVE rs, [IZ + n]	register ⟶ data memory	Limited Register, Immediate-Relative
MOVE [IZ + n], rd	data memory ⟶ register	Immediate-Relative, Limited Register
MOVE n, rd	instruction memory ⟶ register	Immediate, Limited Register
MOVE n, [Ir]	instruction memory ⟶ data memory	Immediate, Indexed

2.0 CPU Description (Continued)

Integer Arithmetic Instructions

The integer arithmetic instructions operate on 8-bit signed (two's complement) binary numbers. Two arithmetic functions are supported: Add and Subtract. Three versions of the Add and Subtract instructions exist: operand ± accumulator, operand ± accumulator ± carry, and immediate operand ± operand. The first two versions support both the register and indexed addressing modes for the destination operand. These two versions also allow the specification of a separate register or data address for the destination operand so that the sources may retain their integrity; (i.e., true three-operand instructions). Note that the currently active "B" register bank selects which accumulator is used in these instructions. The third version, immediate operand ± operand, only supports the register addressing mode for the destination operand with the register as both a source and the destination. Table 2-6 lists the integer arithmetic instructions along with their variations.

Logic Instructions

The logic instructions operate on 8-bit binary data. A full set of logic functions is supported by the BCP: AND, OR, eXclusive OR, and Complement. All the logic functions except complement allow either an immediate operand or the currently active accumulator as an implied operand. Complement only allows one register operand which is both the source and destination. The other logic instructions include the following addressing modes: register, indexed, and immediate. As with the integer arithmetic instructions, the integrity of the sources may be maintained by specifying a destination register which is different from the source. Table 2-7 lists all the logic instructions.

TABLE 2-6. Integer Arithmetic Instructions

Syntax		Instruction Operation	Addressing Modes
ADD	n, rsd	register + n ⟶ register	Immediate, Limited Register
ADDA	Rs, Rd	Rs + accumulator ⟶ Rd	Register, Register
ADDA	Rs, [mlr]	Rs + accumulator ⟶ data memory	Register, Indexed
ADCA	Rs, Rd	Rs + accumulator + carry ⟶ Rd	Register, Register
ADCA	Rs, [mlr]	Rs + accumulator + carry ⟶ data memory	Register, Indexed
SUB	n, rsd	register − n ⟶ register	Immediate, Limited Register
SUBA	Rs, Rd	Rs − accumulator ⟶ Rd	Register, Register
SUBA	Rs, [mlr]	Rs − accumulator ⟶ data memory	Register, Indexed
SBCA	Rs, Rd	Rs − accumulator − carry ⟶ Rd	Register, Register
SBCA	Rs, [mlr]	Rs − accumulator − carry ⟶ data memory	Register, Indexed

TABLE 2-7. Logic Instructions

Syntax		Instruction Operation	Addressing Modes
AND	n, rsd	register & n ⟶ register	Immediate, Limited Register
ANDA	Rs, Rd	Rs & accumulator ⟶ Rd	Register, Register
ANDA	Rs, [mlr]	Rs & accumulator ⟶ data memory	Register, Indexed
OR	n, rsd	register | n ⟶ register	Immediate, Limited Register
ORA	Rs, Rd	Rs | accumulator ⟶ Rd	Register, Register
ORA	Rs, [mlr]	Rs | accumulator ⟶ data memory	Register, Indexed
XOR	n, rsd	register ⊕ n ⟶ register	Immediate, Limited Register
XORA	Rs, Rd	Rs ⊕ accumulator ⟶ Rd	Register, Register
XORA	Rs, [mlr]	Rs ⊕ accumulator ⟶ data memory	Register, Indexed
CPL	Rsd	$\overline{\text{register}}$ ⟶ register	Register

Note: & = logical AND operation
| = logical OR operation
⊕ = logical exclusive OR operation
$\bar{r}$ = one's complement

2.0 CPU Description (Continued)

Shift and Rotate Instructions

The shift and rotate instructions operate on any of the 8-bit CPU registers. The BCP supports shift left, shift right, and rotate operations. Table 2-8 lists the shift and rotate instructions.

Comparison Instructions

The BCP utilizes two comparison instructions. The CMP instruction performs a two's complement subtraction between a register and immediate data. The BIT instruction tests selected bits in a register by ANDing it with immediate data. Neither instruction stores its results, only the ALU flags are affected. Table 2-9 lists both of the comparison instructions.

Program Flow Instructions

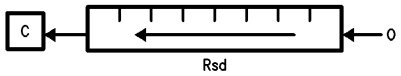
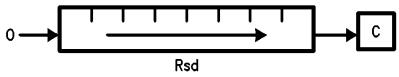
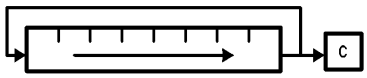
The BCP has a wide array of program flow instructions: unconditional jumps, calls and returns; conditional jumps, calls, and returns; relative or absolute instruction addressing on jumps and calls; a specialized register field decoding jump; and software interrupt capabilities. These instructions redirect program flow by changing the Program Counter.

The unconditional jump instructions support both relative instruction addressing, the (JuMP instruction), and absolute instruction addressing, (the Long JuMP instruction), using the following addressing modes: Immediate, Register, Absolute, and Indexed. Table 2-10 lists the unconditional jump instructions and their variations.

The conditional jump instructions support both relative instruction addressing and absolute instruction addressing using the Immediate and Absolute addressing modes. The conditional relative jump instruction tests flags in the Condition Code Register, {CCR}, and the Transceiver Status Register, {TSR}. Two possible syntaxes are supported for the conditional relative jump instruction; see Table 2-11.

Table 2-12 lists the various flags "f" that the conditional JMP instruction can test and Table 2-13 lists the various conditions "**cc**" that the J**cc** instruction can test for. Keep in

TABLE 2-8. Shift and Rotate Instructions

Syntax	Instruction Operation	Addressing Mode
SHL Rsd,b		Register
SHR Rsd,b		Register
ROT Rsd,b		Register

Note: "b" = the number of bit shifts/rotates to perform.

TABLE 2-9. Comparison Instructions

Syntax	Instruction Operation	Addressing Mode
CMP rs, n	register − n	Limited Register
BIT rs, n	register & n	Limited Register

Note: & = logical AND operation

TABLE 2-10. Unconditional Jump Instructions

Syntax	Instruction Operation	Operand Range	Addressing Mode
JMP n	PC + n (sign extended) ⟶ PC	−128, +127	Immediate
JMP Rs	PC + Rs (sign extended) ⟶ PC	−128, +127	Register
LJMP nn	nn ⟶ PC	0, 64k	Absolute
LJMP [Ir]	Ir ⟶ PC	0, 64k	Indexed

Note: PC = Program Counter; contents initially points to instruction following jump.

2.0 CPU Description (Continued)

mind that the J**cc** instruction is just an optional syntax for the conditional JMP instruction.

The example in *Figure 2-5* demonstrates two possible ways to code the conditional relative jump instruction when testing for a false [Z] flag in {CCR}. In the example, assume that the symbol ''Z'' equals ''000'' binary, that the symbol ''NS'' equals ''0'' binary, and that the symbol ''SKIP.IT'' points to the desired instruction with which to begin execution if [Z] is false.

On the other hand, the conditional absolute jump instruction, LJMP, can test any bit in any currently active CPU register. Table 2-14 shows the conditional long jump instruction syntax.

```
JMP   Z,NS,SKIP.IT   ;If [Z]=0 goto SKIP.IT
      -or-
JNZ   SKIP.IT        ;If [Z]=0 goto SKIP.IT
```

FIGURE 2-5. Coding Examples of Equivalent Conditional Jump Instructions

TABLE 2-11. Conditional Relative Jump Instruction

Syntax	Instruction Operation	Operand Range	Addressing Mode
JMP f,s,n	If the flag ''f'' is in the state ''s'' then PC + n (sign extended) ⟶ PC	−128, +127	Immediate
J**cc** n	If the condition ''**cc**'' is met then PC + n (sign extended) ⟶ PC	−128, +127	Immediate

Note: PC = Program Counter; contents initially points to instruction following jump.

TABLE 2-12. ''f'' Flags

''f''(Binary)	Flag	Flag Name	Register Containing Flag
000	Z	Zero	{CCR}
001	C	Carry	{CCR}
010	V	Overflow	{CCR}
011	N	Negative	{CCR}
100	RA	Receiver Active	{TSR}
101	RE	Receiver Error	{TSR}
110	DAV	Data Available	{TSR}
111	TFF	Transmitter FIFO Full	{TSR}

TABLE 2-13. ''cc'' Conditions Tested

''cc'' Field	Condition Tested for	Flag ''f''s Condition
Z	Zero	[Z] = 1
NZ	Not Zero	[Z] = 0
EQ	Equal	[Z] = 1
NEQ	Not Equal	[Z] = 0
C	Carry	[C] = 1
NC	No Carry	[C] = 0
V	Overflow	[V] = 1
NV	No Overflow	[V] = 0
N	Negative	[N] = 1
P	Positive	[N] = 0
RA	Receiver Active	[RA] = 1
NRA	Not Receiver Active	[RA] = 0
RE	Receiver Error	[RE] = 1
NRE	No Receiver Error	[RE] = 0
DA	Data Available	[DAV] = 1
NDA	No Data Available	[DAV] = 0
TFF	Transmitter FIFO FULL	[TFF] = 1
NTFF	Transmitter FIFO Not Full	[TFF] = 0

TABLE 2-14. Conditional Absolute Jump Instruction

Syntax	Instruction Operation	Operand Range	Addressing Mode
LJMP Rs,p,s,nn	If the bit of register ''Rs'' in position ''p'' is in the state ''s'' then nn ⟶ PC	0, 64k	Register, Absolute

Note: PC = Program Counter

2.0 CPU Description (Continued)

The BCP also has a specialized relative jump instruction called relative Jump with Rotate and Mask on source register, JRMK. This instruction facilitates the decoding of register fields often involved in communications processing. JRMK does this by rotating and masking a copy of its register operand to form a signed program counter displacement which usually points into a jump table. Table 2-15 shows the syntax and operation of the JRMK instruction.

JRMK's masking, (setting to zero), the least significant bit of the displacement allows the construction of a jump table using either one or two word instructions; for instance, a table of JMP and/or LJMP instructions, respectively. The example in *Figure 2-6* demonstrates the JRMK instruction decoding the address frame of the 3299 Terminal Multiplexer protocol which is located in the Receive/Transmit Register, {RTR[4–2]}.

The BCP has two unconditional call instructions; CALL, which supports relative instruction addressing and LCALL, (Long CALL), which supports absolute instruction addressing. These instructions push the following information onto the CPU's internal Address Stack: the address of the next instruction; the status of the Global Interrupt Enable flag, [GIE]; the status of the ALU flags [Z], [C], [N], and [V]; and the status of which register banks are currently active. Table 2-16 lists the two unconditional call instructions. Note that the Address Stack is only twelve positions deep; therefore, the BCP allows twelve levels of nested subroutine invocations, (this includes both interrupts and calls).

TABLE 2-15. JRMK Instruction

Syntax	Instruction Operation	Displacement Range	Addressing Mode
JRMK Rs, b, m	(a) Rotate a copy of register "Rs" "b" bits to the right. (b) Mask the most significant "m" bits and the least significant bit of the above result. (c) PC + resulting displacement (sign extended) ⟶ PC.	−128, +126	Register

Note: PC = Program Counter; contents initially points to instruction following jump.

Example Code

```
JRMK  RTR,1,4    ;decode terminal address
LJMP  ADDR.0     ;jump to device handler #0
LJMP  ADDR.1     ;jump to device handler #1
. . .
LJMP  ADDR.7     ;jump to device handler #7
```

Instruction Execution	JRMK Displacement Register Contents							
(a) Copy {RTR} into JRMK's displacement register:	x	x	x	A2	A1	A0	y	y
(b) Rotate displacement register 1 bit to the right:	y	x	x	x	A2	A1	A0	y
(c) AND result with "00001110" binary mask:	0	0	0	0	A2	A1	A0	0
(d) Sign extend resulting displacement and add it to the program counter, (PC). If the bits A2 A1 A0 equal "0 0 1" binary then + 2 is added to the Program Counter; (i.e., PC + 2 ⟶ PC).	0	0	0	0	0	0	1	0
(e) Execute the instruction pointed to by the PC, which in this example is: LJMP ADDR.1								

FIGURE 2-6. JRMK Instruction Example

TABLE 2-16. Unconditional Call Instructions

Syntax	Instruction Operation	Operand Range	Addressing Mode
CALL n	PC & [GIE] & ALU flags & reg. bank selection ⟶ Address Stack PC + n (sign extended) ⟶ PC	−128, +127	Immediate
LCALL nn	PC & [GIE] & ALU flags & reg. bank selection ⟶ Address Stack nn ⟶ PC	0, 64k	Absolute

Note: PC = Program Counter; contents initially points to instruction following call.
[GIE] = Global Interrupt Enable bit.
& = concatenation operator, combines operands together forming one long operand.

2.0 CPU Description (Continued)

The BCP has one conditional call instruction capable of testing any bit in any currently active CPU register. This call only supports absolute instruction addressing. Table 2-17 shows the conditional call instruction syntax and operation.

The return instruction complements the above call instructions. Two versions of the return instruction exist, the uncondtional return and the conditional return. When the unconditional return instruction is executed, it pops the last address on the CPU's Address Stack into the program counter and it can optionally affect the [GIE] bit, the ALU flags, and the register bank selection. Table 2-18 shows the syntax and operation of the unconditional return instruction.

The conditional return instruction functions the same as the unconditional return instruction if a desired condition is met. As with the conditional jump instruction, the conditional return instruction has two possible syntaxes. Table 2-19 lists the syntax for the conditional return. The "f" flags and the "**cc**" conditions for the return instruction are the same as for the conditional jump instruction, therefore refer to Table 2-12 and Table 2-13 for the listing of "**f**" and "**cc**", respectively.

TABLE 2-17. Conditional Call Instruction

Syntax	Instruction Operation	Operand Range	Addressing Mode
LCALL Rs, p, s, nn	If the bit of register "Rs" in position "p" is in the state "s" then PC & [GIE] & ALU flags & reg. bank selection ⟶ Address Stack nn ⟶ PC End if	0, 64k	Register, Absolute

Note: PC = Program Counter; contents initially points to instruction following call.
[GIE] = Global Interrupt Enable bit
& = concatenation operator, combines operands together forming one long operand.

TABLE 2-18. Unconditional Return Instruction

Syntax	Instruction Operation

RET {g {, rf}}	Case "g" of
	0: leave [GIE] unaffected, (default)
	1: restore [GIE] from Address Stack
	2: set [GIE]
	3: clear [GIE]
	End case
	If "rf" = 1 then
	restore ALU flags from Address Stack
	restore register bank selection from Address Stack
	Else (the default)
	leave the ALU flags and register bank selections unchanged
	End if
	Address Stack ⟶ PC

Note: PC = Program Counter
[GIE] = Global Interrupt Enable bit
{} = surrounds optional operands that are not part of the instruction syntax.
Optional operands may either be specified or omitted.

TABLE 2-19. Conditional Return Instruction

Syntax	Instruction Operand
RETF f, s {, {g}, {, rf}}	If the flag "f" is in the state "s" then perform a RET {g {, rf}}
R**cc** {g {, rf}}	If the condition "**cc**" is met then perform a RET {g {,rf}}

Note: See Table XVIII for an explanation of "RET {g {, rf}}"
{} = surrounds optional operands that are not part of the instruction syntax.
Optional operands may either be specified or omitted.

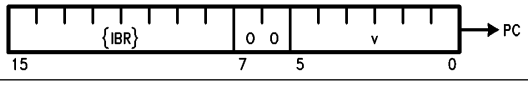
2.0 CPU Description (Continued)

In addition to the above jump, call and return program flow instructions, the BCP is capable of generating software interrupts via the TRAP instruction. This instruction generates a call to any one of 64 possible interrupt table addresses based on its vector number operand. This allows both the simulation of hardware interrupts and the construction of special software interrupts, if desired. The actual interrupt table entry address is determined by concatenating the Interrupt Base Register, {IBR}, to an 8-bit representation of the vector number operand in the TRAP instruction. This instruction may also clear the [GIE] bit, if desired. Table 2-20 shows the syntax and operation of the TRAP instruction.

Miscellaneous Instructions

As stated in the ''CPU Register Set'' section, the BCP has 44 registers with 24 of them arranged into four register banks: Main Bank A, Alternate Bank A, Main Bank B, and Alternate Bank B. The exchange instruction, EXX, selects which register banks are currently available to the CPU, for example either Main Bank A or Alternate Bank A. The deselected register banks retain their current values. The EXX instruction can also alter the state of [GIE], if desired. Table 2-21 shows the EXX instruction syntax and operation.

TABLE 2-20. TRAP Instruction

Syntax	Instruction Operation	Operand Range
TRAP v {, g'}	PC & [GIE] & ALU flags & reg. Bank Selection ⟶ Address Stack If ''g'' = 1 then clear [GIE] Form PC address as shown below:  	0, 63

Note: PC = Program Counter; contents initially points to instruction following call.
[GIE] = Global Interrupt Enable bit
IBR = Interrupt Base Register
& = concatenation operator, combines operands together forming one long operand.
{ } = surrounds optional operands that are not part of the instruction syntax.
Optional operands may either be specified or omitted.

TABLE 2-21. EXX Instruction

Syntax	Instruction Operation
EXX ba, bb {, g}	Case ''ba'' of 0: activate Main Bank A 1: activate Alternate Bank A End case Case ''bb'' of 0: activate Main Bank B 1: activate Alternate Bank B End case Case ''g'' of 0: leave [GIE] unaffected, (default) 1: (reserved) 2: set [GIE] 3: clear [GIE] End case

Note: [GIE] = Global Interrupt Enable bit
{ } = surrounds optional operands that are not part of the instruction syntax.
Optional operands may either be specified or omitted.

2.0 CPU Description (Continued)

2.2 CPU FUNCTIONAL DESCRIPTION

2.2.1 ALU

The BCP provides a full function high speed 8-bit Arithmetic Logic Unit (ALU) with full carry look ahead, signed arithmetic, and overflow decision capabilities. The ALU can perform six arithmetic, nine logic, one rotate and two shift operations on binary data. Full access is provided to all CPU registers as both source and destination operands, and using the indirect addressing mode, results may be placed directly into data memory. All operations which have an internal destination (register addressing) are completed in two (2) T-states. External destination operations (indirect addressing to data memory) complete in three (3) T-states.

Arithmetic operations include addition with or without carry, and subtraction with or without borrow (represented by carry). Subtractions are performed using 2's complement addition to accommodate signed operands. The subtrahend is converted to its 2's complement equivalent by the ALU and then added to the minuend. The result is left in 2's complement form.

The remaining ALU operations include full logic, shift and rotate operations. The logic functions include Complement, AND, OR, Exclusive-OR, Compare and Bit Test. Zero through seven bit right and left shift operations are provided, along with a zero through seven bit right rotate operation. Note that the shift and rotate operations may only be performed on a register, which is both the source and destination. (See the Instruction Set Overview section for detailed descriptions of these operations.)

The BCP ALU provides the programmer with four instruction result status bits for conditional operations. These bits (known as condition code flags) indicate the status (or condition) of the destination byte produced by certain instructions. Not all instructions have an affect on every status flag. (See the Instruction Set Reference section for the specific details on what status flags a given instruction affects.) These flags are held in the Condition Code Register, {CCR}, see *Figure 2-7.*

7	6	5	4	3	2	1	0
TO	RR	RW	BIRQ	N	V	C	Z

where:

N = Negative
C = Carry
V = Overflow
Z = Zero

FIGURE 2-7. Condition Code Register ALU Flags

If an instruction is documented as affecting a given flag, then the flags are set (to 1) or cleared (to 0) under the following conditions:

[N]— The Negative flag is set if the most significant bit (MSB) of the result is one (1), otherwise it is cleared. This flag represents the sign of the result if it is interpreted as a 2's complement number.

[C] — The Carry flag is set if:

a) An addition operation generates a carry, see *Figure 2-8a.*

b) A subtract or compare operation generates a borrow, see *Figure 2-8b.*

c) The last bit shifted out during a shift operation (in either direction) is a one (1), see *Figure 2-9.*

d) The last bit rotated by the rotate operation is a one (1), see *Figure 2-10.*

In all other conditions [C] is cleared.

[V]— Overflow is set whenever the result of an arithmetic or compare operation on signed operands is not representable by the operand size, thereby producing an incorrect result. For example, the addition of the two signed negative numbers in *Figure 2-8a* would set [V] since the correct representation of the result, both sign and magnitude, is not possible in 8 bits. On the other hand, in *Figure 2-8b* and *2-8c* [V] would be cleared because the results are correctly represented in both sign and magnitude. It is important to remember that Overflow is only meaningful in signed arithmetic and that it is the programmer's responsibility to determine if a given operation involves signed or unsigned values.

[Z]— The Zero flag is set only when an operation produces an all bits cleared result (i.e., a zero). In all other conditions [Z] is cleared.

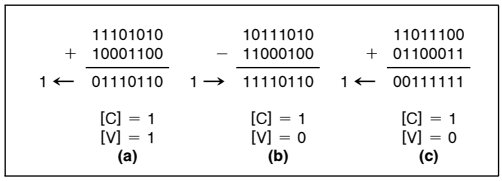
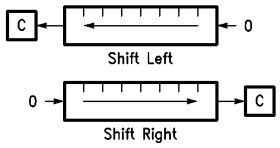
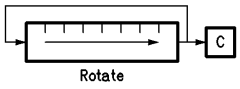


FIGURE 2.8. Carry and Overflow Calculations



TL/F/9336–D3

FIGURE 2-9. Shifts' Effect on Carry



TL/F/9336–D4

FIGURE 2-10. Rotate's Effect on Carry

2.0 CPU Description (Continued)

Several conditions apply to these flags, independent of their operation and the way they are calculated. These conditions are:

1. A flag's previous state is retained when an instruction has no affect on that flag.
2. Direct reading and writing of all ALU flags is possible via the {CCR} register.
3. Currrent flag values are saved onto the address stack during interrupt and call operations, and can be restored to their original values if a return instruction with the restore flags option is executed.
4. Flag status is calculated in parallel with the instruction result, therefore no time penalty is associated with flag operation.

When performing single byte arithmetic (i.e., the values are completely represented in one byte) the Add (ADD,ADDA) and Subtract (SUB,SUBA) instructions should be used, but when performing multi-byte arithmetic the Add with Carry (ADCA) and Subtract with Carry (SBCA) instructions should be used. This is because the carry (in an add operation) or the borrow (in a subtract operation) must be carried forward to the higher order bytes. *Figure 2-11* demonstrates an instruction sequence for a 16-bit add and an instruction sequence for a 16-bit subtract.

Assume the 16-bit variable X is represented by the register pair R4(MSB), R5(LSB), and that the 16-bit variable Y is represented by the register pair R6(MSB), R7(LSB).

To perform the assignment Y = X + Y:

```
MOVE    R7,A    ;GET LSB OF Y
ADDA    R5,R7   ;Y(LSB)=X(LSB)+Y(LSB)
MOVE    R6,A    ;GET MSB OF Y
ADCA    R4,R6   ;Y(MSB)=X(MSB)+Y(MSB)
                 +CARRY
```

To perform the assignment Y = X – Y:

```
MOVE    R7,A    ;GET LSB OF Y
SUBA    R5,R7   ;Y(LSB)=X(LSB)-Y(LSB)
MOVE    R6,A    ;GET MSB OF Y
SBCA    R4,R6   ;Y(MSB)=X(MSB)-Y(MSB)
                 -CARRY
```

FIGURE 2-11. Multi-Byte Arithmetic Instruction Sequences

When using the ALU to perform comparisons, the programmer has two options. If the compare is to a constant value then the CMP instruction can be used, else one of the subtract instructions must be used. When determining the results of any compare, the programmer must keep in mind whether they are comparing signed or unsigned values. Table 2-22 lists the Boolean condition that must be met for unsigned comparisons and Table 2-23 lists the Boolean condition that must be met for signed comparisons.

TABLE 2-22

Unsigned Comparison Results	
Comparison: x − y	**Boolean Condition**
x < y	C
x ≤ y	C | Z
x = y	Z
x ≥ y	$\overline{C}$
x > y	$\overline{C}$ & $\overline{Z}$

Note: & = logical AND
| = logical OR
$\overline{z}$ = one's complement

TABLE 2-23

Signed Comparison Results	
Comparison: x − y	**Boolean Condition**
x < y	(N&$\overline{V}$) | ($\overline{N}$&V)
x ≤ y	Z | (N&$\overline{V}$) | ($\overline{N}$&V)
x = y	Z
X ≥ y	(N&V) | ($\overline{N}$&$\overline{V}$)
x > y	(N&V&$\overline{Z}$) | ($\overline{N}$&$\overline{V}$&$\overline{Z}$)

Note: & = logical AND
| = logical OR
$\overline{z}$ = one's complement

2.2.2 Timing

Timing on the BCP is controlled by an internal oscillator and circuitry that generates the internal timing signals. This circuitry in the CPU is referred to as Timing Control. The internal timing of the CPU is synchronized to an internal clock called the CPU clock, CPU-CLK. A period of CPU-CLK is referred to as a T-state. The clock for the BCP is provided by a crystal connected between X1 and X2 or from a clock source connected to X1. This clock will be referred to as the oscillator clock, OCLK. The frequency of OCLK is divided in half when the CPU clock select bit, [CCS], in the Device Control Register, {DCR}, is set to a one. Either OCLK or OCLK/2 is used by Timing Control to generate CPU-CLK and other synchronous signals used to control the CPU timing.

After the BCP is reset, [CCS] is high and CPU-CLK is generated from OCLK/2. Since the output of the divider that creates OCLK/2 can be high or low after reset, CPU-CLK can also be in a high or low state. Therefore, the exact number of clock cycles to the start of the first instruction cannot be determined. Automatic test equipment can synchronize to the BCP by asserting $\overline{\text{RESET}}$ as shown in *Figure 2-12*. The falling edge of $\overline{\text{RESET}}$ generates a clear signal which causes CPU-CLK to fall. The next rising edge of X1 removes the clear signal from CPU-CLK. The second rising edge of X1 will cause CPU-CLK to rise and the relationship between X1 and CPU-CLK can be determined from this point.

Writing a zero to [CCS] causes CPU-CLK to switch from OCLK/2 to OCLK. The transition from OCLK to OCLK/2 occurs following the end of the instruction that writes to

2.0 CPU Description (Continued)

[CCS] as shown in *Figure 2-13*. The switch occurs on the falling edge of X1 when CPU-CLK is low. CPU-CLK can be changed back to OCLK/2 by writing a one to [CCS]. The point at which CPU-CLK changes depends on whether there has been an odd or even number of T-states since [CCS] was set low. The change would require a maximum of two T-states and a minimum of one T-state following the end of the instruction that writes to [CCS].

The CPU is a RISC processor with a limited number of instructions which execute in a short period of time. The maximum instruction cycle time is four T-states and the minimum is two T-states. Six types of instruction timing are used in the CPU: two T-state, three T-state program control, three T-state data memory access, four T-state read data memory access, four T-state program control, and four T-state two word program control. The first T-state of each instruction is T1 and the last T-state is T2. Intermediate T-states required to complete the instruction are referred to as TX.

The instruction clock output, ICLK, defines the instruction boundaries. ICLK rises at the beginning of each instruction and falls one-half T-state after the next address is generated on the instruction address bus, IA. Thus, ICLK indicates the start of each instruction and when the next instruction address is valid.

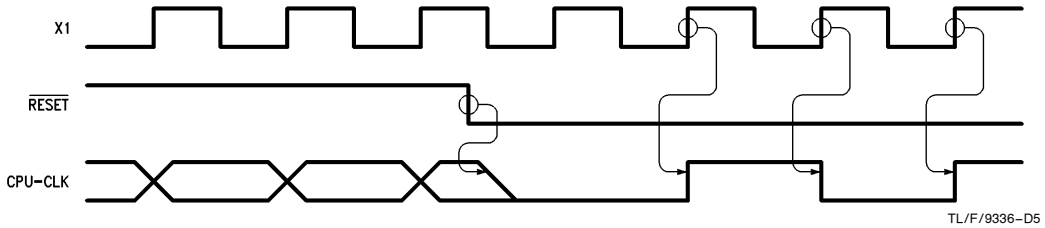


FIGURE 2-12. CPU-CLK Synchronization with X1

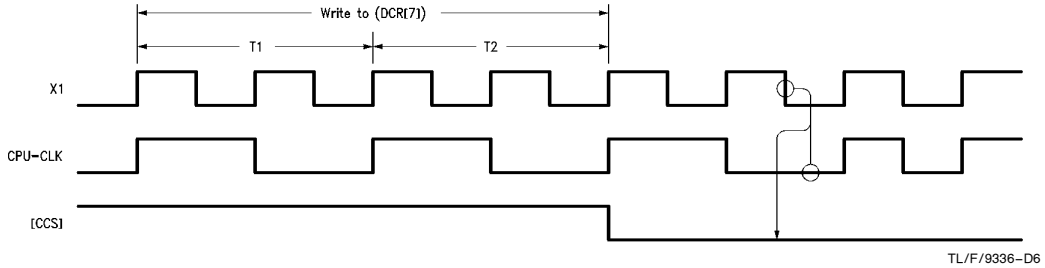


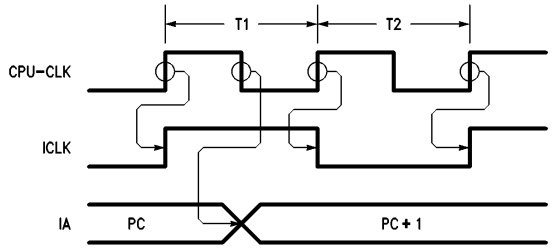
FIGURE 2-13. Changing from OCLK/2 to OCLK

2.0 CPU Description (Continued)

Figure 2-14 shows the relationship between CPU-CLK, ICLK, and IA for a two T-state instruction. The rising edge of CPU-CLK generates ICLK at the start of T1. The next falling edge of CPU-CLK increments the instruction address which appears on IA. ICLK falls one-half T-state later. The instruction completes during T2 which ends with ICLK rising, signifying the beginning of the next instruction.

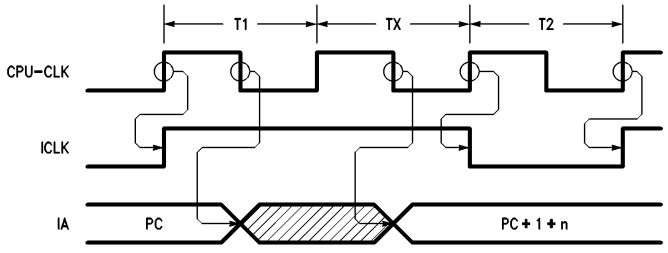
The three T-state program control instruction is similar and is shown in *Figure 2-15*. An additional T-state, TX, is added between T1 and T2. ICLK rises at the beginning of T1 as before but falls at the end of TX. The next instruction address is generated one-half T-state before the end of TX and the instruction ends with T2.

The three T-state data memory access instruction timing is shown in *Figure 2-16*. Again, TX is inserted between T1 and T2. ICLK rises at the beginning of the instruction and falls at the end of T1. The next instruction address appears on IA one-half clock cycle before ICLK falls. The address latch enable output, ALE, rises halfway through T1 and falls halfway through TX. The BCP has a 16-bit data memory address bus and an 8-bit data bus. The data bus is multiplexed with the lower 8 bits of the address bus and ALE is used to latch the lower 8 bits of the address during a data memory access. The upper 8 bits of the address become valid one-half T-state after the beginning of T1 and go invalid one-half T-state after the end of T2. The lower 8 bits of the address become valid on the address-data bus, AD, when ALE rises and goes invalid one-half T-state after ALE falls. *Figure 2-16* shows a write to data memory in which case AD switches from address to data at the beginning of T2. The data is held valid until one-half T-state after the end of T2. The write strobe, $\overline{\text{WRITE}}$, falls at the beginning of T2 and rises at the end of T2. A read of data memory is shown in *Figure 2-17*. The read timing is the same as a write except one-half T-state after ALE falls AD goes into a high impedance state allowing data to enter the BCP from data memory. AD returns to an active state at the end of T2. The read strobe, $\overline{\text{READ}}$, timing is identical to $\overline{\text{WRITE}}$.



TL/F/9336–D7

FIGURE 2-14. Two T-state Instruction



TL/F/9336–D8

FIGURE 2-15. Three T-state Program Control Instruction

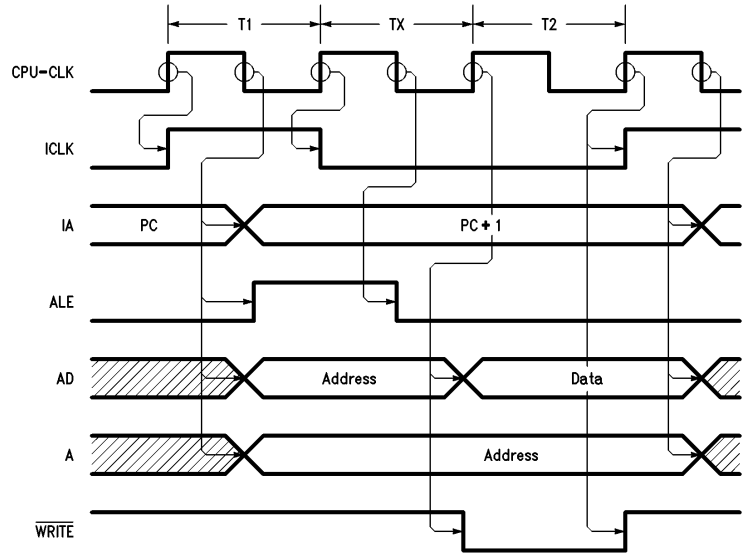


FIGURE 2-16. Three T-state Data Memory Write Instruction

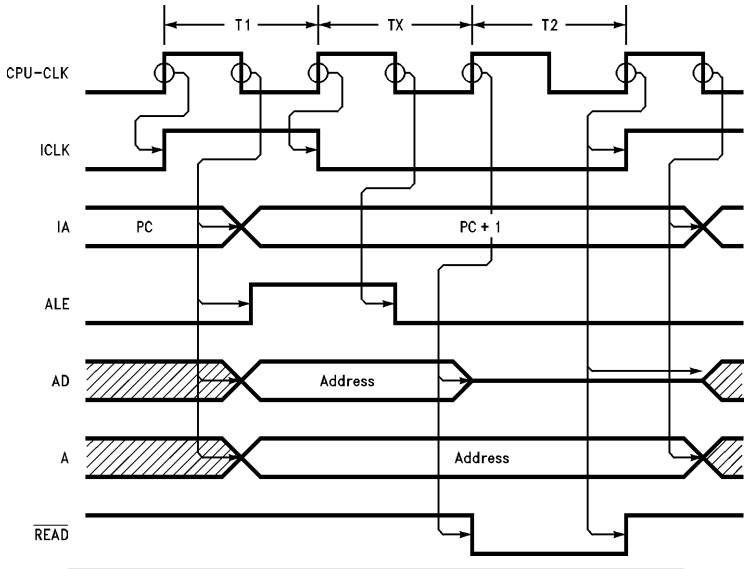


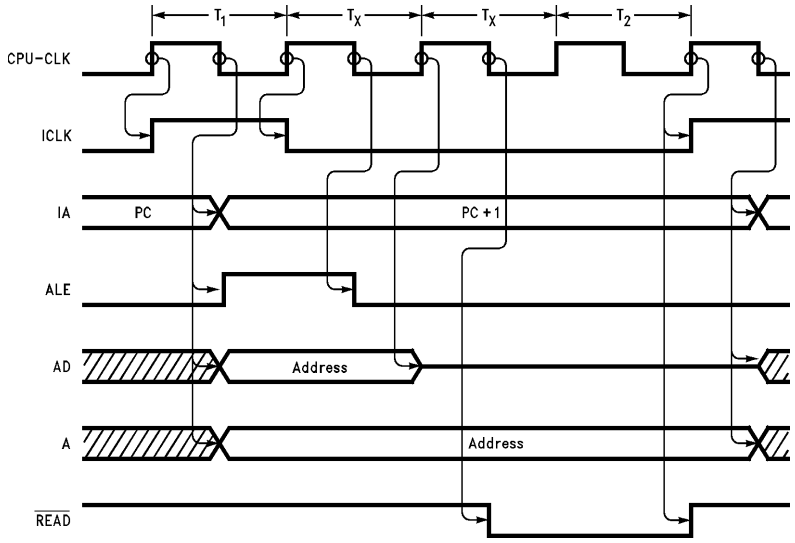
FIGURE 2-17. Three T-state Data Memory Read Instruction [4TR] = 0

2.0 CPU Description (Continued)

When the Four T-state Read mode is selected ([4TR] = 1), a second TX state is inserted before T2 and the timing of the read strobe, $\overline{\text{READ}}$, is changed such that $\overline{\text{READ}}$ falls one-half T-state after the beginning of the second TX. *Figure 2-18* shows a Four T-state Read of data memory. The extra half T-state before $\overline{\text{READ}}$ falls allows more time for the BCP to TRI-STATE the AD lines before the memory circuit begins driving those lines.

The four T-state program control instruction timing is shown in *Figure 2-19*. The instruction has two TX states inserted between T1 and T2. ICLK rises at the beginning of T1 and falls at the end of the second TX. The next instruction address becomes valid halfway through the second TX. The four T-state two word program control instruction timing is the same as two consecutive two T-state instructions and is shown in *Figure 2-20.*

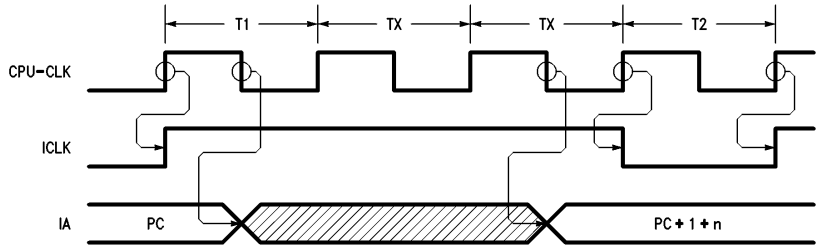
This timing describes the minimum cycle time required by each type of instruction. The BCP can be slowed down by changing the number of wait states selected in the Device Control Register, {DCR}. The BCP can be programmed for up to three instruction memory wait states (instruction wait states) and seven data memory wait states (data wait states). Instruction wait states affect all instruction types while data wait states affect only data memory access instructions. Bits three and four in {DCR} control the number of instruction wait states and bits zero, one and two are used to select the number of data wait states. The relationships between the control bits and the number of wait states selected are shown in Table 2-24 and Table 2-25. The BCP is configured with three instruction wait states and seven data wait states, and [4TR] set to zero after reset. A write to {DCR[4,3]} to change the number of instruction wait states takes effect on the following instruction if that instruction is a three T-state or four T-state program control instruction. For the other instruction types, the new number of instruction wait states will take effect on the instruction fol-



TL/F/9336–H5

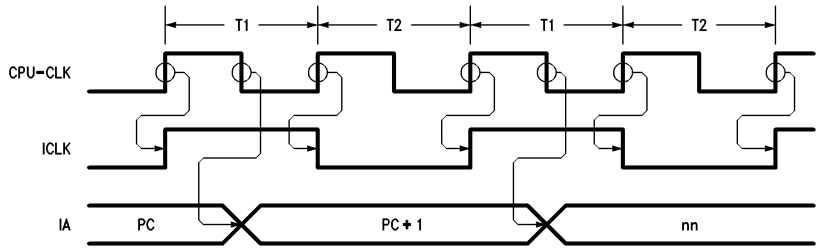
FIGURE 2-18. Four T-state Data Memory Read Instruction [4TR] = 1

2.0 CPU Description (Continued)



TL/F/9336–E2

FIGURE 2-19. Four T-state Program Control Instruction



TL/F/9336–E3

FIGURE 2-20. Four T-state Two Word Instruction

TABLE 2-24. Data Memory Wait States

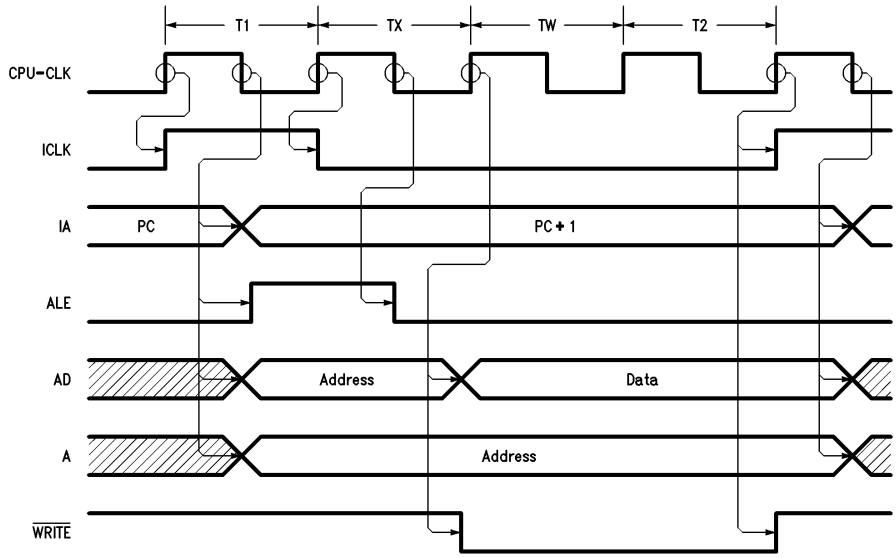
{DCR[2–0]}	Data Wait States
000	0
001	1
010	2
011	3
100	4
101	5
110	6
111	7

TABLE 2-25. Instruction Memory Wait States

{DCR[4,3]}	Instruction Wait States
00	0
01	1
10	2
11	3

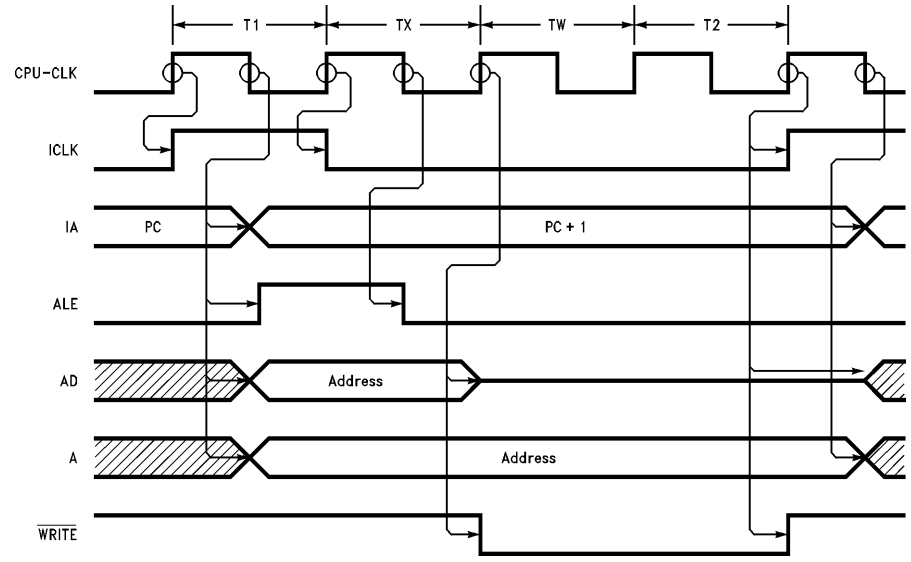
2.0 CPU Description (Continued)

lowing the instruction after the write to {DCR}. A write to {DCR[2–0]} to change the number of data wait states will take effect on the next data memory access instruction even if it immediately follows the write to {DCR}. A write to {DCR [2–0]} to change the number of data wait states or to {ACR [4TR]} will take effect on the next data memory access instruction even if it immediately follows write to {DCR} or {ACR}. Both instruction and data wait states cause the insertion of additional T-states prior to T2 and these T-states are referred to as TW. The purpose of instruction wait states is to increase the time from instruction address generation to the beginning of the next instruction cycle. Data wait states increase the time from data memory address generation to the removal of the strobe at the end of data memory access instructions. Therefore, instruction and data wait states are counted concurrently in a data memory access instruction and TX of a data memory access instruction is counted as one instruction wait state. The actual number of wait states added to a data memory access is calculated as the maximum between the number of data wait states and one less than the number of instruction wait states. *Figure 2-21* shows a write of data memory with one wait state. This could be accomplished by selecting two instruction wait states or one data wait state. The effect of the wait state is to increase the time the write strobe is active and the data is valid on AD. The same situation for a read of data memory is shown in *Figure* 2-22. Note that if [4TR] is set to one then one data wait state has no additional affect on a read of data memory and the timing is the same as shown in *Figure 2-18.* The affect of two data memory wait states and [4TR] set to one is shown in *Figure 2-23.* A two T-state instruction with two instruction wait states is shown in *Figure 2-24* and a four T-state instruction with one instruction wait state is shown in *Figure 2-25.* As stated earlier, instruction wait states are inserted before T2. Adding wait states to a four T-state two word instruction causes the wait states to count twice when calculating total instruction cycle time. The wait states are added to each of the two words of the instruction.



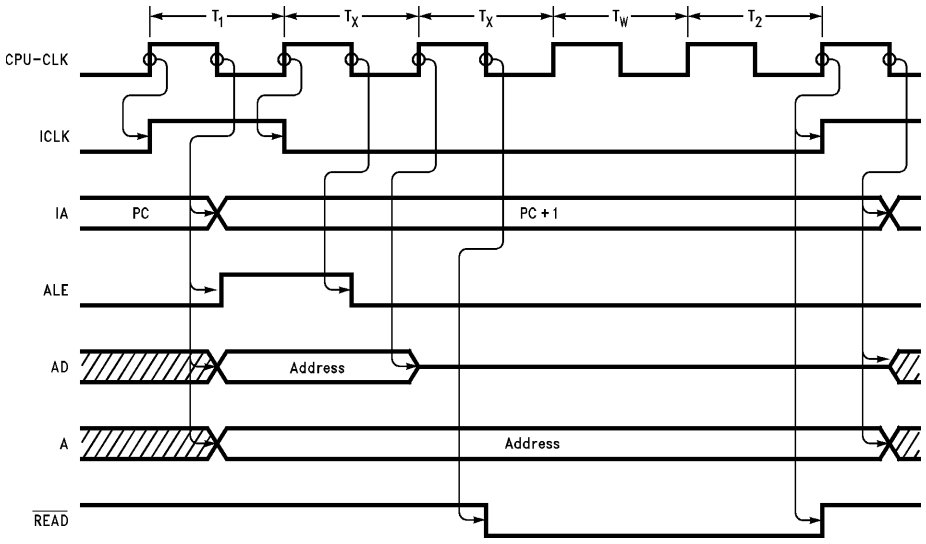
TL/F/9336–E4

FIGURE 2-21. Data Memory Write with One Wait State



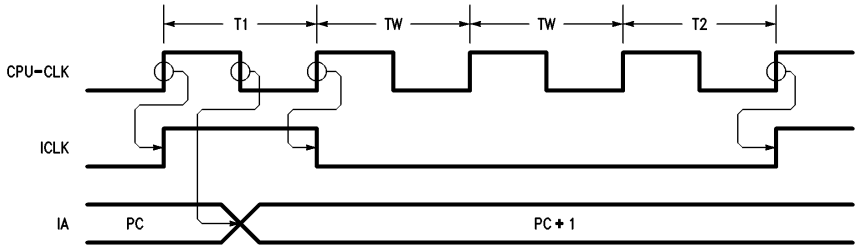
TL/F/9336–E5

FIGURE 2-22. Data Memory Read with One Wait State and [4TR] = 0



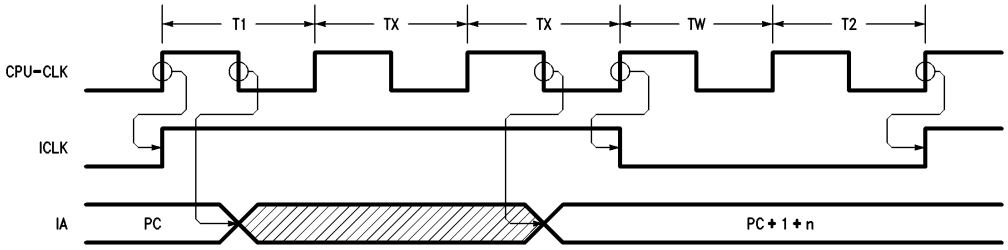
TL/F/9336–H6

FIGURE 2-23. Data Memory Read with Two Wait States and [4TR] = 1



TL/F/9336–E6

FIGURE 2-24. Two T-state Instruction with Two Wait States

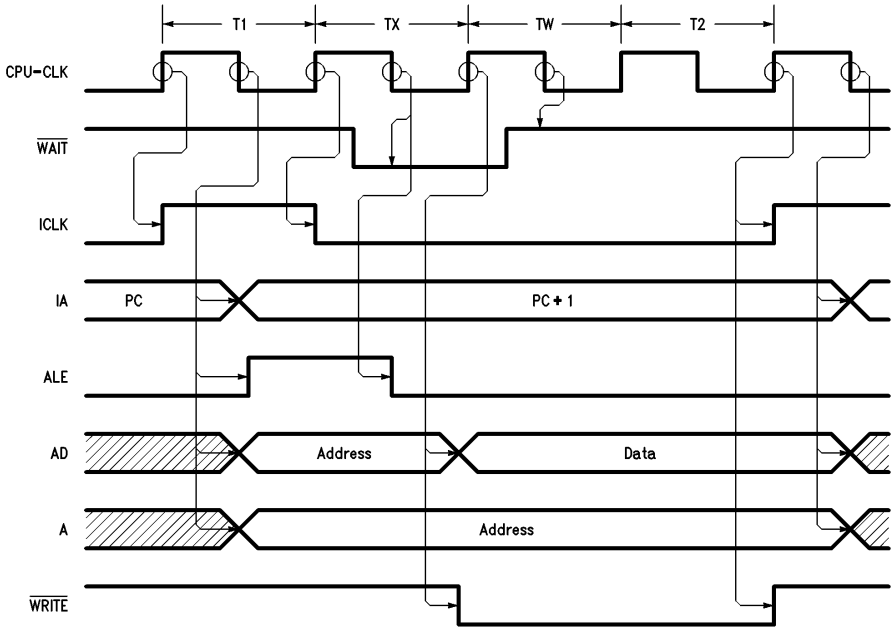


TL/F/9336–E7

FIGURE 2-25. Four T-state Instruction with One Wait State

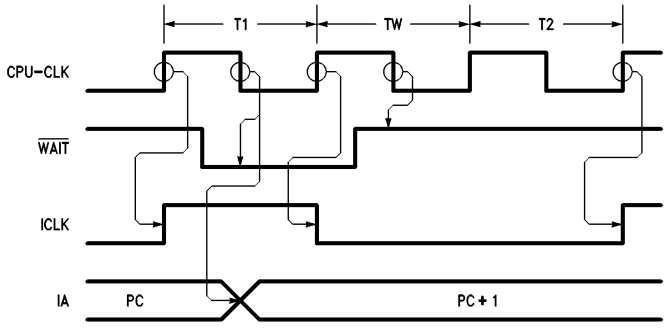
2.0 CPU Description (Continued)

The $\overline{\text{WAIT}}$ pin can also be used to add wait states to BCP instruction execution. The CPU will be waited as long as $\overline{\text{WAIT}}$ is low. To wait a given instruction, $\overline{\text{WAIT}}$ must be asserted low one-half T-state prior to the beginning of T2 in the instruction to be affected. *Figure 2-26* shows $\overline{\text{WAIT}}$ asserted during a write to data memory. In order to wait this instruction, $\overline{\text{WAIT}}$ must fall prior to the falling edge of CPU-CLK in TX. One wait state is added to the access and $\overline{\text{WAIT}}$ rises prior to the falling edge of CPU-CLK in TW which allows the access to finish. If $\overline{\text{WAIT}}$ had remained low, the access would have been held off indefinitely. Programmed wait states would delay when $\overline{\text{WAIT}}$ must be asserted since they would delay the beginning of T2. *Figures 2-27* through *Figure 2-29* depict the use of $\overline{\text{WAIT}}$ with three other instruction types. In all three cases, $\overline{\text{WAIT}}$ is asserted one-half T-state prior to when T2 would normally begin. Also, it is evident that the effect of $\overline{\text{WAIT}}$ on instruction timing is identical to adding programmed wait states.



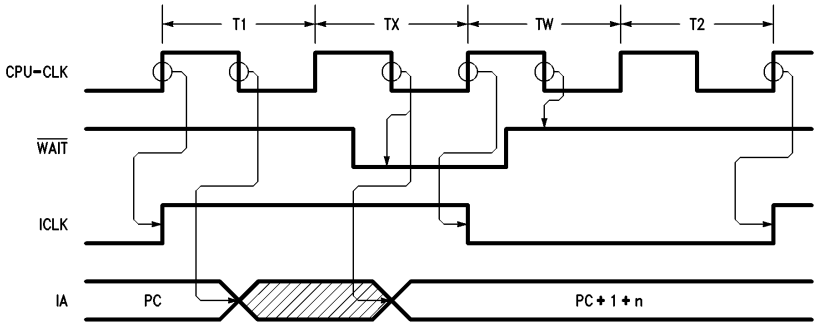
TL/F/9336–E8

FIGURE 2-26. Data Memory Access $\overline{\text{WAIT}}$ Timing



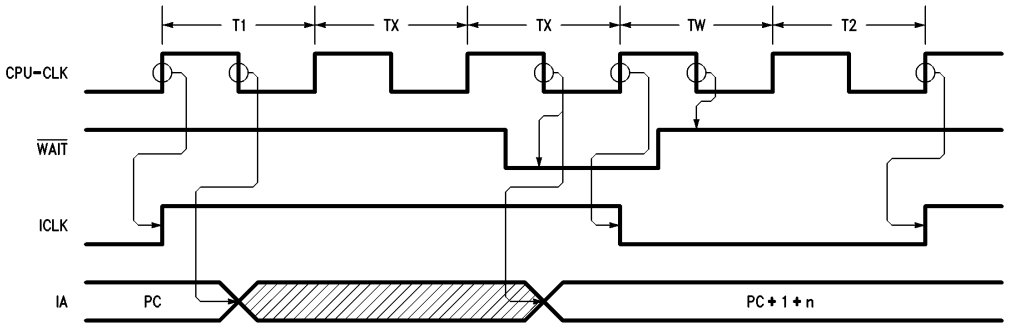
TL/F/9336–E9

FIGURE 2-27. Two T-state Instruction $\overline{\text{WAIT}}$ Timing



TL/F/9336–F1

FIGURE 2-28. Three T-state Program Control Instruction WAIT Timing

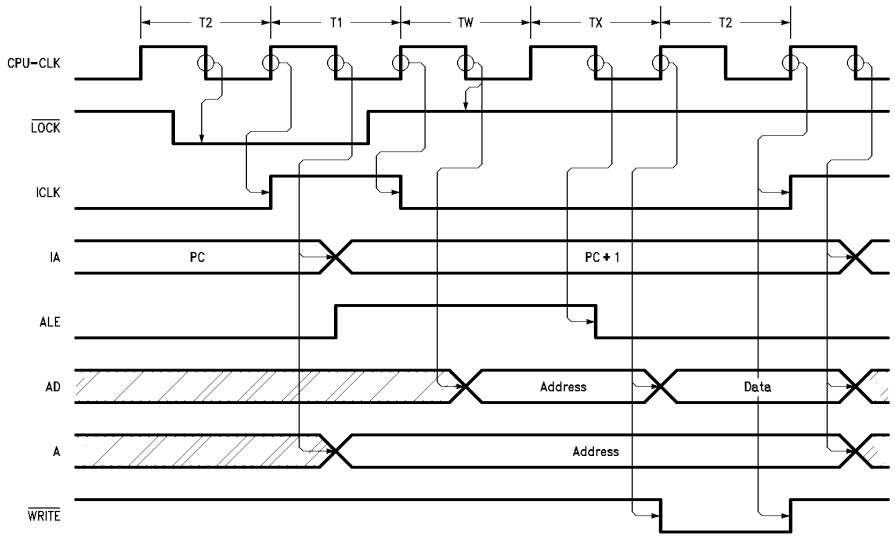


TL/F/9336–F2

FIGURE 2-29. Four T-state Program Control Instruction WAIT Timing

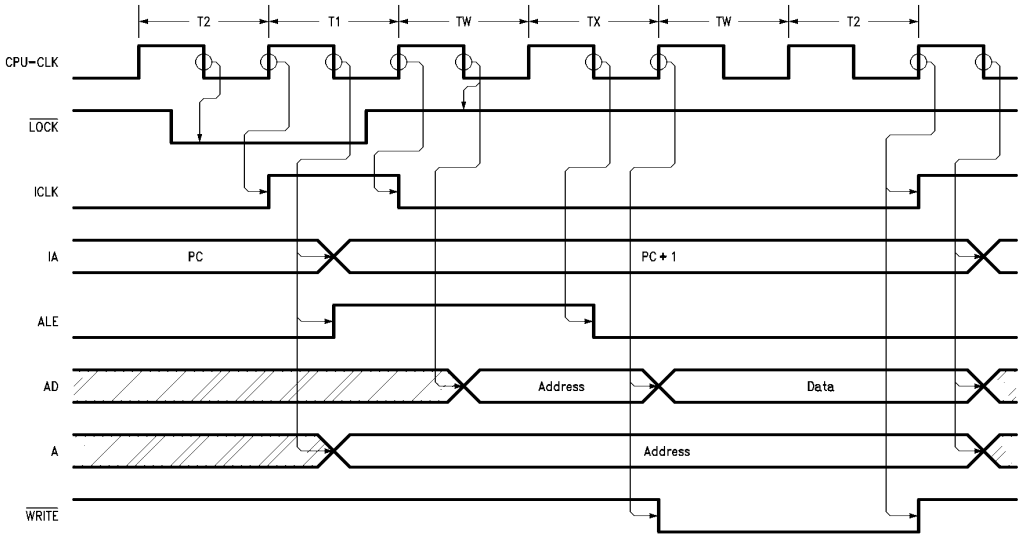
LOCK is another input which affects BCP instruction timing. LOCK prevents the BCP from accessing data memory. When asserted low, LOCK will cause the BCP to wait when it executes a data memory access instruction. The BCP will be waited until LOCK is taken high. To prevent a given access of data memory, LOCK must be asserted low one-half T-state prior to the beginning of the instruction accessing data memory. *Figure 2-30* shows LOCK being used to wait a write to data memory. LOCK falls prior to the falling edge of CPU-CLK before T1. In order to guarantee at least one wait state, LOCK is held low until after the falling edge of CPU-CLK in T1. This causes the insertion of TW into the cycle prior to TX. ALE remains high and the address is delayed on AD until LOCK is removed. After LOCK rises the access concludes normally with ALE falling halfway through TX and WRITE occurring during T2. Note that LOCK waits the access at a different point in the cycle than programmed wait states or WAIT. Additional wait states could occur from these sources prior to T2. *Figure 2-31* shows an example of LOCK holding off a write to data memory with one programmed wait state.

With timing similar to LOCK, the BCP will be delayed from making a data memory access by an access from the remote system. If the remote system is accessing the Remote Interface Configuration register, {RIC}, or data memory, the BCP will be waited by the Remote Interface and Arbitration System, RIAS, until the remote access is finished. The length of time the BCP is waited depends on the speed of the remote system and the type of remote access. The wait states are added prior to TX in the same manner as for LOCK shown in *Figure 2-30*. A more detailed description of the operation of RIAS can be found in Section 4.0, Remote Interface and Arbitration System.



TL/F/9336–F3

FIGURE 2-30. LOCK Timing



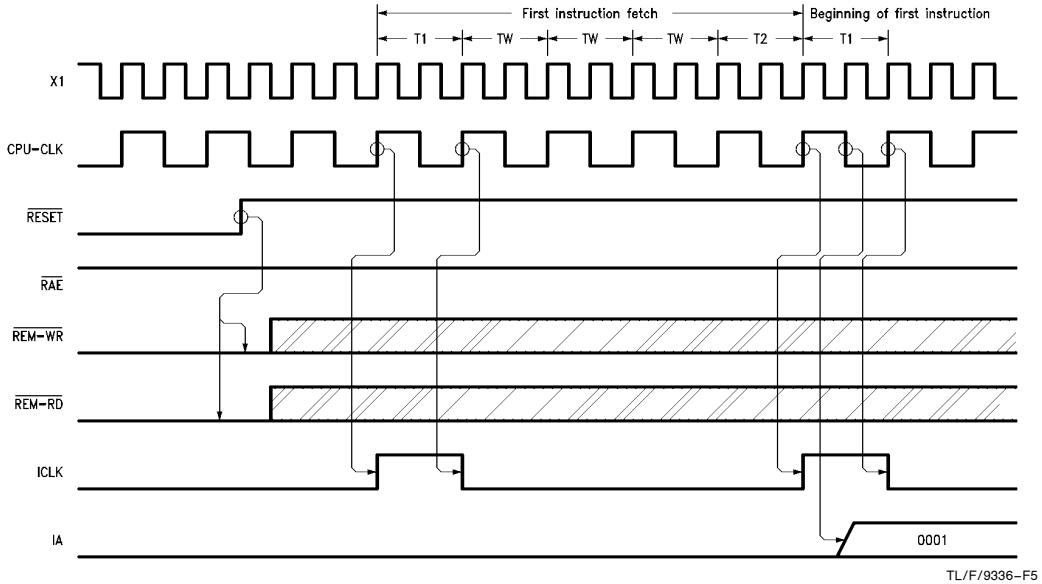
TL/F/9336–F4

FIGURE 2-31. LOCK Timing with One Wait State

2.0 CPU Description (Continued)

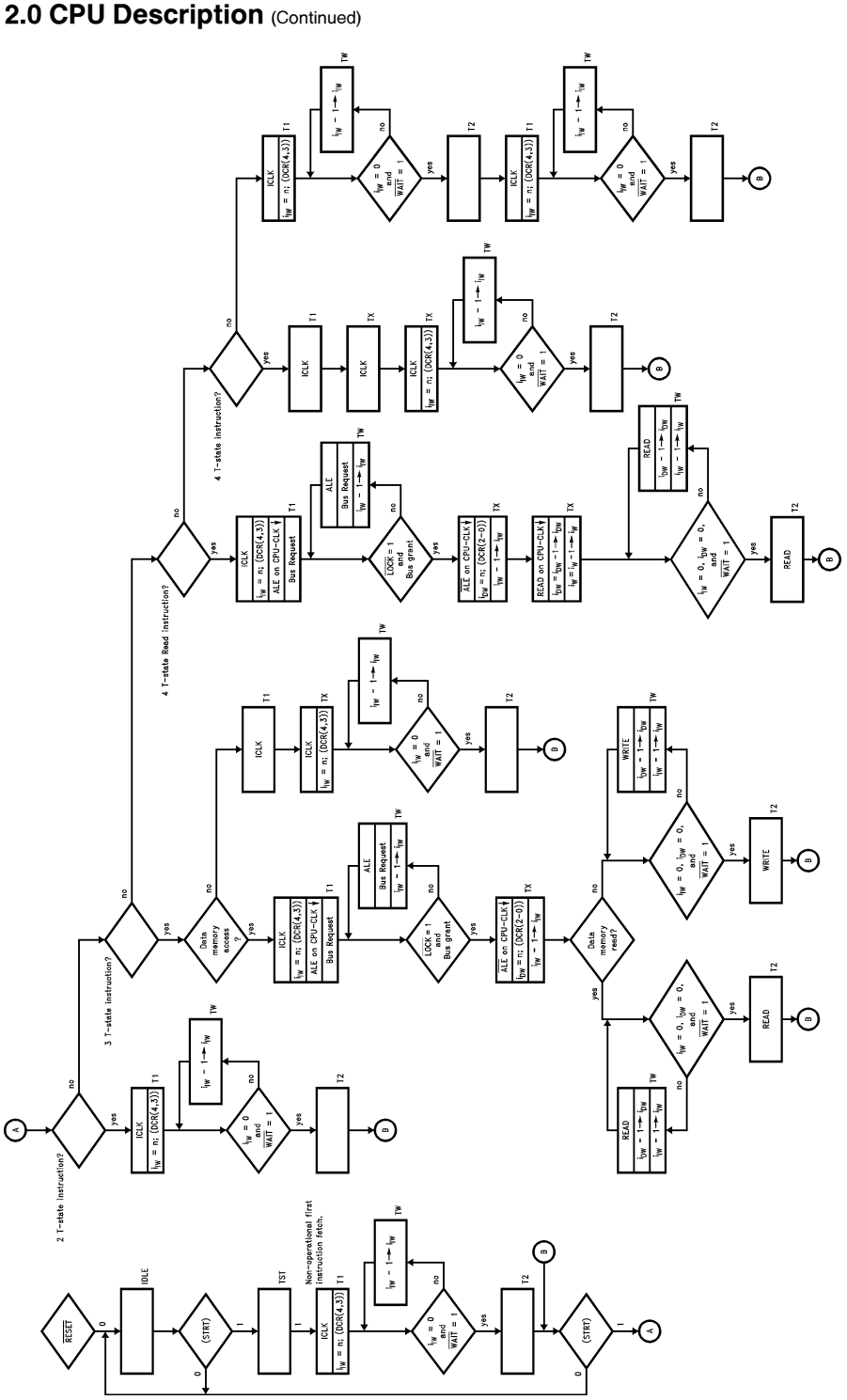
The CPU will be stopped after $\overline{\text{RESET}}$ is asserted low. The CPU can be externally controlled by changing the state of the start bit, [STRT], in {RIC}. The CPU starts executing instructions from the current address in the program control register when a one is written to [STRT] and stops when [STRT] is cleared. The CPU will complete the current instruction before stopping. Controlling the CPU from {RIC} requires a processor to access {RIC}. If no external processor is present, the CPU can be made to start automatically after reset by holding $\overline{\text{REM-WR}}$ and $\overline{\text{REM-RD}}$ low and $\overline{\text{RAE}}$ high while $\overline{\text{RESET}}$ is transitioning from low to high. The CPU "kick-starts" and will begin executing instructions from address zero. The timing for kick-starting the CPU is shown in *Figure 2-32.* ICLK rises on the rising edge of CPU-CLK one T-state after $\overline{\text{RESET}}$ is de-asserted. The falling edge of ICLK signifies the beginning of the first instruction fetch. Three instruction wait states and T2 precede the first instruction.

A functional state diagram describing the timing of the CPU is shown in *Figure 2-33.* The functional state diagram is similar to a flow chart, except that transitions to a new state (states are denoted as rectangular boxes) can only occur on the rising edge of the CPU-CLK. A state box can specify several actions, and each action is separated by a horizontal line. A signal name listed in a state box indicates that that pin will be asserted high when Timing Control has entered that state. When the signal is omitted from a box, it is asserted low. (Note: this requires using the inversion of a signal in some cases.) Decision blocks are shown as diamonds and their meaning is the same as in a flow chart. The functional state diagram is a generalized approach to determining instruction flow while allowing for any combination of wait states and control signals. Timing Control always starts from a reset in the state IDLE. After $\overline{\text{RESET}}$ goes high, Timing Control remains in IDLE until [STRT] is written high. If the BCP kick-starts, Timing Control enters TST on the next rising edge of CPU-CLK. Timing Control starts with a dummy instruction cycle in order to fetch the first instruction. ICLK goes high in T1 and the instruction wait state counter is loaded. ICLK falls when either T2 or TW is entered as determined by the value of i_{IW} and $\overline{\text{WAIT}}$. The normal instruction flow begins after T2 at B on the diagram. As an example, consider a three T-state data memory write instruction with one data wait state. The instruction cycle path for this instruction would begin at T1 following the decision block for data memory access. In T1, ICLK is asserted high, the instruction wait state counter is loaded, and a bus request to RIAS is generated. Also, ALE is asserted high on the falling edge of CPU-CLK during T1. A branch decision is now made based on the state of $\overline{\text{LOCK}}$ and the response from RIAS to the bus request. Assuming that $\overline{\text{LOCK}}$ is not asserted and a remote access is not in progress, Timing Control enters TX on the next rising edge of CPU-CLK. In TX, the data wait state counter is loaded and the instruction wait state counter is decremented. In this example, the instruction wait state counter is at zero and is not counting. The data wait state counter is loaded with one. ALE goes low on the falling edge of CPU-CLK during TX. The next decision block checks for a read of data memory. This example is a write to data memory so the decision is no and the branch is to the right. The wait state conditions are evaluated in the following decision block. i_{DW} is one and Timing Control enters TW on the next rising edge of CPU-CLK. $\overline{\text{WRITE}}$ is asserted low when TW is entered and the data wait state counter is decremented to zero. The decision on i_{DW}, i_{IW}, and $\overline{\text{WAIT}}$ is now true and T2 is entered on the next rising edge of CPU-CLK. $\overline{\text{WRITE}}$ remains low. The CPU will stop execution if [STRT] is low at B in the diagram. Otherwise, the next instruction will be executed beginning at A. To summarize, this instruction went through the following states: T1, TX, TW, and T2. The complete instruction cycle is shown in *Figure 2-21.* Any instruction cycle can be analyzed in a similar manner using this functional state diagram.



TL/F/9336–F5

FIGURE 2-32. CPU Start-Up Timing



TL/F/9336–F6

FIGURE 2-33. Functional State Diagram of CPU Timing

2.0 CPU Description (Continued)

2.2.3 Interrupts

The DP8344B has two external and four internal interrupt sources. The external interrupt sources are the Non-Maskable Interrupt pin, $\overline{NMI}$, and the Bi-directional Interrupt Request pin, $\overline{BIRQ}$.

External

A non-maskable interrupt is detected by the CPU when a falling edge is detected at the $\overline{NMI}$ pin. The interrupt is automatically cleared internally when the CPU recognizes the interrupt.

$\overline{BIRQ}$ can function as both an interrupt into the DP8344B and as an output which can be used to interrupt other devices. $\overline{BIRQ}$ is configured as an input or output according to the state of [BIC] in the Auxiliary Control Register, {ACR}. $\overline{BIRQ}$ is an input if [BIC] is a zero and an output when [BIC] is a one. The reset state of [BIC] is a zero, causing $\overline{BIRQ}$ to be an input after the BCP is reset. [BIRQ] in the Condition Code Register, {CCR}, is a read only bit which mirrors the state of $\overline{BIRQ}$ regardless of whether $\overline{BIRQ}$ is configured as an input or output. This bit is updated at the beginning of T1 of each instruction.

When $\overline{BIRQ}$ is configured as an input, an interrupt will occur if the pin is held low. $\overline{BIRQ}$ must be held low until the interrupt is recognized or the interrupt will not be processed. Due to the prioritizing of interrupts as described below, $\overline{BIRQ}$ may not be recognized by the CPU until higher priority interrupts have been serviced. $\overline{BIRQ}$ will be recognized after higher priority interrupts have been processed. The low state on $\overline{BIRQ}$ should be removed after the CPU recognizes the interrupt or the interrupt will be processed multiple times.

When $\overline{BIRQ}$ is configured as an output, its state is controlled by [IM3] in the Interrupt Control Register, {ICR}. Changing the state of this bit will change $\overline{BIRQ}$ at the beginning of T1 of the instruction following the write to [IM3]. Note that [BIRQ] in {CCR} is also updated at the beginning of T1. Therefore, there is a one instruction cycle delay from when [IM3] changes to when the new value of $\overline{BIRQ}$ is made available in [BIRQ]. [BIS] in the Remote Interface Configuration register, {RIC}, mirrors the state of [IM3]. When $\overline{BIRQ}$ is an output, writing a one to [BIS] will change the state of [IME] thus changing $\overline{BIRQ}$ and allowing a remote processor to acknowledge an interrupt from the BCP. Note, if the BCP code operates on [IM3] at the same time that the remote processor acknowledges the interrupt by writing a one to [BIS], $\overline{BIRQ}$ will toggle and then assume the state of [IM3] resulting from the BCP code operation. Therefore, if the designer chooses to operate on [IM3] while waiting for the remote processor to acknowledge a $\overline{BIRQ}$ interrupt, the designer should ensure that the remote processor is locked out from accessing [BIS] during the operation on [IM3]. This can be accomplished by setting [LOR] in {ACR}, having the BCP perform a data memory access to ensure that any current remote accesses are complete, operating an [IM3], and finally clearing [LOR]. $\overline{BIRQ}$ will change state two T-states after the end of the write to [BIS]. Writing a one to [BIS] will have no effect on [IM3] when $\overline{BIRQ}$ is an input. Table 2-26 summarizes the relationship between $\overline{BIRQ}$ and its associated register bits.

TABLE 2-26. $\overline{BIRQ}$ Control Summary

(a) $\overline{BIRQ}$ is an Input ([BIC] = 0): Remote Processor Controls the State of $\overline{BIRQ}$

[IM3]	[BIS]	$\overline{BIRQ}$	[BIRQ]
0	[IM3] = 0	Active Interrupt to the BCP: state of $\overline{BIRQ}$ controlled by the Remote Processor	Reflects the state of $\overline{BIRQ}$
1	[IM3] = 1	Masked Interrupt to the BCP: state of $\overline{BIRQ}$ controlled by the Remote Processor	Reflects the state of $\overline{BIRQ}$

(b) $\overline{BIRQ}$ is an Output ([BIC] = 1): BCP Controls the State of $\overline{BIRQ}$

[IM3]	[BIS]	$\overline{BIRQ}$	[BIRQ]
0	[IM3] = 0	State of [IM3] = 0	Reflects the state of $\overline{BIRQ}$ = 0
1	[IM3] = 1	State of [IM3] = 1	Reflects the state of $\overline{BIRQ}$ = 1

(c) $\overline{BIRQ}$ is an Output ([BIC] = 1): Remote Processor Acknowledges $\overline{BIRQ}$

[BIS]	[IM3]	[BIS]	$\overline{BIRQ}$	[BIRQ]
Remote Processor writes a 1 to [BIS]	Toggles	[IM3]	State of [IM3]	Reflects the state of $\overline{BIRQ}$

2.0 CPU Description (Continued)

Internal

The internal interrupts consist of the Transmitter FIFO Empty, TFE, interrupt, the Line Turn Around, LTA, interrupt, the Time Out, TO, interrupt, and a user selectable receiver interrupt source. The receiver interrupt source is selected from either the Receiver FIFO, Full, RFF, interrupt, the Data Available, DA, interrupt, or the Receiver Active, RA, interrupt. The receiver interrupt is selected using bits [RIS1] and [RIS0] in the Interrupt Control Register, {ICR}. See the Section 3.0, Transceiver for a description of these interrupts.

Masking

The BCP uses two levels of interrupt masking: a global interrupt mask which affects all interrupts except $\overline{\text{NMI}}$ and individual interrupt mask bits. Global enabling and disabling of the interrupts is performed by changing the state of the Global Interrupt Enable bit, [GIE], in {ACR}. The maskable interrupts are disabled when [GIE] is a zero and enabled when [GIE] is a one. [GIE] is a zero after the BCP is reset. [GIE] is a read/write register bit and may be changed by using any instruction that can write to {ACR}. In addition, the RET, RETF, and EXX instructions have option fields which can be used to alter the state of [GIE]. The EXX instruction can set or clear [GIE] as well as leaving it unchanged. The RET and RETF instructions can restore [GIE] to the value that was saved on the address stack at the time the interrupt was recognized. These instructions also provide the options of clearing or setting [GIE] or leaving it unchanged. [GIE] is set to a zero when an interrupt is recognized by the CPU. It is necessary to set [GIE] to a one if interrupts are to be recognized within an interrupt routine.

The individual interrupt mask bits are located in {ICR}. When set to a one, bits [IM0], [IM1], [IM2], [IM3], and [IM4] in {ICR} mask the receiver interrupt, TFE interrupt, LTA interrupt, $\overline{\text{BIRQ}}$ interrupt, and TO interrupt, respectively. To enable an interrupt, its mask bit must be set to a zero. The interrupts and associated mask bits are shown in Table 2-27. These bits are set to a one when the DP8344 is reset.

Masking interrupts with [GIE] or the mask bits in {ICR} prevents the CPU from acknowledging interrupts but does not prevent the interrupts from occurring. Therefore, if an interrupt is asserted, it will be processed as soon as it is unmasked by changing [GIE] to a one and/or changing the appropriate mask bit in {ICR} to a zero.

Priorites

When more than one interrupt is unmasked and asserted, the CPU processes the interrupt with the highest priority first. $\overline{\text{NMI}}$ has the highest priority followed by the receiver interrupt, TFE, LTA, $\overline{\text{BIRQ}}$, and TO. Each time the interrupts are sampled, the highest priority interrupt is processed first, regardless of how long a lower priority interrupt has been active. Interrupt priority is summarized in Table 2-27.

TABLE 2-27. {ICR} Interrupt Mask Bits and Interrupt Priority

Interrupt	Mask Bit	Priority
$\overline{\text{NMI}}$	—	Highest
RFF, DA, RA	[IMO]	
TFE	[IM1]	
LTA	[IM2]	
$\overline{\text{BIRQ}}$	[IM3]	
TO	[IM4]	Lowest

A call to the interrupt address is generated when an interrupt is detected by the CPU. The address for each interrupt is constructed by concatenating the Interrupt Base Register, {IBR}, contents with the individual interrupt code as shown in Table 2-28. There is room between the interrupt addresses for a maximum of four instruction words.

TABLE 2-28. Interrupt Vector Generation

Interrupt	Code
$\overline{\text{NMI}}$	111
RFF, DA, RA	001
TFE	010
LTA	011
$\overline{\text{BIRQ}}$	100
TO	101

Interrupt Vector

{IBR} Contents	0 0 0	Code	0 0
15	8	5	2 0

Interrupts are sampled by each falling edge of the CPU clock with the last falling edge prior to the start of the next instruction determining whether an interrupt will be processed. The timing of a typical interrupt event is shown in *Figure 2-34.* The interrupt occurs during the current instruction and is sampled by the falling edge of the CPU clock. The next instruction is not operated on and its address is stored in the internal address stack along with [GIE], the ALU flags, and the register bank positions. The address stack is twelve words deep. A two T-state internal call is now executed in place of the non-executed instruction. This call will cause a branch to the interrupt address that is generated in the first half of T-state T1. Also, [GIE] is cleared at the end of the first half of T-state T1. The internal call to the interrupt address is subject to instruction wait states as configured in {DCR}.

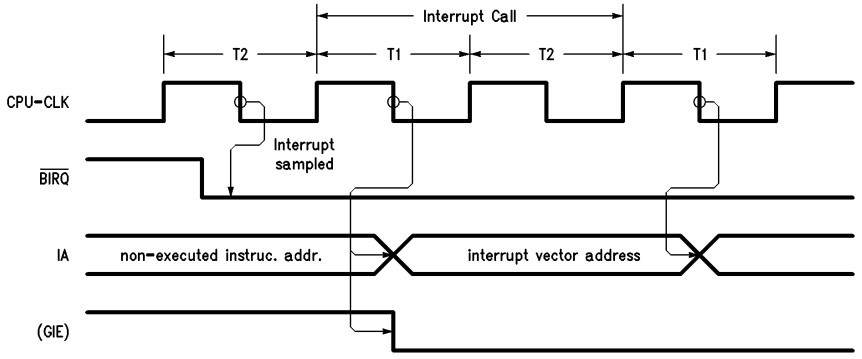
2.2.4 Oscillator

The crystal oscillator is an on-chip amplifier which may be used with an external crystal to generate accurate CPU and transceiver clocks. The input to this amplifier is X1, pin 33. The output of the amplifier is X2, pin 34. When X1 and X2 are connected to a crystal and external capacitors *(Figure 2-35),* the combined circuit forms a Pierce crystal oscillator with the crystal operating at parallel resonance. Crystals that oscillate over the frequency range of 2 MHz to 20 MHz may be used. The recommended crystal parameters for operation with the oscillator are given in Table 2-29. The external capacitor values should be chosen to provide the manufacturer's specified load capacitance for the crystal when combined with the parasitic capacitance of the trace, socket, and package. As an example, a crystal with a specified load capacitance of 20 pF used in a circuit with 13 pF per pin parasitic capacitance will require external capacitor values of 27 pF each. This provides an equivalent capacitance of 40 pF on each side of the crystal, and has a 20 pF series equivalent value across the crystal.

As an alternative to the crystal oscillator, an external clock source may be used. In this case, the external clock source should be connected to X1 and no external circuitry should be connected to X2 *(Figure 2-36).* The DP8344 can supply a clock source, equal in frequency to the crystal oscillator or external clock source, to other circuitry via pin 35, the CLKOUT output. This output is a buffered version of the signal at X1.

TABLE 2-29. Recommended Crystal Parameters

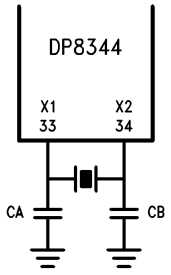
AT Cut, Parallel Resonant
Fundamental Mode
Load Capacitor = 20 pF
Series Resistance $< 20\Omega$
Frequency Tolerance 0.005% at 25°C
Stability 0.01% 0°–70°C
Drive Level 0.5 mW Typical



TL/F/9336–F7

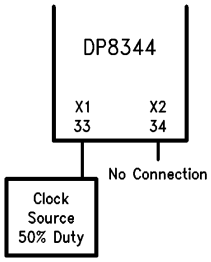
FIGURE 2-34. Interrupt Timing

2.0 CPU Description (Continued)



TL/F/9336–F8

FIGURE 2-35. DP8344B Operation with Crystal



TL/F/9336–F9

FIGURE 2-36. DP8344B Operation with External Clock

3.0 Transceiver

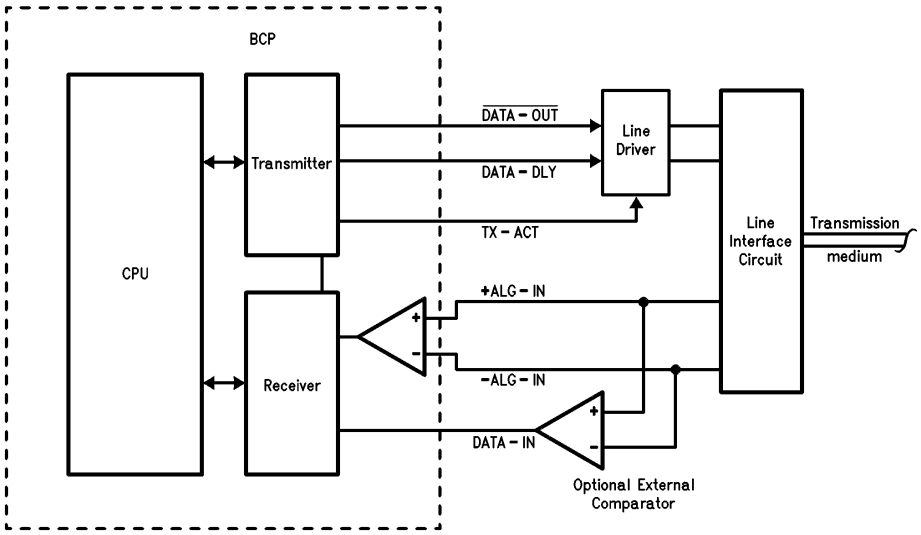
3.1 TRANSCEIVER ARCHITECTURAL DESCRIPTION

The transceiver section operates as an on-chip, independent peripheral, implementing all the necessary formatting required to support the physical layer of the following serial communications protocols:

- IBM 3270 (including 3299)
- IBM 5250
- NSC general purpose 8-bit

The CPU and transceiver are tightly coupled through the CPU register space, with the transceiver appearing to the CPU as a group of special function registers and three dedicated interrupts. The transceiver consists of separate transmitter and receiver logic sections, each capable of independent operation, communicating with the CPU via an asynchronous interface. This interface is software configurable for both polled and interrupt-driven interaction, allowing the system designer to optimize his product for the specific application.

The transceiver connects to the line through an external line interface circuit which provides the required DC and AC drive characteristics appropriate to the application. A block diagram of such an interface is shown in *Figure 3-1*. An on-chip differential analog comparator, optimized for use in a transformer coupled coax interface, is provided at the input to the receiver. Alternatively, if an external comparator is necessary, the input signal may be routed to the DATA-IN pin.



TL/F/9336–33

FIGURE 3-1. System Block Diagram, Showing Details of the Line Interface

3.0 Transceiver (Continued)

The transceiver has several modes of operation. It can be configured for single line, half-duplex operation in which the receiver is disabled while the transmitter is active. Alternatively, both receiver and transmitter can be active at the same time for multi-channel (such as repeater) or loopback operation. The transceiver has both internal and external loopback capabilities, facilitating testing of both the software and external hardware. At all times, both transmitter and receiver operate according to the same protocol definition.

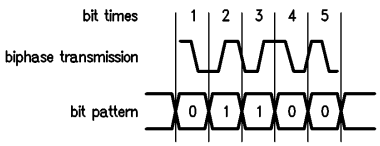
3.1.1 Protocols

In all protocols, data is transmitted serially in discrete messages containing one or more frames, each representing a single word of information. Biphase (Manchester II) encoding is used, in which the data stream is divided into discrete time intervals (bit-times) denoted by a level transition in the center of the bit-time. For the IBM 3270, 3299 and NSC general purpose 8-bit protocols, a mid-bit transition from low to high represents a biphase ''1'', and a mid-bit transition from high to low represents a biphase ''0''. For the 5250 protocol, the definition of biphase logic levels is exactly reversed, i.e. a biphase ''1'' is represented by a high to low transition. Depending on the bit sequence, there may or may not be a transition on the bit-time boundary. The biphase encoding of a simple bit sequence is illustrated in *Figure 3-2(a)*.

Each transmission begins with a unique start sequence consisting of 5 biphase encoded ''1's'', (referred to as ''line quiesce pulses'') followed by a 3 bit-time code violation and the sync bit of the first frame, *Figure 3-2(b)*. The three bit-time code violation does not conform to the rules of Manchester encoding and forms a unique recognition pattern for bit time synchronization by the receiver logic. The first bit of any frame is the sync bit, a biphase ''1''. The frame is then formatted according to the requirements of the protocol. If a multi-frame message is being transmitted, additional frames are appended to the end of the first frame—except for the 5250 protocol, where there may be an optional number of ''fill bits'' (biphase ''0'') between each frame.

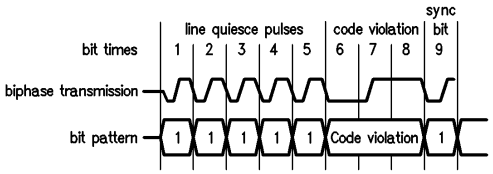
Depending on the protocol, when all data has been transmitted, the end of a message will be indicated either by the transmission of an ending sequence, or (for 5250) simply by the cessation of transitions on the differential line. Later model 5250 equipment has incorporated a ''line hold'' at the end of the message. The line hold maintains the final differential state on the line for several bit times to eliminate noise or reflections that could be interpreted as a continuance of the message. The ending sequence for all but 5250 protocols consists of a single biphase ''0'' followed by a low to high transition on the bit-time boundary and two bit-times with no transitions (two mini-code violation), *Figure 3-2(c).*

The various protocol framing formats are shown in *Figures 3-3* through *3-5*. The diagrams use a bit pattern drawing convention which, for clarity, shows the bit-time boundaries but not the biphase transitions in the center of the bit times. The timing relationship between the biphase encoded bit stream and the bit pattern diagrams is consistent with *Figure 3-2*.



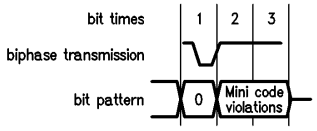
TL/F/9336–34

(a) Biphase Encoding



TL/F/9336–36

(b) Starting Sequence



TL/F/9336–35

(c) Ending Sequence

FIGURE 3-2. Biphase Encoding

3.1.1.1 IBM 3270

The framing format of the IBM 3270 coax protocol is shown in *Figures 3-3(a)* and *(b)*, for both single and multi-frame messages. Each message begins with a starting sequence and ends with an ending sequence, as shown in *Figures 3-2(b)* and *(c)*. Each 12-bit frame begins with a sync bit (B1) followed by an 8-bit data byte (MSB first), a 2-bit control field, and the frame delimiter bit (B12), representing even parity on the previous 11 bits. The bit rate on the coax line is 2.3587 MHz.

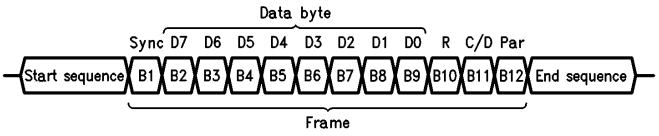
3.1.1.2 IBM 3299

Adding 3299 multiplexers to the 3270 environment requires an address to be transmitted along with each message from the controller to the multiplexer. The IBM 3299 Terminal Multiplexer protocol provides this capability by defining an additional 8-bit frame as the first frame of every message sent from the controller, as shown in *Figure 3-3(c)*. This frame contains a 6-bit data field along with the normal sync and word parity bits. The protocol currently utilizes bits B2–B4 as an address field that directs the message through the multiplexor hardware. Following the address frame, the rest of the message follows standard 3270 convention. The bit rate, 2.3587 MHz, is the same as standard 3270.

3.1.1.3 IBM 5250

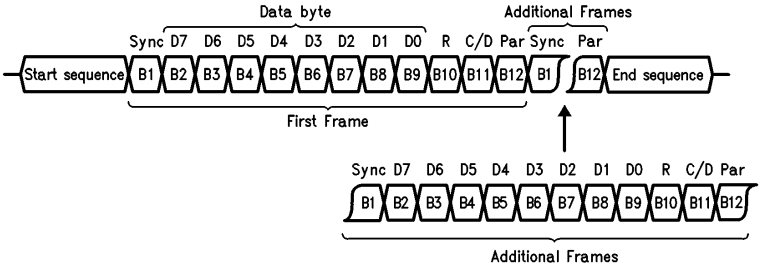
The framing format of the IBM 5250 twinax protocol is shown in *Figure 3-4*, for both single and multi-frame messages. Each message begins with the starting sequence shown in *Figure 3-2(b)*, and ends with 3 fill bits (biphase ''0''). A 16-bit frame is employed, consisting of a sync bit (B15); an 8-bit data byte (B7–B14) (LSB first); a 3-bit station address field (B4–B6); and the last bit (B3) representing

3.0 Transceiver (Continued)



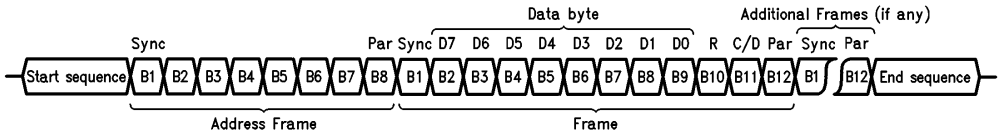
TL/F/9336–37

(a) 3270 Single-Byte Message



TL/F/9336–38

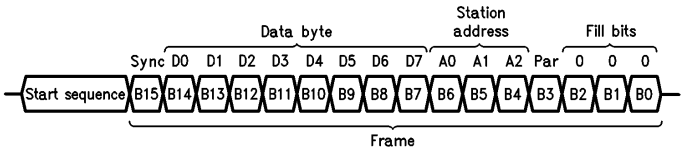
(b) 3270 Multi-Byte Message



TL/F/9336–39

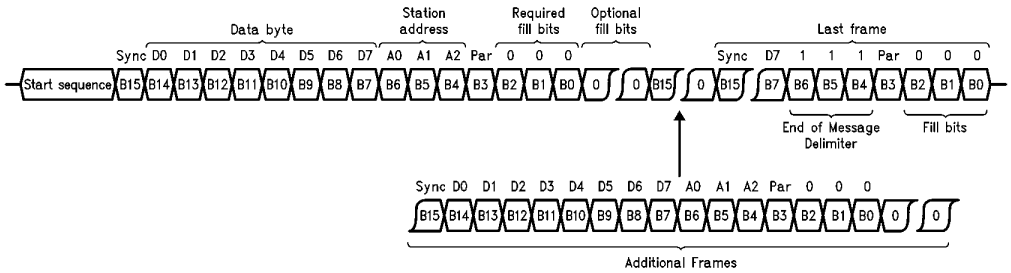
(c) 3299 Controller/Multiplexer Message

FIGURE 3-3. 3270/3299 Protocol Framing Format



TL/F/9336–40

(a) 5250 Single-Byte Message



TL/F/9336–41

(b) 5250 Multi-Byte Message

FIGURE 3-4. 5250 Protocol Framing Format

3.0 Transceiver (Continued)

even word parity on the previous 12 bits. Following the parity bit, 3 biphase "0" fill bits (B0–B2) are transmitted. Following these required fill bits, up to 240 additional fill bits can be inserted between frames before the next sync bit and the start of the next frame of a multi-byte message. The bit rate on the twinax line is 1 MHz.

3.1.1.4 General Purpose 8-Bit

The framing format of the general purpose 8-bit protocol is shown in *Figure 3-5*, for both single and multi-frame messages. It is identical to that used by the National Semiconductor DP8342 transmitter and DP8343 receiver chips. Each message begins with a starting sequence and ends with an ending sequence, as shown in *Figures 3-2(b)* and *(c)*. A 10-bit frame is employed, consisting of the sync bit (B1); an 8-bit data byte (B2–B9) (LSB first); and the last bit of the frame (B10) representing even word parity on the previous 9 bits. For multiplexed applications, the first frame can be designated as an address frame, with all 8 bits available for the logical address. (See General Purpose 8-bit Modes in this section.)

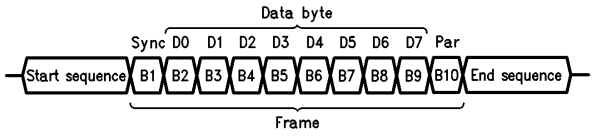
3.2 TRANSCEIVER FUNCTIONAL DESCRIPTION

A block diagram of the transceiver, revealing external inputs and outputs and details of the CPU interface, is shown in *Figure 3-6*. The transmitter and receiver are largely independent of each other, sharing only the clock, reset and protocol select signals. The transceiver is mapped into the CPU register space, thus the status of the transceiver can always be polled. In addition, the CPU/Transceiver interface can be configured for an interrupt-driven environment. (See Transceiver Interrupts in this section.)

Both transmitter and receiver are reset by a common Transceiver Reset bit, [TRES], allowing the CPU to independently reset the transceiver at any time. The Transceiver is also reset whenever the CPU reset is asserted, including the required power-up reset. When [TRES] is asserted, both transmitter and receiver FIFO's are emptied resulting in the Transmit FIFO Empty flag [TFE] being asserted and the Data Available flag [DAV] cleared. Other flags cleared by [TRES] are Transmit FIFO Full [TFF] and Transmitter Active [TA] in the transmitter and Line Active [LA], Receiver Active [RA], Receiver Error [RE], Receive FIFO Full [RFF], Data Error or Message End [DEME], [POLL], [ACK], and [RAR] command flags in the receiver. When [TRES] is asserted, external pin TX-ACT is cleared, DATA-DLY goes to a state equal to the complement of Transmitter INvert [TIN] in {TMR}, and $\overline{\text{DATA-OUT}}$ goes into a state equal to the complement of [TIN] exclusive or'ed with the Advance Transmitter Active [ATA] in {TCR}. In other words, when [TRES] is asserted, DATA-DLY = [TIN], and $\overline{\text{DATA-OUT}}$ = [TIN] $\oplus$ [ATA]. When [TRES] is asserted under software control, it is necessary to wait at least one instruction after asserting [TRES] before seeing the resulting reset state of the affected flags in the CPU. The transmitter and receiver are clocked by a common Transceiver Clock, TCLK, at a frequency equal to eight times the required serial data rate. TCLK can either be obtained from the on-chip oscillator divided by 1, 2 or 4, or from an external clock applied to the X-TCLK pin. TCLK selection is controlled by two Transceiver Clock Select bits, [TCS 1–0] located in the Device Control Register, {DCR}. [TCS 1–0] should only be changed when the transceiver is inactive.

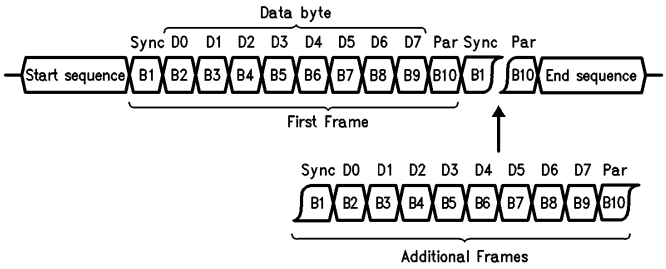
Since the TCLK source can be asynchronous with respect to the CPU clock, the CPU/Transceiver interface can be asynchronous. All flags from the Transceiver are therefore latched at the start of all instructions, and parallel data is transferred through 3 word FIFOs in both the transmitter and receiver.

Protocol selection is controlled by three Protocol Select bits, [PS2–0] in the Transceiver Mode Register, {TMR} (see Table 3-1). Enough flexibility is provided for the BCP to operate in all required positions in the network. It is not pos-



TL/F/9336–42

(a) 8-Bit Single-Byte Message



TL/F/9336–43

(b) 8-Bit Multi-Byte Message

FIGURE 3-5. General Purpose 8-Bit Protocol Framing Format

3.0 Transceiver (Continued)

sible for the transmitter and receiver to operate with different protocols at the same time. The protocol mode should only be changed when both transmitter and receiver are inactive.

If both transmitter and receiver are connected to the same line, they should be configured to operate sequentially (half-duplex). This mode of operation is achieved by clearing the RePeater ENable control bit [RPEN] in {TMR}. In this mode, an active transmitter will disable the receiver, preventing simultaneous operation of transmitter and receiver. If the transmitter FIFO is loaded while the receiver is actively processing an incoming signal, the receiver will be disabled and flag the CPU that a ''Receiver Disabled While Active'' error has occurred. (See Receiver Errors in this section.) On power-up/reset the transceiver defaults to this half-duplex mode.

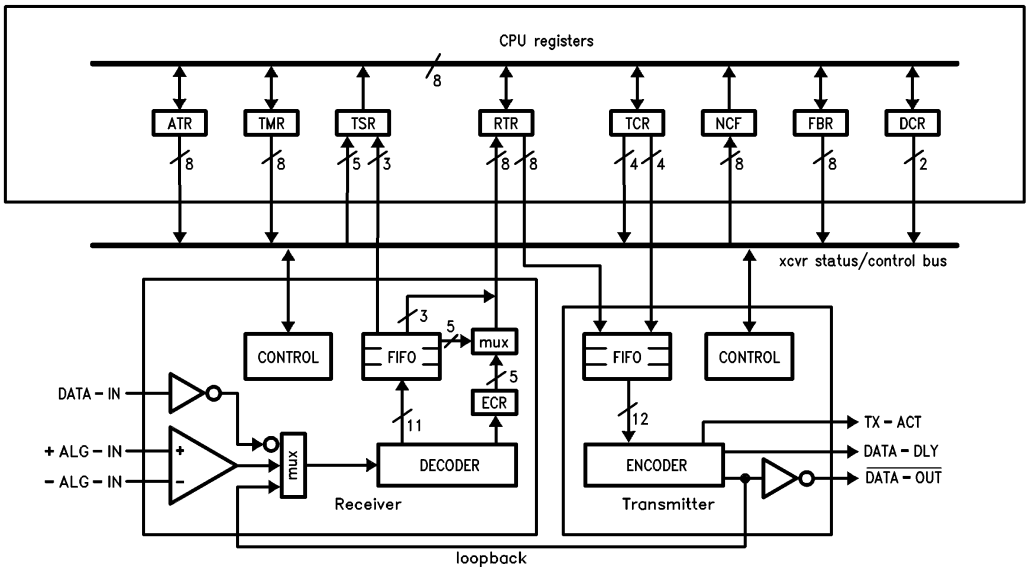
By asserting the Repeat Enable flag [RPEN], the receiver is not disabled by the transmitter, allowing both transmitter and receiver to be active at the same time. This feature provides for the implementation of a repeater function or loopback for test purposes.

The transmitter output can be connected to the receiver input, implementing a local (on-chip) loopback, by asserting [LOOP]. [RPEN] must also be asserted to enable both the transmitter and receiver at the same time. With [LOOP] asserted, the output TX-ACT is disabled, keeping the external line driver in TRI-STATE. The internal flag [TA] is still enabled, as are the serial data outputs.

TABLE 3-1. Protocol Mode Definition

PS2–0	Protocol Mode	Comments
0 0 0	3270	Standard IBM 3270 protocol.
0 0 1	3299 Multiplexer	Receiver expects first frame to be address frame. Transmitter uses standard 3270, no address frame.
0 1 0	3299 Controller	Transmitter generates address frame as first frame. Receiver expects standard 3270, no address frame.
0 1 1	3299 Repeater	Both transmitter and receiver operate with first frame as address frame.
1 0 0	5250	Non-promiscuous mode. [DAV] asserted only when first frame address matches {ATR}.
1 0 1	5250 Promiscuous	[DAV] asserted on all valid received data without regard to address field.
1 1 0	8-Bit	General-purpose 8-bit protocol with first frame address. Non-promiscuous mode. [DAV] asserted only when first frame address matches {ATR}.
1 1 1	8-Bit Promiscuous	[DAV] asserted on all valid received frames.

3.0 Transceiver (Continued)



TL/F/9336–44

KEY TO REGISTERS

RTR	Receive/Transmit Register	ATR	Auxiliary Transceiver Register
TSR	Transceiver Status Register	NCF	Network Command Register
TCR	Transceiver Command Register	FBR	Fill-Bit Register
TMR	Transceiver Mode Register	DCR	Device Control Register

FIGURE 3-6. Block Diagram of Transceiver, Showing CPU Interface

3.0 Transceiver (Continued)

3.2.1 Transmitter

The transmitter accepts parallel data from the CPU, formats it according to the desired protocol and transmits it as a serial biphase-encoded bit stream. A block diagram of the transmitter logic is shown in *Figure 3-6*. Two biphase outputs, $\overline{\text{DATA-OUT}}$, DATA-DLY, and the external line driver enable, TX-ACT, provide the data and control signals for the external line interface circuitry. The two biphase outputs are valid only when TX-ACT is asserted (high) and provide the necessary phase relationship to generate the ''predistortion'' waveform common to all of the transceiver protocols. See *Figure 3-7* for the timing relationships of these outputs as well as the output of the line driver. For a recommended 3270/3299 coax interface, see Section 3.2.5.1 3270 Line Interface. For a recommended 5250 twinax interface see Section 3.2.5.2 5250 Line Interface.

The capability is provided to invert $\overline{\text{DATA-OUT}}$ and DATA-DLY via the Transmitter Invert bit, [TIN], located in the Transceiver Mode Register, {TMR}. In addition, the timing relationship between TX-ACT and the two biphase outputs can be modified with the Advance Transmitter Active control, [ATA]. When [ATA] is cleared low (the power-up condition), the transmitter generates exactly five line quiesce bits at the start of each message, as shown in *Figure 3-7*. If [ATA] is asserted high, the transmitter generates a sixth line quiesce bit, adding one biphase bit time to the start sequence transmission. The line driver enable, TX-ACT, is asserted halfway through this bit time, allowing an additional half-bit to precede the first full line quiesce of the transmitted waveform. Also, the state of DATA-DLY is such that no predistortion results on the line during this first half line quiesce. This modified start sequence is depicted in the dotted lines shown in *Figure 3-7* and is used to limit the initial transient voltage amplitude when the message begins.

Data is loaded into the transmitter by writing to the Receive/Transmit Register {RTR}, causing the first location of the FIFO to be loaded with a 12-bit word (8 bits from {RTR} and 4 bits from the Transceiver Command Register {TCR}. The data byte to be transmitted is loaded into {RTR}, and {TCR} contains additional information required by the protocol. It is important to note that if {TCR} is to be changed, it must be loaded before {RTR}. A multi-frame transmission is accomplished by sequentially loading the FIFO with the required data, the transmitter taking care of all necessary frame formatting.

If the FIFO was previously empty, indicated by the Transmit FIFO Empty flag [TFE] being asserted, the first word loaded into the FIFO will asynchronously propagate to the last location in approximately 40 ns, leaving the first two locations empty. It is therefore possible to load up the FIFO with three sequential instructions, at which time the Transmit FIFO Full flag [TFF] will be asserted. If {RTR} is written while [TFF] is high, the first location of the FIFO will be over-written and that data will be destroyed.

When the first word is loaded into the FIFO, the transmitter starts up from idle, asserting TX-ACT and the Transmitter Active flag [TA], and begins generating the start sequence. After a delay of approximately 16 TCLK cycles (2 biphase bit times), the word in the last location of the FIFO is loaded into the encoder and prepared for transmission. If the FIFO was full, [TFF] will be de-asserted when the encoder is loaded, allowing an additional word to be loaded into the FIFO.

When the last word in the FIFO has been loaded into the encoder, [TFE] goes high, indicating that the FIFO is empty. To ensure the continuation of a multi-frame message, more data must then be loaded into the FIFO before the encoder starts the transmission of the last bit of the current frame (the frame parity bit for 3270, 3299, and 8-bit modes; the last of the three mandatory fill bits for 5250). This maximum load time from [TFE] can be calculated by subtracting two from the number of bits in each frame of the respective protocol, and multiplying that result by the bit rate. This number represents the best case time to load—the worst case value is dependent on CPU performance. Since the CPU samples the transceiver flags and interrupts at instruction boundaries, the CPU clock rate, wait states (from programmed wait states, asserting the WAIT pin, or remote access cycles), and the type of instruction currently being executed can affect when the flag or interrupt is first presented to the CPU.

If there is no further data to transmit (or if the load window is missed), the ending sequence (3270/3299/8-bit) is generated and the transmitter returns to idle, de-asserting TX-ACT and [TA]. In 5250 mode, the three required fill bits are sent and TX-ACT and [TA] are de-asserted at a time dependent on the value of bits 7 through 3 of the Auxiliary Transceiver Register {ATR}. If {ATR[7–3]} = 00000, TX-ACT and [TA] are de-asserted at the end of the third required fill bit resulting in no additional ''line hold'' at the end of the message. Each increment of {ATR[7–3]} results in an additional half bit time of line hold up to a maximum of 15.5 bit times.

Data should not be loaded into the FIFO after the transmitter is committed to ending the message and before the [TA] flag is deasserted. If this occurs, the load will be missed by the transmitter control logic and the word(s) will remain in the FIFO. This condition exists when [TA] and [TFE] are both low at the same time, and can be cleared by resetting the transceiver (asserting [TRES]) or by loading more data into the FIFO, in which case the first frame(s) transmitted will contain the word(s) left in the FIFO from the previous message.

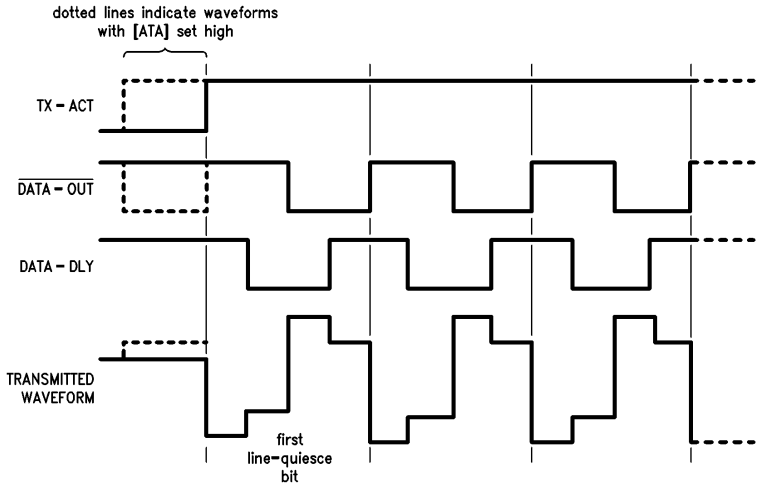


FIGURE 3-7. Transmitter Output

3.2.2 Receiver

The receiver accepts a serial biphase-encoded bit stream, strips off the framing information, checks for errors and reformats the data for parallel transfer to the CPU. The block diagram in *Figure 3-6* depicts the data flow from the serial input(s) to the FIFO's parallel outputs. Note that the FIFO outputs are multiplexed with the Error Code Register {ECR} outputs.

The receiver and transmitter share the same TCLK, though in the receiver this clock is used only to establish the sampling rate for the incoming biphase encoded data. All control timing is derived from a clock signal extracted from this data. Several status flags and interrupts are made available to the CPU to handle the asynchronous nature of the incoming data stream. See *Figure 3-8* for the timing relationships of these flags and interrupts relative to the incoming data.

The input source to the decoder can be either the on-chip analog line receiver, the DATA-IN input or the output of the transmitter (for on-chip loopback operation). Two bits, the Select Line Receiver [SLR] and Loopback [LOOP], control this selection. For interfacing to the on-chip analog line receiver, see Section 3.2.5.1, 3270 Line Interface. An example of an external comparator circuit for interfacing to twinax cable in 5250 environments is contained in Section 3.2.5.2, 5250 Line Interface. The selected serial data input can be inverted via the Receiver Invert [RIN] control bit.

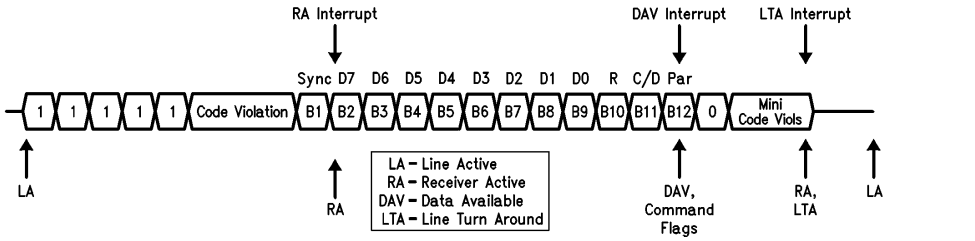
The receiver continually monitors the line, sampling at a frequency equal to eight times the expected data rate. The Line Active flag [LA] is asserted whenever an input transition is detected and will remain asserted as long as another input transition is detected within 16 TCLK cycles. If another transition is not detected in this time frame, [LA] will be de-asserted. The propagation delay from the occurrence of the edge to [LA] being set is approximately 1 transceiver clock cycle. This function is independent of the mode of operation of the transceiver; [LA] will continue to respond to input signal transitions, even if the transmitter is activated and the receiver disabled.

If the receiver is not disabled by the transmitter or by asserting [TRES], the decoder will adjust its internal timing to the incoming transitions, attempting to synchronize to valid biphase-encoded data. When synchronization occurs, the biphase clock will be extracted and the serial NRZ (Non-Return to Zero) data will be analyzed for a valid start sequence, see *Figure 3-2(b)*. The minimum number of line quiesce bits required by the receiver logic is selectable via the Receiver Line Quiesce [RLQ] control bit. If this bit is set high (the power-up condition), three line quiesce bits are required; if set low, only two are needed. Once the start sequence has been recognized, the receiver asserts the Receiver Active flag [RA] and enables the error detection circuitry. The propagation delay from the occurrence of the mid-bit edge of the sync bit in the starting sequence to [RA] being set is approximately 3 transceiver clock cycles.

The NRZ serial bit stream is now clocked into a serial to parallel shift register and analyzed according to the expected data pattern as defined by the protocol. If no errors are detected by the word parity bit, the parallel data (up to a total of 11-bits, depending on the protocol) is passed to the first location of the FIFO. It then propagates asynchronously to the last location in approximately 40 ns, at which time the Data Available flag [DAV] is asserted, indicating to the CPU that valid data is available in the FIFO. The propagation delay from the occurrence of the mid-bit edge of the parity bit of the frame to [DAV] being set is approximately 5 transceiver clock cycles.

Of the possible 11-bits in the last location of the FIFO, 8-bits (data byte) are mapped into {RTR} and the remaining bits (if any) are mapped into the Transceiver Status Register {TSR [2–0]}. The CPU accesses the data byte by reading {RTR}, and the 5250 address field or 3270 control bits by reading {TSR}. When reading the FIFO, it is important to note that {TSR} must be read before {RTR}, since reading {RTR} advances the FIFO. Once [DAV] has been recognized as set by the CPU, the data can be read by any instruction with {RTR] as the source. All instructions with {RTR} as the source (except BIT, CMP, JRMK, JMP reg-

3.0 Transceiver (Continued)



TL/F/9336–46

FIGURE 3-8. Timing of Receiver Flags Relative to Incoming Data

ister, LJMP conditional, and LCALL conditional) will result in popping the last location of the FIFO, presenting a new word (if present) for future CPU access. Data in the FIFO will propagate from one location to the next in approximately 10–15 ns, therefore the CPU is easily able to unload the FIFO with a set of consecutive instructions.

If the received bit stream is a multi-byte message, the receiver will continue to process the data and load the FIFO. After the third load (if the CPU has not accessed the FIFO), the Receive FIFO Full flag [RFF] will be asserted. The propagation delay from the occurrence of the mid-bit edge of the parity bit of the frame to [RFF] being set is approximately 5 transceiver clock cycles. If there are more than 3 frames in the incoming message, the CPU has approximately one frame time (sync bit to start of parity bit) to start unloading the FIFO. Failure to do so will result in an overflow error condition and a resulting loss of data (see Receiver Errors).

If there are no errors detected, the receiver will continue to process the incoming frames until the end of message is detected. The receiver will then return to an inactive state, clearing [RA] and asserting the Line Turn-Around flag, [LTA] indicating that a message was received with no errors. The propagation delay from the occurrence of the edge starting the first minicode violation to [RA] cleared and [LTA] set is approximately 17 transceiver clock cycles in 3270, 3299, and 8-bit modes. In 5250 modes, the assertion of [LTA] and clearing of [RA] are dependent on how the transmission line ends after the transmission of the three required fill bits (see 5250 Modes). For the 3270 and 3299 protocols, [LTA] can be used to initiate an immediate transmitter FIFO load; for the other protocols, an appropriate response delay time may be needed. [LTA] is cleared by loading the transmitter's FIFO, writing a one to [LTA] in the Network Command flag register, or by asserting [TRES].

Receiver Errors

If the Receiver Active flag, [RA], is asserted by the receiver logic, the selected receiver input source is continuously checked for errors, which are reported to the CPU by asserting the Receiver Error flag, [RE], and setting the appropriate receiver error flag in the Error Code Register {ECR}. If a condition occurs which results in multiple errors being created, only the first error detected will be latched into {ECR}. Once an error has been detected and the appropriate error flag has been set, the receiver is disabled, clearing [RA] and preventing the Line Turn-Around flag and interrupt [LTA] from being asserted. The Line Active flag [LA] remains asserted if signal transitions continue to be detected on the input.

5 error flags are provided in {ECR}:

7	6	5	4	3	2	1	0
rsv	rsv	rsv	OVF	PAR	IES	LMBT	RDIS

[OVF] **Overflow**—Asserted when the decoder writes to the first location of the FIFO while [RFF] is asserted. The word in the first location will be over-written; there will be no effect on the last two locations.

[PAR] **Parity Error**—Asserted when a received frame fails an even (word) parity check.

[IES] **Invalid Ending Sequence**—Asserted during an expected end sequence when an error occurs in the mini code-violation. Not valid in 5250 modes.

[LMBT] **Loss of Mid-Bit Transition**—Asserted when the expected biphase-encoded mid-bit transition does not occur within the expected window. Indicates a loss of receiver synchronization.

[RDIS] **Receiver Disabled While Active**—Asserted when an active receiver is disabled by the transmitter being activated.

To determine which error has occurred, the CPU must read {ECR}. This is accomplished by asserting the Select Error Codes control bit, [SEC], and reading {RTR}. The {ECR} is only 5 bits wide, therefore the upper 3 bits are still the output of the receive FIFO (see *Figure 3-6)*. All instructions with {ECR} as the source (except BIT, CMP, JRMK, JMP register, LJMP conditional, and LCALL conditional) will clear the error condition and return the receiver to idle, allowing the receiver to again monitor the incoming data stream for a new start sequence. The [SEC] control bit must be de-asserted to read the FIFO's data from {RTR}.

If data is present in the FIFO when the error occurs, the Data Available flag [DAV] is de-asserted when the error is detected and re-asserted when {ECR} is read. Data present in the FIFO before the error occurred is still available to the CPU. The flexibility is provided, therefore, to read the error type and still recover data loaded into the FIFO before the error occurred. The Transceiver Reset, [TRES] can be asserted at any time, clearing both Transceiver FIFOs and the error flags.

3.0 Transceiver (Continued)

3.2.3 Transceiver Interrupts

The transceiver has access to 3 CPU interrupt vectors, one each for the transmitter and receiver, and a third, the Line Turn-Around interrupt, providing a fast turn around capability between receiver and transmitter. The receiver interrupt is the CPU's highest priority interrupt (excluding NMI), followed by the transmitter and Line Turn-Around interrupts, respectively. The three interrupt vector addresses and a full description of the interrupts are given in Table 3-2.

The receiver interrupt is user-selectable from 4 possible sources (only 3 used at present) by specifying a 2-bit field, the Receiver Interrupt Select bits [RIS1-0] in the Interrupt Control Register {ICR}. A full description is given in Table 3-3.

The RFF + RE interrupt occurs only when the receive FIFO is full (or an error is detected). If the number of frames in a received message is not exactly divisible by 3, one or two words could be left in the FIFO at the end of the message, since the CPU would receive no indication of the presence of that data, it is recommended that this interrupt be used together with the line turn-around interrupt, whose service routine can include a test for whether any data is present in the receive FIFO.

For additional information concerning interrupts, refer to Sections 2.1.1.3, Interrupt Control Registers, and 2.2.3, Interrupts.

3.2.4 Protocol Modes

3270/3299 Modes

As shown in Table 3-1, the transceiver can operate in 4 different 3270/3299 modes, to accommodate applications of the BCP in different positions in the network. The 3270 mode is designed for use in a device or a controller which is not in a multiplexed environment. For a multiplexed network, the 3299 multiplexer and controller modes are designed for each end of the controller to multiplexer connection, the 3299 repeater mode being used for an in-line repeater situated between controller and multiplexer.

For information on how parallel data loaded into the transmit FIFO and unloaded from the receive FIFO maps into the serial bit positions, see *Figure 3-9*.

To transmit a frame, {TCR [3–0]} must first be set up with the correct control information, after which the data byte can be written to {RTR}. The resulting composite 12-bit word is loaded into the transmit FIFO where it propagates through to the last location to be loaded into the encoder and formatted for transmission.

When formatting a 3270 frame, {TCR [2]} controls whether the transmitter is required to format a data frame or a command frame. If {TCR [2]} is low, the transmitter logic calcu-

TABLE 3-2. Transceiver Interrupts

Interrupt	Vector Address	Description
Receiver	000100	User selectable from 4 possible sources, see Table 3-3.
Transmitter	001000	Set when [TFE] asserted, indicating that the transmit FIFO is empty, cleared by writing to {RTR}. **Note:** [TRES] causes [TFE] to be asserted.
Line Turn-Around	001100	Set when a valid end sequence is detected, cleared by writing to {RTR}, writing a one to [LTA], or asserting [TRES]. In 5250 modes, interrupt is set when the last fill bit has been received and no further input transitions are detected. Will not be set in 5250 or 8-bit non-promiscuous modes unless an address match was received.

The interrupt vector is obtained by concatenating {IBR} with the vector address as shown:

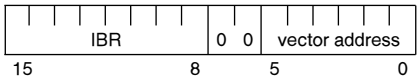


TABLE 3-3. Receiver Interrupts

Interrupt	RIS1,0	Description
RFF + RE	0 0	Set when [RFF] or [RE] asserted. If activated by [RFF], indicating that the receive FIFO is full, interrupt is cleared by reading from {RTR}. If activated by [RE], indicating that an error has been detected, interrupt is cleared by reading from {ECR}.
DAV + RE	0 1	Set when [DAV] or [RE] asserted. If activated by [DAV], indicating that valid data is present in the receive FIFO, interrupt is cleared by reading from {RTR}. If activated by [RE], indicating that an error has been detected, interrupt is cleared by reading from {ECR}.
Not Used	1 0	Reserved for future product enhancement.
RA	1 1	Set when [RA] asserted, indicating the receipt of a valid start sequence, cleared by reading {ECR} or {RTR}.

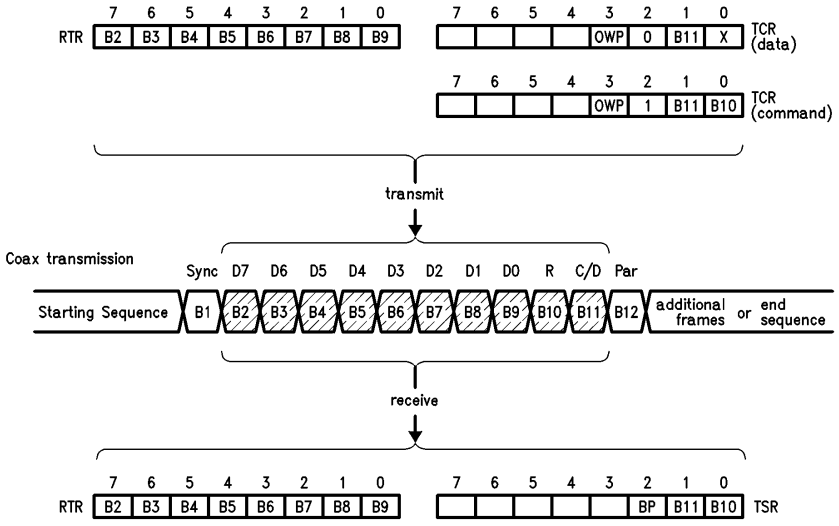
All receiver interrupts can be cleared by asserting [TRES].

3.0 Transceiver (Continued)

lates odd parity on the data byte (B2–B9) and transmits this value for B10. If {TCR [2]} is high, B10 takes the state of {TCR [0]}. Odd Word Parity [OWP] controls the type of parity calculated on B1–B11 and transmitted as B12, the frame delimiter. If [OWP] is high, odd parity is output; otherwise even parity is transmitted. In this manner the system designer is provided with maximum flexibility in defining the transmitted 3270 control bits (B10–B12).

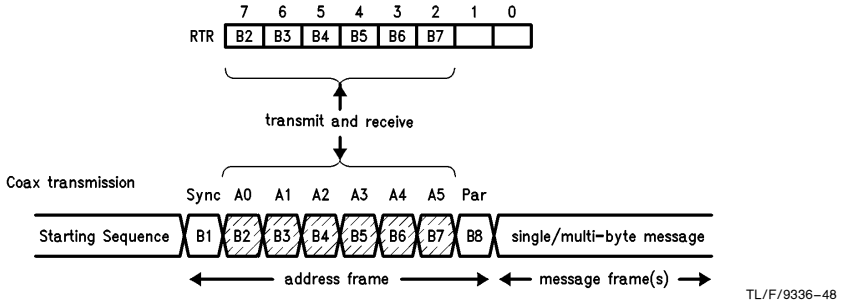
When data is written to {RTR}, the least significant 4 bits of {TCR} are loaded into the FIFO along with the data being written to {RTR}. The same {TCR} contents can therefore be used for more than one frame of a multi-frame transmission, or changed for each frame.

When a 3270 frame is received and decoded, the decoder loads the parallel data into the receive FIFO where it propagates through to the last location and is mapped into {RTR} and {TSR}. Bits B2–B11 are exactly as received; Byte Parity [BP] is odd parity on B2–B9, calculated in the decoder. Reading {RTR} will advance the receive FIFO, therefore {TSR} must be read first if this information is to be utilized.



TL/F/9336–47

(a) 3270 Data and Command Frames



TL/F/9336–48

(b) 3299 Address Frame

FIGURE 3-9. 3270/3299 Frame Assembly/Disassembly Procedure

3.0 Transceiver (Continued)

When formatting a 3299 address frame, the procedure is the same as for a 3270 frame, with {RTR [7–2]} defining the address to be transmitted. The only bit in {TCR} which has any functional meaning in this mode is [OWP], which controls the type of parity required on B1–B8. Similarly, when the receiver de-formats a 3299 address frame, the received address bits are loaded into {RTR [7–2]}; {RTR [1–0]} and {TSR [2–0]} are undefined.

The POLL, POLL/ACK and TT/AR flags in the Network Command Flag Register are valid only in 3270 and 3299 (excluding the 3299 address frame) modes. These flags are decodes of their respective coax commands as defined in Table 3-4. The Data Error or Message End [DEME] flag (also in the {NCF} register) indicates different information depending on the selected protocol. In 3270 and 3299, [DEME] is set when B10 of the received frame does not match the locally generated odd parity on bits B2–B9 of the received frame. [DEME] is not part of the receiver error logic, it functions only as a status flag to the CPU. These flags are decoded from the last location in the FIFO and are valid only when [DAV] is asserted; they are cleared by reading {RTR} and must be checked before advancing the receiver FIFO.

5250 Modes

The biphase data is inverted in the 5250 protocol relative to 3270/3299 (see the Protocol section—IBM 5250). Depending on the external line interface circuitry, the transceiver's biphase inputs and outputs may need to be inverted by asserting the [RIN] (Receiver INvert) and [TIN] (Transmitter INvert) control bits in {TMR}.

For information on how data must be organized in {TCR} and {RTR} for input to the transmitter, and how data extracted from a received frame is organized by the receiver and mapped into {TSR} and {RTR}, see *Figure 3-10*.

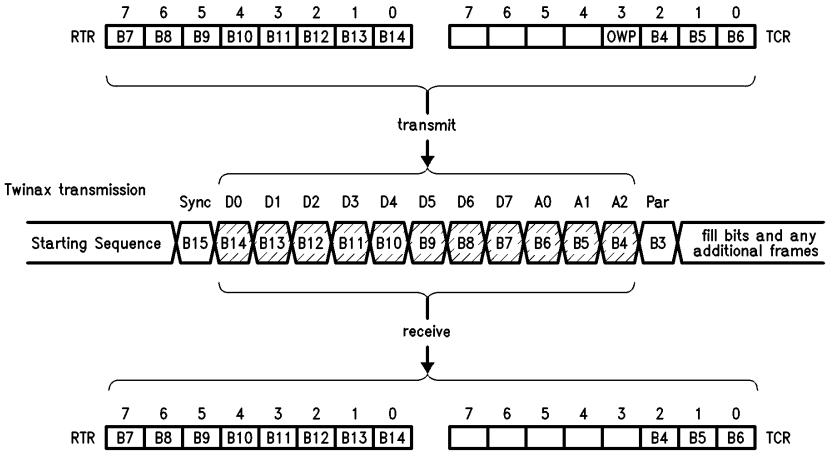
To transmit a 5250 message, the least significant 4 bits of {TCR} must first be set up with the correct address and parity control information. The station address field (B4–B6) is defined by {TCR[2–0]}, and [OWP] controls the type of parity (even or odd) calculated on B4–B15 and transmitted as B3. When the 8-bit data byte is written to {RTR}, the resulting composite 12-bit word is loaded into the transmit FIFO, starting the transmitter. The same {TCR} contents can be used for more than one frame of a multi-frame transmission, or changed for each frame.

The 5250 protocol defines bits B0–B2 as fill bits which the transmitter automatically appends to the parity bit (B3) to

TABLE 3-4. Decode of 3270 Coax Commands

Received Word										Flag	Description
B2	**B3**	**B4**	**B5**	**B6**	**B7**	**B8**	**B9**	**B10**	**B11**		
0	0	0	0	0	0	0	0	0	0	RAR	TT/AR (Clean Status) Received
X	X	X	1	0	0	0	1	X	1	ACK	POLL/ACK Command Received
X	X	X	0	0	0	0	1	X	1	POLL	POLL Command Received

All flags cleared by reading {RTR}.



TL/F/9336–49

FIGURE 3-10. 5250 Frame Assembly/Disassembly Description

3.0 Transceiver (Continued)

form the 16-bit frame. Additional fill bits may be inserted between frames of a multi-frame transmission by loading the fill bit register, {FBR}, with the one's complement of the number of fill bits to be transmitted. A value of FF (hex), corresponds to the addition of no extra fill bits. At the conclusion of a message the transmitter will return to the idle state after transmitting the 3 fill bits of the last frame (no additional fill bits will be transmitted).

As shown in Table 3-1, the transceiver can operate in 2 different 5250 modes, designated "promiscuous" and "non-promiscuous". The transmitter operates in the same manner in both modes.

In the promiscuous mode, the receiver passes all received data to the CPU via the FIFO, regardless of the station address. The CPU must determine which station is being addressed by reading {TSR [2–0]} before reading {RTR}.

In the non-promiscuous mode, the station address field (B4–B6) of the first frame must match the 3 least significant bits of the Auxiliary Transceiver Register, {ATR [2–0]}, before the receiver will pass the data on to the CPU. If no match is detected in the first frame of a message, and if no errors were found on that frame, the receiver will reset to idle, looking for a valid start sequence. If an address match is detected in the first frame of a message, the received data is passed on to the CPU. For the remainder of the message all received frames are decoded in the same manner as the promiscuous mode.

To maintain maximum flexibility, the receiver logic does not interpret the station address or command fields in determining the end of a 5250 message. The message typically ends with no further line transitions after the third fill bit of the last frame. This end of message must be distinguished from a loss of synchronization between frames of a multi-byte transmission condition by looking for line activity some time after the loss of synchronization occurs. When the loss of synchronization occurs during fill bit reception, the receiver monitors the Line Active flag, [LA], for up to 11 biphase bit times (11 μs at the 1 MHz data rate). If [LA] goes inactive at any point during this period, the receiver returns to the idle state, de-asserting [RA] and asserting [LTA]. If, however, [LA] is still asserted at the end of this window, the receiver interprets this as a real loss of synchronization and flags the [LMBT] error condition to the CPU. (See Receiver Errors in this section.)

In the 5250 modes, the Data-Error-or-Message-End [DEME] flag is a decode of the 111 station address (the end of message delimiter) and is valid only when [DAV] is asserted. This function allows the CPU to quickly determine when the end of message has been received.

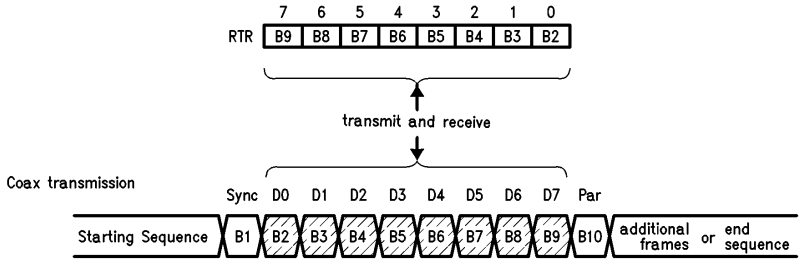
The transmitter has the flexibility of holding TX-ACT active at the end of a 5250 message, thus reducing line reflections and ringing during this critical time period. The amount of hold time is programmable from 0 μs to 15.5 μs in 500 ns increments (assuming TCLK is 8 MHz), and is set by writing the selected value to the upper 5-bits of the Auxiliary Transceiver Register, {ATR [7–3]}.

General Purpose 8-Bit Modes

As shown in Table 3-1, the transceiver can operate in 2 different 8-bit modes, designated "promiscuous" and "non-promiscuous". In the non-promiscuous mode, the first frame data byte (B2–B9) must match the contents of {ATR[7–0]} before the receiver will load the FIFO and assert [DAV]. If no match is made on the first frame, and if no errors were found on that frame, the receiver will go back to idle, looking for a valid start sequence. The address comparator logic is not enabled in the promiscuous mode, and therefore all received frames are passed through the receive FIFO to the CPU. The transmitter operates in the same manner in both modes.

The serial bit positions relative to the parallel data loaded into the transmit FIFO and presented to the CPU by the receiver FIFO are shown in *Figure 3-11*. To transmit a frame, the data byte is written to {RTR}, loading the transmit FIFO where it propagates through to the last location to be loaded into the encoder and formatted for transmission. Only [OWP] in {TCR} is loaded into the transmitter FIFO in both protocol modes; {TCR [2–0]} are don't cares. B10 is defined by a parity calculation on B1–B9; odd if [OWP] is high and even if [OWP] is low.

When a frame is received, the decoder loads the processed data into the receive FIFO where it propagates through to the last location and is mapped into {RTR}. All bits are exactly as received. Reading the data is accomplished by reading {RTR}. {TSR [2–0]} are undefined in the 8-bit modes.



TL/F/9336–50

FIGURE 3-11. General Purpose 8-Bit Frame Assembly/Disassembly Procedure

3.0 Transceiver (Continued)

3.2.5 Line Interface

3.2.5.1 3270 Line Interface

In the 3270 environment, data is transmitted between a control unit and a device via a single coax cable or twisted pair cable. The coax type is RG62AU with a maximum length of 1.5 kilometers. The twisted pair cable has become more prevalent to reduce cabling and routing costs. Typically, a 24 AWG unshielded twisted pair is used to achieve the cost reduction goals. The length of the twisted pair cable is a minimum of 100 feet to a maximum of 900 feet. The 3270 protocol utilizes a transformer to isolate the peripheral from the cabling system.

An effective line interface design must be able to accept either coax or twisted pair cabling and compensate for noise, jitter and reflections in the cabling system. There must be an adequate amount of jitter tolerance to offset the effects of filtering and noise. Some filtering is needed to reduce ambient noise caused by surrounding hardware. Such filtering must not introduce transients that the receiver comparator translates into data jitter.

An effective driver design should also attempt to compensate for the filtering effects of the cable. Higher data frequencies become attenuated more than lower frequency signals as cable length is increased, yielding greater disparity in the amplitudes of these signals. This effect generates greater jitter at the receiver. The 3270 signal format allows for a high voltage (predistorted) magnitude and a low voltage (nondistorted) magnitude within each data bit time. Increasing the predistorted-to-nondistorted signal level ratio counteracts the filtering phenomenon because the lower frequency signals contain less predistortion than do higher frequency signals. Thus, the amplitude of the higher frequency signals is ''boosted'' more than the lower frequency signals. Unfortunately, a low signal level is more susceptible to reflection-induced errors at short cable length. Proper impedance matching and slower edge rates must be utilized to eliminate as much reflection as possible at these lengths.

Additionally, shielded or balanced operation must be adequately supported. Shielded operation implies the use of coax cable, where balanced implies the use of twisted pair cable. Proper termination should be employed, and a termination slightly greater than the characteristic impedance of theline may actually provide more desirable waveforms than a perfectly matched termination. Board layout should make the comparator lines as short as possible. Lines should be placed closely together to avoid the introduction of differential noise. These lines should not pass near ''noisy'' lines. A ground plane should isolate all ''noisy'' lines.

BCP Design

The line interface design for the receiver is shown in *Figure 3-12.* An offset of approximately 17 mV separates the comparator inputs, making the receiver more immune to ambient noise present on the circuit board. A 2:1:1 (arranged as a 3:1) transformer increases any voltage sensitivity lost by introducing the offset. A bandpass filter is employed to reduce edge rate to the comparator and eliminate ambient noise. The bandwidth (30 kHz to 30 MHz) was chosen to provide sufficient attenuation for noise while producing minimum data jitter.

The driver design, *Figure 3-13,* incorporates a National Semiconductor DS3487 and a resistor network to generate the proper signal levels. The predistorted-to-nondistorted ratio was chosen to be about 3 to 1. The coax/twisted pair front end, *Figure 3-14,* includes an ADC brand connector to switch between coax and twisted pair cable. The coax interface has the shield capacitively coupled to ground. The 510Ω resistor and the filter loading produce a termination of about 95Ω. The twisted pair interface balances both lines and possesses an input impedance of about 100Ω. This termination is somewhat higher than the characteristic impedance (about 96Ω) of twisted pair. Terminations of this type produce reflections that do not tend to generate mid-bit errors. Such terminations have the benefit of creating a larger voltage at the receiver over longer cable lengths. For a more detailed explanation of the 3270 line interface, see Application Note ''A Combined Coax/Twisted Pair 3270 Line Interface for the DP8344 Biphase Communications Processor''.

3.2.5.2 5250 Line Interface

The 5250 environment utilizes twinax in a multi-drop configuration, where eight devices can be ''daisy-chained'' over a total distance of 5,000 feet and eleven splices, (each physical device is considered a splice). Twinax connectors are bulky and expensive, but are very sturdy. Twinaxial cable is a shielded twisted pair that is nearly ⅓ of an inch thick.

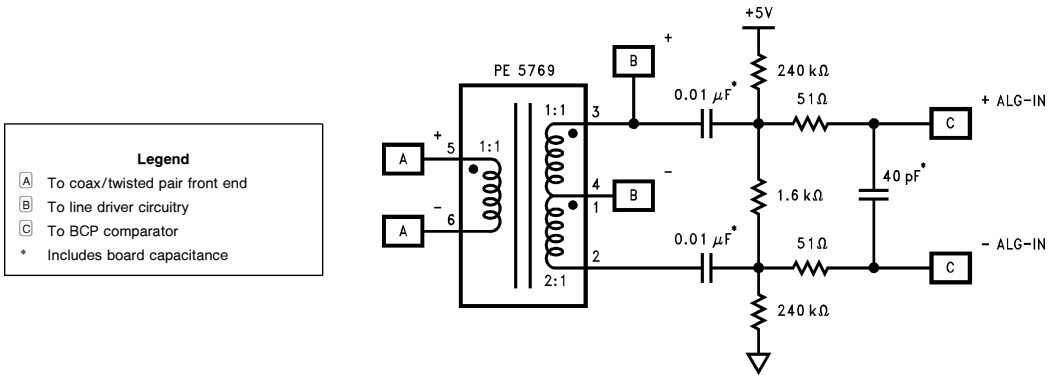
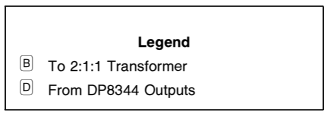
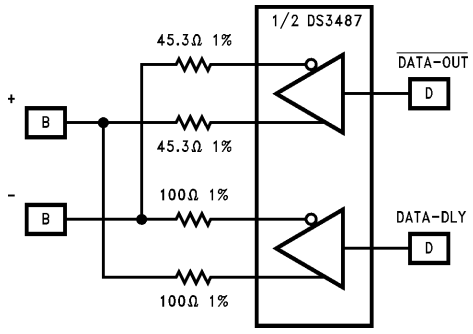


FIGURE 3-12. BCP Receiver Design

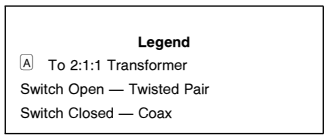
3.0 Transceiver (Continued)

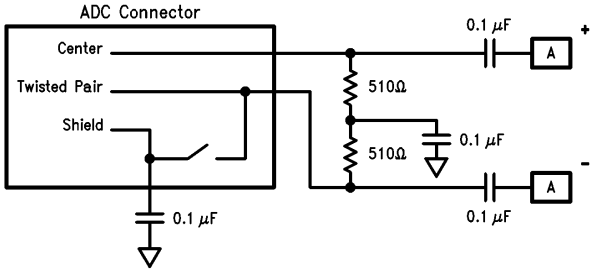




TL/F/9336–G2

FIGURE 3-13. BCP Driver Design





TL/F/9336–G3

FIGURE 3-14. BCP Coax/Twisted Pair Front End

The cable shield must be continuous throughout the transmission system, and be grounded at the system unit and each station. Since twinax connectors have exposed metal connected to their shield grounds, care must be taken not to expose them to noise sources. The polarity of the two inner conductors must also be maintained throughout the transmission system.

The transmission system is implemented in a balanced current mode; every receiver/transmitter pair is directly coupled to the twinax at all times. Data is impressed on the transmission line by unbalancing the line voltage with the driver current. The system requires passive termination at both ends of the transmission line. The termination resistance value is given by:

$R_t = Z_{O/2}$; where

R_t: Termination Resistance

Z_O: Characteristic Impedance

In practice, termination is accomplished by connecting both conductors to the shield via 54.9Ω, 1% resistors; hence the characteristic impedance of the twinax cable of 107Ω ±5% at 1.0 MHz. Intermediate stations must not terminate the line; each is configured for ''pass-through'' instead of ''terminate'' mode. Stations do not have to be powered on to pass twinax signals on to other stations; all of the receiver/transmitter pairs are DC coupled. Consequently, devices must never output any signals on the twinax line during power-up or down that could be construed as data, or interfere with valid data transmission between other devices.

Driver Circuits for the DP8344B

The transmitter interface on the DP8344B is sufficiently general to allow use in 3270, 5250, and 8-bit transmission systems. Because of this generality, some external hardware is needed to adapt the outputs to form the signals necessary to drive the twinax line. The chip provides three signals: $\overline{\text{DATA-OUT}}$, DATA-DLY and TX-ACT. $\overline{\text{DATA-OUT}}$ is biphase serial data (inverted). DATA-DLY is the biphase serial data output (non-inverted) delayed one-quarter bit-time. TX-ACT, or transmitter active, signals that serial data is being transmitted when asserted. $\overline{\text{DATA-OUT}}$ and DATA-DLY can be used to form the A and B phase signals with their three levels by the circuit shown in *Figure 3-15.* TX-ACT is used as an external transmitter enable. The BCP can invert the sense of the $\overline{\text{DATA-OUT}}$ and DATA-DLY signals by asserting [TIN] {TMR[3]}. This feature allows both 3270 and 5250 type biphase data to be generated, and/or utilization of inverting on non-inverting transmitter stages.

Drivers for the 5250 environment may not place any signals on the transmission system when not activated. The power-on and off conditions of drivers must be prevented from causing noise on the system since other devices may be in operation. *Figure 3-15* shows a ''DC power good'' signal enabling the driver circuit. This signal will lock out conduction in the drivers if the supply voltage is out of tolerance.

Twinax signals can be viewed as consisting of two distinct phases, phase A and phase B, each with three levels, off,

3.0 Transceiver (Continued)

high and low. The off level corresponds with 0 mA current being driven, the high level is nominally 62.5 mA, +20% −30%, and the low level is nominally 12.5 mA, +20% −30%. When these currents are applied to a properly terminated transmission line the resultant voltages impressed at the driver are: off level is 0V, low level is 0.32V ±20%, high level is 1.6V ±20%. The interface must provide for switching of the A and B phases and the three levels. A bimodal constant current source for each phase can be built that has a TTL level interface for the BCP.

Receiver Circuits

The pseudo-differential mode of the twinax signals make receiver design requirements somewhat different than the coax 3270 world. Hence, the analog receiver on the BCP is not well suited to receiving twinax data. The BCP provides both analog inputs to an on-board comparator circuit as well as a TTL level serial data input, DATA-IN. The sense of this serial data can be inverted by the BCP by asserting [RIN], {TMR[4]}.

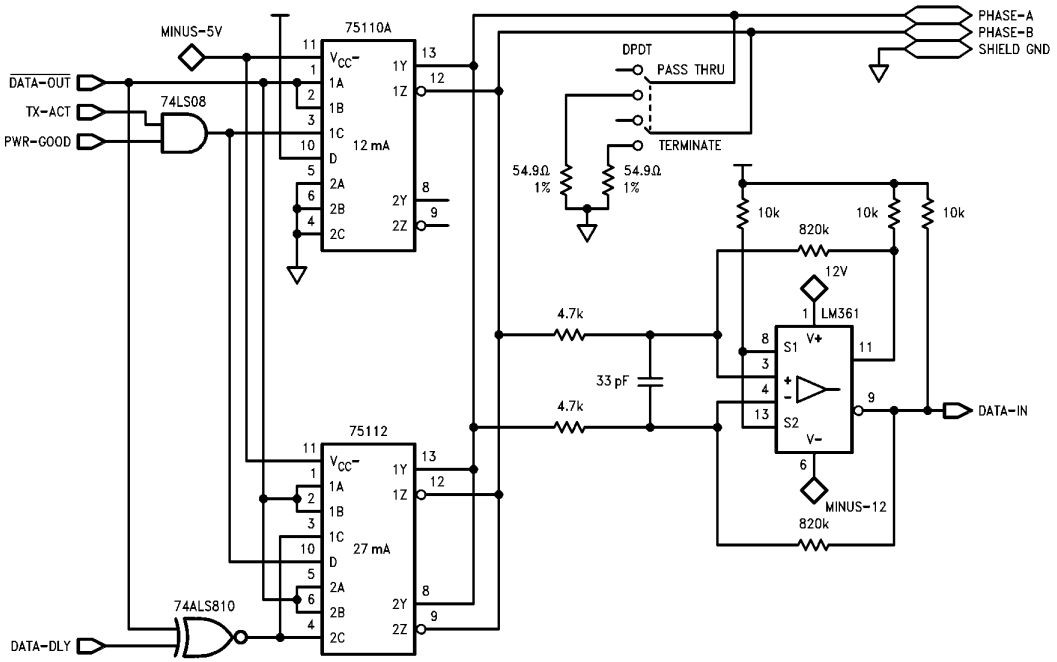
The external receiver circuit must be designed with care to ensure reliable decoding of the bit-stream in the worst environment. Signals as small as 100 mV must be detected. In order to receive the worst case signals, the input level switching threshold or hysteresis for the receiver should be nominally 29 mV ±20%. This value allows the steady state, worst case signal level of 100 mV ±66% of its amplitude before transitioning.

To achieve this, a differential comparator with complementary outputs can be applied, such as the National LM361. The complementary outputs are useful in setting the hysteresis or switching threshold to the appropriate levels. The LM361 also provides excellent common mode noise rejection and a low input offset voltage. Low input leakage current allows the design of an extremely sensitive receiver, without loading the transmission line excessively.

In addition to good analog design techniques, a low pass filter with a roll-off of approximately 1 MHz should be applied to both the A and B phases. This filter essentially conducts high frequency noise to the opposite phase, effectively making the noise common mode and easily rejectable.

Layout considerations for the LM361 include proper bypassing of the ±12V supplies at the chip itself, with as short as possible traces from the pins to 0.1 μF ceramic capacitors. Using surface mount chip capacitors reduces lead inductance and is therefore preferable in this case. Keeping the input traces as short and even in length is also important. The intent is to minimize inductance effects as well and standardize those effects on both inputs. The LM361 should have as much ground plane under and around it as possible. Trace widths for the input signals especially should be as wide as possible; 0.1 inch is usually sufficient. Finally, keep all associated discrete components nearby with short routing and good ground/supply connections.

For a more detailed explanation of the 5250 line interface, see application note "Interfacing the DP8344 to Twinax."



TL/F/9336–G4

FIGURE 3-15. 5250 Line Interface Schematic

4.0 Remote Interface and Arbitration System (RIAS)

INTRODUCTION

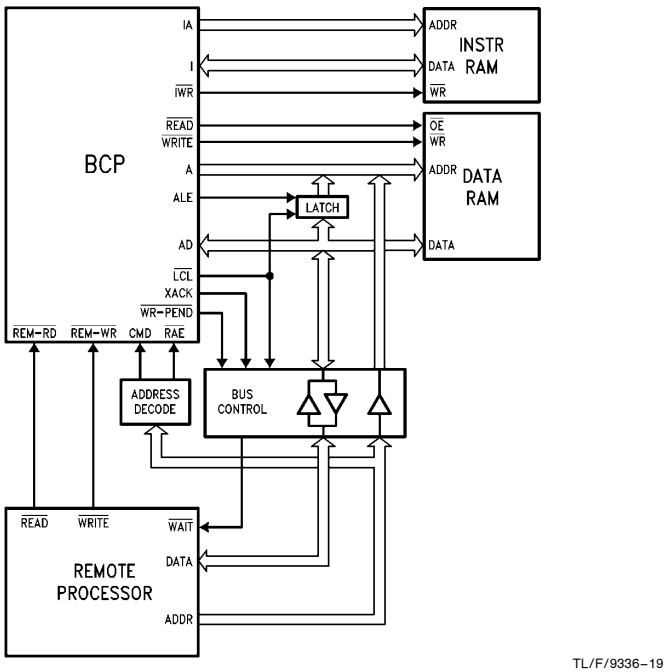
Communication with the BCP is based on the BCP's ability to share its data memory. A microprocessor (or any intelligent device) can read and write to any BCP data location while the BCP CPU is executing instructions. This capability is part of the BCP's Remote Interface and Arbitration System (RIAS). Sharing data memory is possible because RIAS's arbitration logic allocates use of the BCP's data and address buses. RIAS has been designed so that accesses of BCP data memory by another device minimally impact its performance as well as the BCP's. In addition to data memory accesses, RIAS allows another device to control how BCP programs are loaded, started and debugged.

4.1 RIAS ARCHITECTURAL DESCRIPTION

Interfacing to the BCP is accomplished with the control signals listed in Table 4-1. *Figure 4-1* shows the BCP interfaced to Instruction Memory, Data Memory, and an intelligent device, termed the Remote Processor (RP). Instruction and Data are separate memory systems with separate address buses and data paths. This arrangement allows continuous instruction fetches without interleaved data accesses. Instruction Memory (IMEM) is interfaced to the BCP through the Instruction (I) and Instruction Address (IA) buses. IMEM is 16 bits wide and can address up to 64k memory. Data Memory (DMEM) is eight bits wide and can also address up to 64k memory. The DMEM address is formed by the 8-bit upper byte (A bus) and the 8-bit lower byte (AD bus). The AD bus must be externally latched because it also serves as the path for data between the BCP and DMEM. For further information on how AD bus is used, refer to Section 2.2.2 CPU Timing.

The Remote Processor's address and data buses are connected to the BCP's address and data buses through the bus control circuitry. The RP's address lines decode a chip select for the BCP called Remote Access Enable ($\overline{\text{RAE}}$). Basically, the BCP's Data Memory has been memory mapped into the RP's memory. A Remote Access of the BCP occurs when $\overline{\text{REM-RD}}$ or $\overline{\text{REM-WR}}$, along with $\overline{\text{RAE}}$ is asserted low. $\overline{\text{REM-RD}}$ and $\overline{\text{REM-WR}}$ can be directly connected to the Remote Processor's read and write lines, or for more complicated systems the $\overline{\text{REM-RD}}$ and $\overline{\text{REM-WR}}$ signals may be controlled by a combination of address decode and the RP's read and write signals. To the RP, an access of the BCP will appear as any other memory system access. This configuration allows the RP to read and write Data Memory, read and write the BCP's Program Counter, and read and write BCP Instruction Memory. These functions are selected by control bits in the Remote Interface Configuration register {RIC}. This register can be accessed only by the RP and not by the BCP CPU. If the Remote Processor executes a remote access with the Command input (CMD) high, {RIC} is accessed through the BCP's AD bus.

In *Figure 4-1*, the Remote Processor's address lines are decoded to form the CMD input. When a remote access takes place with CMD low, the memory system designated in {RIC} is accessed. *Figure 4-2* shows the contents of {RIC}. The two least significant bits are the Memory Select bits [MS1–0] which designate the type of remote access: to Data Memory, the Program Counter, or Instruction Memory. This register also contains the BCP start bit [STRT], three interface select bits [FBW, LR, LW], the Single-Step bit [SS], and the Bi-directional Interrupt Status bit [BIS]. Refer to the RIAS Reference Section for a more detailed description of the contents of this register and the function of each bit.



TL/F/9336–19

FIGURE 4-1. BCP/Remote Processor Interface

TABLE 4-1. RIAS Inputs and Outputs

Signal	In/Out	Pin	Reset State	Function
CMD	In	45	X	**Co**mman**D** input. When high, remote accesses are directed to the Remote Interface Configuration register, {RIC}. When low, remote accesses are directed to Data Memory, Instruction Memory or the Program Counter as determined by {RIC [1,0]}.
$\overline{\text{LCL}}$	Out	31	0	**LoCaL**. Normally low, goes high when the BCP relinquishes the data and address bus to service a remote access.
$\overline{\text{LOCK}}$	In	44	X	Asserting this input Low will **LOCK** out local (BCP) accesses to Data Memory. Once the remote processor has been granted the bus, $\overline{\text{LOCK}}$ gives it sole access to the bus and BCP accesses are ''waited''.
$\overline{\text{RAE}}$	In	46	X	**R**emote **A**ccess **E**nable. Setting this input low allows host access of BCP functions and memory.
$\overline{\text{REM-RD}}$	In	47	X	**REM**ote **R**ea**D**. When low along with $\overline{\text{RAE}}$, a remote read cycle is requested; serviced by the BCP when the data bus becomes available.
$\overline{\text{REM-WR}}$	In	48	X	**REM**ote **WR**ite. When low along with $\overline{\text{RAE}}$, a remote write cycle is requested; serviced by the BCP when the data bus becomes available.
$\overline{\text{WR-PEND}}$	Out	49	1	**WR**ite **PEND**ing. In a system configuration where remote write cycles are latched, $\overline{\text{WR-PEND}}$ will go low, indicating that the latches contain valid data which have yet to be serviced by the BCP.
XACK	Out	50	1	Transfer **ACK**nowledge. Normally high, goes low on $\overline{\text{REM-RD}}$ or $\overline{\text{REM-WR}}$ going low (if $\overline{\text{RAE}}$ low) returning high when the transfer is complete. Normally used as a ''wait'' signal to a remote processor. (In the Latched Write mode, XACK will only transition if a second remote access begins before the first one completes.)
$\overline{\text{WAIT}}$	In	54	X	Asserting this input low will add wait states to both remote accesses and to the BCP instruction cycle. **$\overline{\text{WAIT}}$** will extend a remote access until it is set high.

7	6	5	4	3	2	1	0	
BIS	SS	FBW	LR	LW	STRT	MS1	MS0	RIC

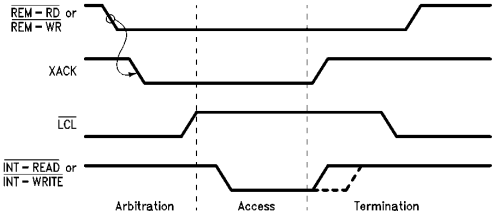
BIS —Bidirectional Interrupt Status
SS —Single-Step
FBW —Fast Buffered Write mode
LR —Latched Read mode
LW —Latched Write mode
STRT —BCP CPU start/stop
MS1–0 —Memory Selection

FIGURE 4-2. Remote Interface Control Register

4.1.1 Remote Arbitration Phases

The BCP CPU and RIAS share the internal CPU-CLK. This clock is derived from the X1 crystal input. It can be divided by two by setting [CCS] = 1 in {DCR} or run undivided by setting [CCS] = 0. The frequency at which the Remote Processor is run need not bear any relationship to the CPU-CLK. A remote access is treated as an asynchronous event and data is handshaked between the Remote Processor and the BCP.

The two key handshake signals involved in the BCP/RP interface are Transfer Acknowledge (XACK) and Local ($\overline{\text{LCL}}$). Internally, two more signals control the access timing: $\overline{\text{INT-READ}}$ and $\overline{\text{INT-WRITE}}$. The timing for a generic Remote Access is shown in *Figure 4-3*. A remote access is



TL/F/9336–20

FIGURE 4-3. Generic Remote Access ($\overline{\text{RAE}}$ = 0)

initiated by the RP asserting $\overline{\text{REM-RD}}$ or $\overline{\text{REM-WR}}$ with $\overline{\text{RAE}}$ low. There is no set-up/hold time relationship between $\overline{\text{RAE}}$ and $\overline{\text{REM-RD}}$ or $\overline{\text{REM-WR}}$. These signals are internally gated together such that if RAE (REM-RD + REM-WR) is true, a remote access will begin. A short delay later, XACK will fall. This signal can be fed back to the RP's wait line to extend its read or write cycle, if necessary. When the BCP's

4.0 Remote Interface and Arbitration System (RIAS) (Continued)

arbitration logic determines that the BCP is not using data memory, $\overline{LCL}$ rises, relinquishing control of the address and data buses to the RP. The remote access can be delayed at most one BCP instruction (providing [LOR] is not set high). If the CPU is executing a string of data memory accesses, RIAS has an opportunity to break in at the completion of every instruction. The time period between $\overline{REM\text{-}RD}$ or $\overline{REM\text{-}WR}$ being asserted (with $\overline{RAE}$ low) and $\overline{LCL}$ rising is called the Arbitration Phase. It is a minimum of one T-state, but can be increased if the BCP CPU is accessing Data Memory (local access) or if the BCP has set the Lock Out Remote bit [LOR].

The CMD pin is internally latched on the first falling edge of the CPU-CLK after a remote access has been initiated by asserting $\overline{RAE}$ low along with asserting $\overline{REM\text{-}RD}$ or $\overline{REM\text{-}WR}$ low. If the remote interface is asynchronous, the CMD signal must be valid simultaneously or before $\overline{RAE}$ is asserted low along with $\overline{REM\text{-}RD}$ or $\overline{REM\text{-}WR}$ being asserted low. The value of CMD is only sampled once during each remote access and will remain in effect for the duration of the remote access.

After the Arbitration Phase has ended, the Access Phase begins. Either Data Memory, Instruction Memory, the Program Counter, or {RIC} is read or written in this phase. Either $\overline{INT\text{-}READ}$ or $\overline{INT\text{-}WRITE}$ will fall one T-state after $\overline{LCL}$ rises. These two signals provide the timing for the different types of accesses. $\overline{INT\text{-}READ}$ times the transitions on the AD bus for Remote Reads and forms the external $\overline{READ}$ line. $\overline{INT\text{-}WRITE}$ clocks data into the PC and {RIC} and forms the $\overline{IWR}$ and $\overline{WRITE}$ lines. $\overline{INT\text{-}READ}$ and $\overline{INT\text{-}WRITE}$ rise with XACK, or shortly after.

The duration of the Access Phase depends on the type of memory being accessed. Data Memory and Instruction Memory accesses are subject to any programmed wait states and all remote accesses are waited by asserting $\overline{WAIT}$ low. The minimum time in the Access Phase is 2 T-states.

The rising edge of XACK indicates the Access Phase has ended and the Termination Phase has begun. If the RP was doing a read operation, this edge indicates that valid data is available to the RP. During the Termination Phase the BCP is regaining control of the buses. $\overline{LCL}$ falls one T-state after XACK and since the RP is no longer being waited, it can deassert $\overline{REM\text{-}RD}$ or $\overline{REM\text{-}WR}$. The duration of this phase is a minimum of one T-state, but can be extended depending on the interface mode chosen in {RIC}.

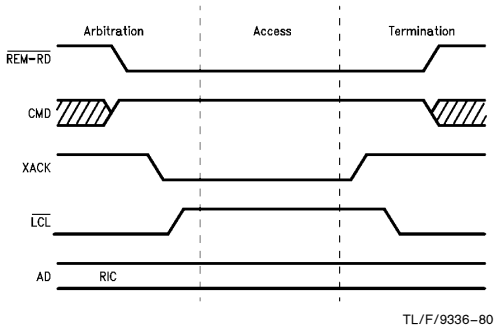
4.1.2 Access Types

There are four types of accesses an RP can make of the BCP:

—Remote Interface Control Register {RIC}
—Data Memory (DMEM)
—Program Counter (PC)
—Instruction Memory (IMEM)

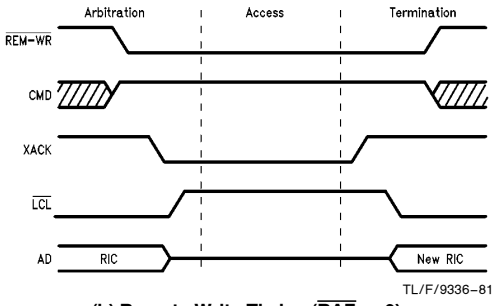
An access of {RIC} is accomplished by asserting $\overline{RAE}$ and $\overline{REM\text{-}RD}$ or $\overline{REM\text{-}WR}$ with the CMD pin asserted high. The Remote Interface Configuration register is accessed through the AD bus as shown in *Figure 4-4(c)*. A read or write of {RIC} can take place while the BCP CPU is executing instructions. Timing for this access is shown in *Figures 4-4(a)* and *(b)*. Note that in the Remote Read *Figure 4-4(a),* AD does not transition. This is because the contents of {RIC} are active on the bus by default. The AD bus is in TRI-STATE during a Remote Write *Figure 4-4(b)* while $\overline{LCL}$ is high. The byte being written to {RIC} is latched on the rising edge of XACK and can be seen on AD after $\overline{LCL}$ falls.

The Access Phase, in this case, is always two T-states (unless $\overline{WAIT}$ is low) because {RIC} is not subject to any programmed wait states.



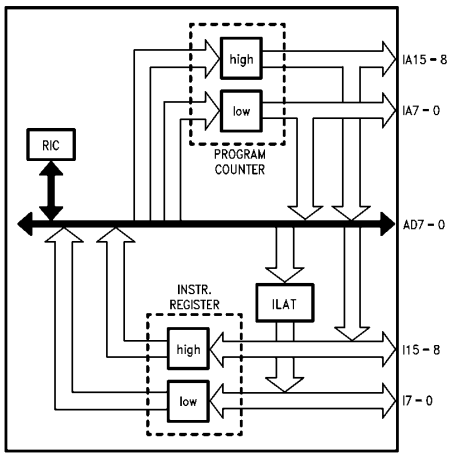
TL/F/9336–80

(a) Remote Read Timing ($\overline{RAE}$ = 0)



TL/F/9336–81

(b) Remote Write Timing ($\overline{RAE}$ = 0)



TL/F/9336–82

(c) RIC to AD Connectivity

FIGURE 4-4. Generic RIC Access

4.0 Remote Interface and Arbitration System (RIAS) (Continued)

Remote Accesses other than to {RIC} are accomplished with the CMD pin low in conjunction with asserting $\overline{RAE}$ low along with $\overline{REM\text{-}WR}$ or $\overline{REM\text{-}RD}$ being taken low. The type of access performed is defined by the Memory Select bits in {RIC}, as shown in *Figure 4-5*.

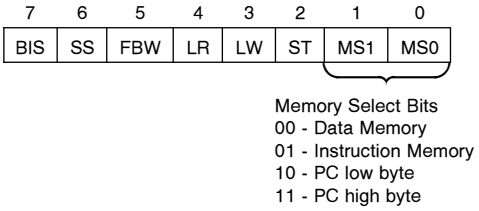


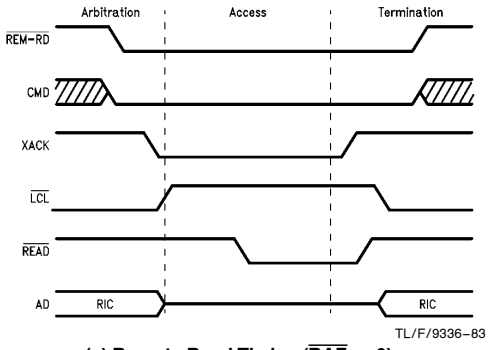
FIGURE 4-5. Memory Select Bits in {RIC}

Reads or writes of Data Memory (DMEM) are preceded by setting the Memory Select bits in {RIC} for a DMEM access: [MS1,0] = 00. After that, the RP simply reads or writes to BCP Data Memory as many times as it needs to. A DMEM access, as well as a {RIC} access, can be made while the BCP CPU is executing instructions. All other accesses must be executed with the BCP CPU stopped.

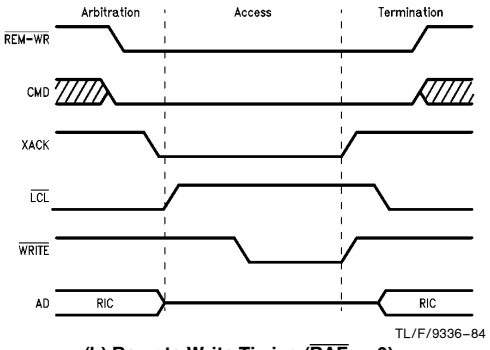
The timing for a Data Memory read and write are shown in *Figure 4-6*. The access is initiated by asserting $\overline{RAE}$ and $\overline{REM\text{-}RD}$ or $\overline{REM\text{-}WR}$ while CMD is low. The BCP responds by bringing its address and data lines into TRI-STATE and allowing the RP to control DMEM. $\overline{READ}$ is asserted in the Access Phase of a Remote Read *Figure 4-6(a)*. It will stay low for a minimum of one T-state, but can be extended by adding programmable data wait states or by taking $\overline{WAIT}$ low. $\overline{WRITE}$ is asserted in the Access Phase with a remote write. It too is a minimum of one T-state and can be increased by adding programmable wait states or by taking $\overline{WAIT}$ low.

Figure 4-7(c) shows the data path from the Program Counter to the AD bus. Both high and low PC bytes can be written or read through AD. The RP has independent control of the high and low bytes of the Program Counter—the byte being accessed is specified in the Memory Select bits. The high byte of the PC is accessed by setting [MS1–0] = 11. Setting [MS1–0] = 10 allows access to the low byte of the PC. After the Memory Select bits are set by a Remote Write to {RIC}, the byte selected can be read or written by the RP by executing a Remote Access with CMD low. Remote accesses to both the high and low bytes of the PC, as well as the instruction memory access must be executed with the BCP CPU idle. Four accesses by the RP are necessary to read or write both the high and low bytes of the PC. Timing for a PC access is shown in *Figure 4-7(a)* and *(b)*. The PC becomes valid on a Remote Read *(a)* one T-state after $\overline{LCL}$ rises and one T-state before XACK rises. AD is in TRI-STATE while $\overline{LCL}$ is high for a Remote Write *(b)*. Time in the Access Phase is two T-states if $\overline{WAIT}$ is not asserted.

Instruction memory (IMEM) is accessed through another internal path: from AD to the I bus, shown in *Figure 4-8(c)*. The memory is accessed first low byte, then high byte. Low and high bytes of the 16-bit I bus are alternately accessed for Remote Reads. An 8-bit holding register, ILAT, retains the low byte until the high byte is written by the Remote Processor for the write to IMEM. The BCP increments the PC after the high byte has been accessed.



(a) Remote Read Timing ($\overline{RAE}$ = 0)

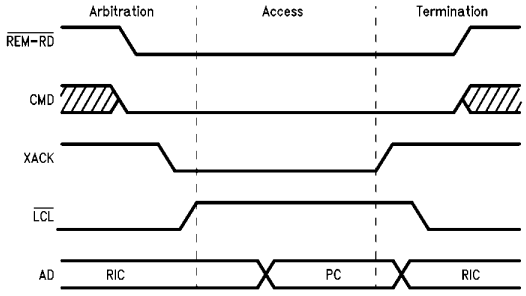


(b) Remote Write Timing ($\overline{RAE}$ = 0)

FIGURE 4-6. Generic DMEM Access

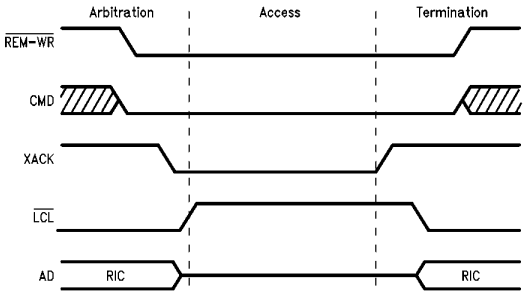
Timing for an IMEM access is shown in *Figure 4-8(a)* and *(b)*. As before, the Memory Select bits are first set to instruction memory: [MS1–0] = 01. It is only necessary to set [MS1–0] once for repeated IMEM accesses. (Instruction Memory is the power-up Memory Selection state.) A simple state machine keeps track of which instruction byte is expected next—low or high byte. The state machine powers up looking for the low instruction byte and every IMEM access causes this state machine to switch to the alternate byte. Accesses other than to IMEM will not cause the state machine to switch to the alternate byte, but writing 01 to the Memory Select bits in {RIC} (i.e. [MS1–0] = 01, pointing to IMEM) will always force the state machine to the ''low byte state''. This way the instruction word boundary can be reset without resetting the BCP. When the BCP is reset the state machine will also be forced to the ''low byte state.''

Figure 4-8(a) shows a Remote Read of Instruction memory. Both the low byte, then the high byte can be seen on back to back remote reads. An instruction byte becomes active on the AD bus one T-state after $\overline{LCL}$ rises and is valid when XACK rises. This time period will be a minimum of one T-state, but can be extended up to three more T-states by instruction wait states.



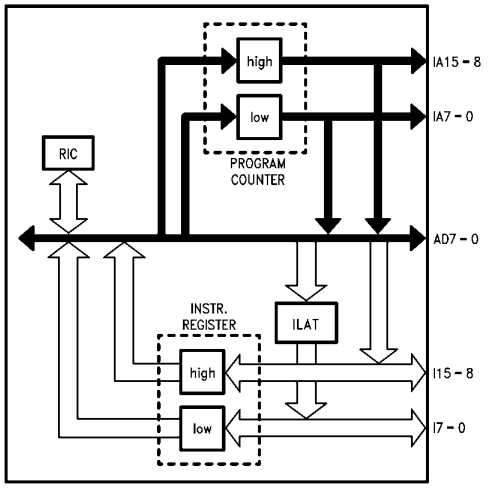
TL/F/9336–85

(a) Remote Read Timing ($\overline{RAE}$ = 0)



TL/F/9336–86

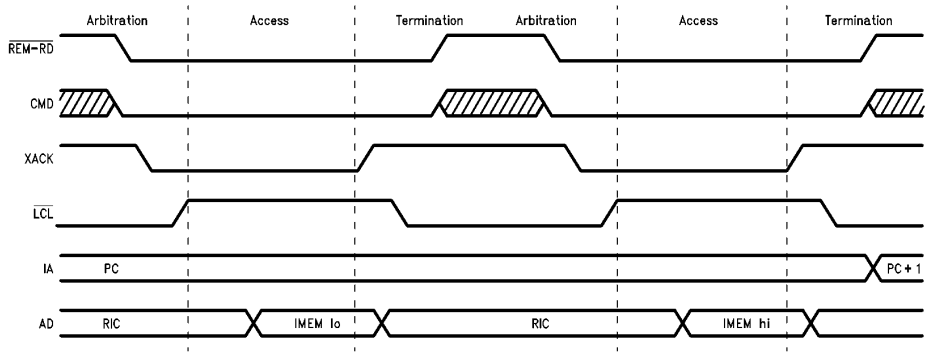
(b) Remote Write Timing ($\overline{RAE}$ = 0)



TL/F/9336–87

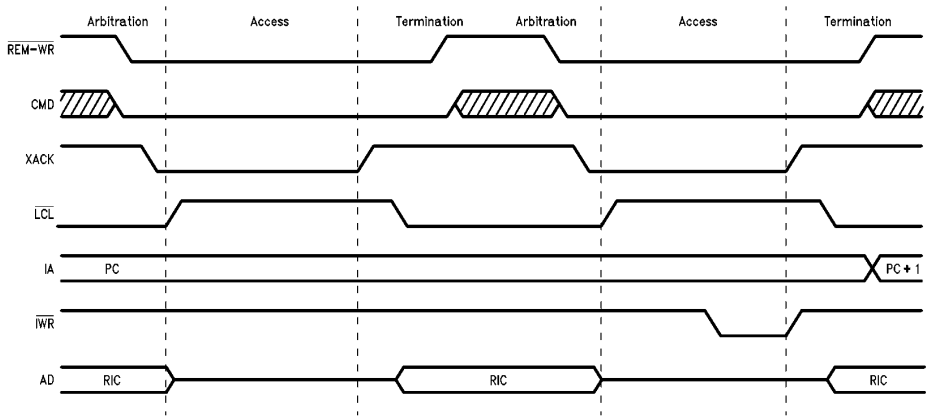
(c) IA to AD Connectivity

FIGURE 4-7. Generic PC Access



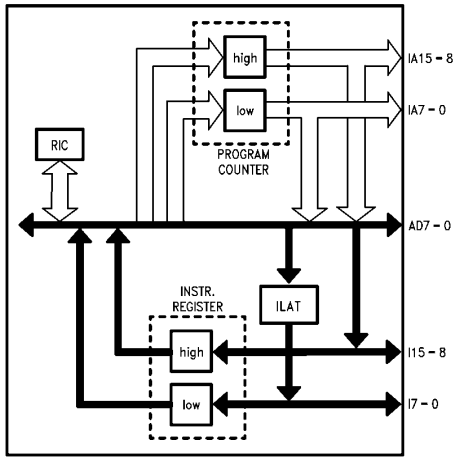
TL/F/9336–88

(a) Remote Read Timing ($\overline{RAE}$ = 0)



TL/F/9336–89

(b) Remote Write Timing ($\overline{RAE}$ = 0)



TL/F/9336–90

(c) I to AD Connectivity

FIGURE 4-8. Generic IMEM Access

4.0 Remote Interface and Arbitration System (RIAS) (Continued)

In addition, $\overline{WAIT}$ can delay the rising edge of XACK indefinitely. One T-state after XACK rises, {RIC} will once again be active on AD. Timing is similar for a Remote Write. AD is in TRI-STATE while $\overline{LCL}$ is high. $\overline{LCL}$ is asserted for a minimum of three T-states, but can be extended by instruction wait states and the $\overline{WAIT}$ pin. $\overline{IWR}$ clocks the instruction into memory during the write of the high byte. The Instruction Address (PC) is incremented about one T-state after $\overline{LCL}$ falls on a high byte access for both Remote Reads and Writes.

Soft-loading Instruction Memory is accomplished by first setting the BCP Program Counter to the starting address of the program to be loaded. The Memory Select bits are then set to IMEM. BCP instructions can then be moved from the Remote Processor to the BCP—low byte, high byte—until the entire program is loaded.

4.1.3 Interface Modes

The Remote Interface and Arbitration System will support TRI-STATE buffers or latches between the Remote Processor and the BCP. The choice between buffers and latches depends on the type of system that is being interfaced to. Latches will help prevent the faster system from slowing to the speed of the slower system. Buffers can be used if the Remote Processor (RP) requires that data be handshaked between the systems.

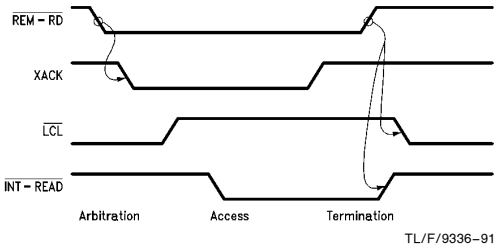
Figure 4-9 shows the timing of Remote Reads via a buffer *(a)* and a latch *(b)* (called a Buffered Read and Latched Read). The main difference in these modes is in the Termination Phase. The Buffered Read handshakes the data back to the RP. When the BCP deasserts XACK, data is valid and the RP can deassert $\overline{REM\text{-}RD}$. Only after $\overline{REM\text{-}RD}$ goes high is $\overline{LCL}$ removed. In the Latched Read *Figure 4-9(b)* XACK rises at the same time, but the Termination Phase completes without waiting for the rising edge of $\overline{REM\text{-}RD}$. One half T-state after XACK rises, $\overline{INT\text{-}READ}$ rises and one half T-state later $\overline{LCL}$ falls. The BCP can use the buses one T-state after $\overline{LCL}$ falls. The minimum time (no wait states, no arbitration delay) the BCP CPU could be prevented from using the bus is four T-states in the Latched Read Mode.

A Buffered Read prevents the BCP CPU from using the bus during the time RP is allocated the buses. This time period begins when $\overline{LCL}$ rises and ends when $\overline{REM\text{-}RD}$ is removed. If the $\overline{REM\text{-}RD}$ is asserted longer than the minimum Buffered Read execution time (four T-states), then the BCP may be unnecessarily prevented from using the buses. Therefore, if there are no overriding reasons to use the Buffered Read Mode, the Latched Read Mode is preferable.

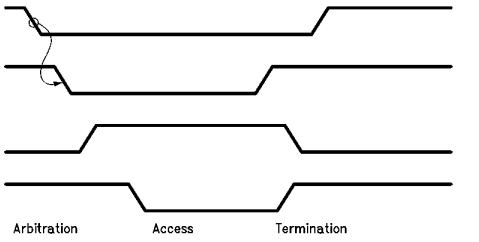
There are three Remote Write Modes—two require buffers and one requires latches. The timing for the writes utilizing buffers is shown in *Figure 4-10*. The Slow Buffered Write *(a)* is handshaked in the same manner as the Buffered Read and thus has the same timing. The Fast Buffered Write has similar timing to the Latched Read. This timing similarity exists because the BCP terminates the remote access without waiting for the RP to deassert $\overline{REM\text{-}WR}$.

In both cases, XACK falls a short delay after $\overline{REM\text{-}WR}$ falls and $\overline{LCL}$ rises when the RP is given the buses. One T-state after $\overline{LCL}$ rises, $\overline{INT\text{-}WRITE}$ falls. The termination in the Slow Buffered Write mode keys off $\overline{REM\text{-}WR}$ rising, as shown in *Figure 4-10(a)*. $\overline{INT\text{-}WRITE}$ rises a prop-delay later and $\overline{LCL}$ falls one T-state later. The Fast Buffered Write, shown in *Figure 4-10(b)*, begins the Termination Phase with the rising edge of XACK. $\overline{INT\text{-}WRITE}$ rises at the same time as XACK, and $\overline{LCL}$ falls one T-state later. The BCP can begin a local access one T-state after $\overline{LCL}$ transitions.

A Fast Buffered Write is preferable to the Slow Buffered Write if RP's write cycles are slow compared to the minimum Fast Buffered Write execution time. The Fast Buffered Write assumes, though, that data is available to the BCP by the time $\overline{INT\text{-}WRITE}$ rises.

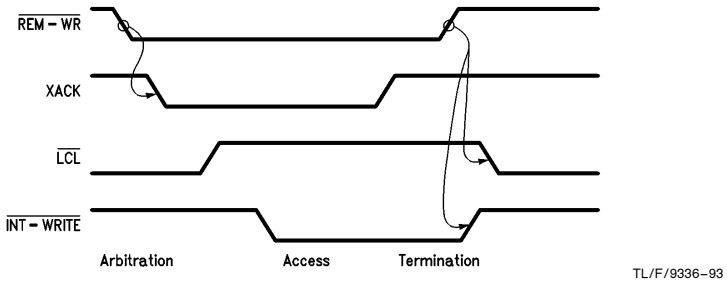


(a) Buffered Read

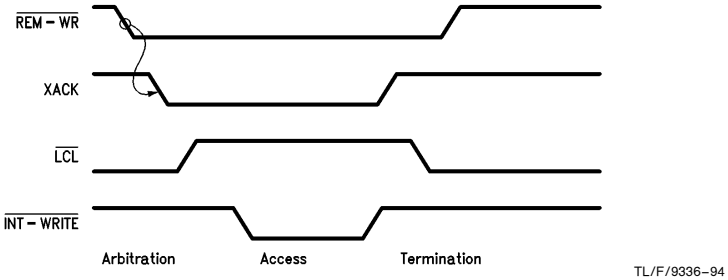


(b) Latched Read

FIGURE 4-9. Read from Remote Processor



(a) Slow Buffered Write



(b) Fast Buffered Write

FIGURE 4-10. Buffered Write from Remote Processor

In both Buffered Write Modes, XACK is asserted to wait the RP. The Latched Write Mode makes it possible for the RP to write to the BCP without getting waited. The timing for the Latched Write Mode is shown in *Figure 4-11*. When the Remote Processor writes to the BCP, its address and data buses are externally latched on the rising edge of $\overline{\text{REM-WR}}$. Even though $\overline{\text{REM-WR}}$ has been asserted XACK does not switch. The BCP only begins remote access execution after the trailing edge of $\overline{\text{REM-WR}}$. Since the RP is not requesting data back from the BCP, it can continue execution without waiting for the BCP to complete the remote access. After $\overline{\text{REM-WR}}$ is deasserted, $\overline{\text{WR-PEND}}$ is taken low to prevent overwrite of the latches. A minimum of two T-states later $\overline{\text{LCL}}$ switches and AD, A, and the external address latch go into TRI-STATE, allowing the latches which contain the remote address and data to become active. If the RP attempts to initiate another access before the current write is complete, XACK is taken low to wait the RP and the address and the data are safe because $\overline{\text{WR-PEND}}$ prevents the latches from opening. The Access Phase ends when $\overline{\text{INT-WRITE}}$ rises and the data is written. One T-state later, $\overline{\text{LCL}}$ falls and one T-state after that $\overline{\text{WR-PEND}}$ rises. If another access is pending, it can begin in the next T-state. This is indicated by XACK rising when $\overline{\text{WR-PEND}}$ rises.

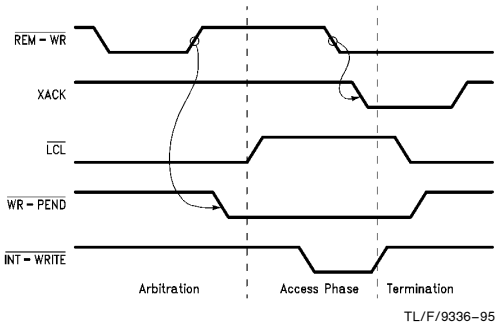
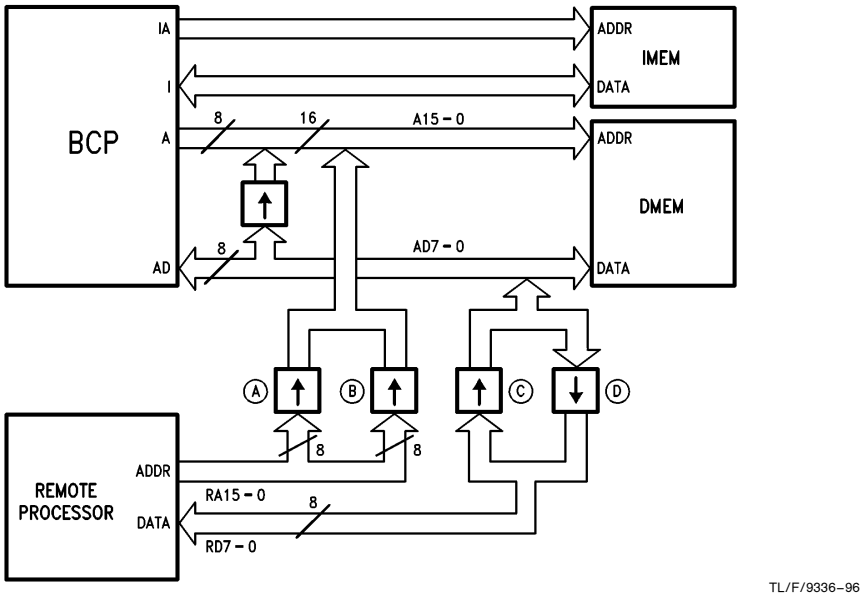


FIGURE 4-11. Latched Write from Remote Processor

A minimum BCP/RP interface utilizes four TRI-STATE buffers or latches. A block diagram of this interface is shown in *Figure 4-12*. The blocks A, B, C, and D indicate the location of buffers or latches. Blocks A and B isolate 16 bits of the RP's address bus from the BCP's Data Address bus. Two more blocks, C and D, bidirectionally isolate 8 bits of the RP's data bus from the BCP AD bus.

4.0 Remote Interface and Arbitration System (RIAS) (Continued)



TL/F/9336–96

FIGURE 4-12. Minimum BCP/Remote Processor Interface

The BCP Remote Arbitrator State Machine (RASM) must know what hardware interfaces to the RP in order to time the remote accesses correctly. To accomplish this, three Interface Mode bits in {RIC} are used to define the hardware interface. These bits are the Latched Write bit [LW], the Latched Read bit [LR] and the Fast Buffered Write bit [FBW]. See *Figure 4-13*.

7	6	5	4	3	2	1	0
BIS	SS	FBW	LR	LW	ST	MS1	MS0

Interface Mode Bits

– 0 – - Buffered Read

– 1 – - Latched Read

0 – 0 - Slow Buffered Write

1 – 0 - Fast Buffered Write

X – 1 - Latched Write

FIGURE 4-13. Interface Mode Bits

All combinations of Remote Reads or Writes with buffers or latches can be configured via the Interface Mode bits. A Buffered Read is accomplished by using a buffer for block D and setting [LR] = 0. Conversely, using a latch for block D and setting [LR] = 1 configures the RASM for Latched Reads. Using buffers for blocks A, B, and C and setting [LW] = 0 allows either a Slow or Fast Buffered Write. Setting [FBW] = 0 configures RASM for a Slow Buffered Write and [FBW] = 1 designates a Fast Buffered Write. A Latched Write is accomplished by using latches for blocks A, B, and C and setting [LW] = 1.

4.1.4 Execution Control

The BCP can be started and stopped in two ways. If the BCP is not interfaced to another processor, it can be started by pulsing $\overline{\text{RESET}}$ low while both $\overline{\text{REM-RD}}$ and $\overline{\text{REM-WR}}$ are low. Execution then begins at location zero. If there is a Remote Processor interfaced to the BCP, a write to {RIC} which sets the start bit [STRT] high will begin execution at the current PC location. Writing a zero to [STRT] stops execution after the current instruction is completed. A Single-Step is accomplished by writing a one to the Single-Step bit [SS] in {RIC}. This will execute the instruction at the current PC, increment the PC, and then return to idle. [SS] returns low after the single-stepped instruction has completed. [SS] is a write only bit and will always appear low when {RIC} is read.

Two pins ($\overline{\text{WAIT}}$ and $\overline{\text{LOCK}}$), and one register bit, [LOR], can also affect the BCP CPU or RIAS execution. The $\overline{\text{WAIT}}$ pin can be used to add wait states to a remote access. When $\overline{\text{WAIT}}$ must be asserted low to add wait states is dependent on which remote access mode is being used. The information needed to calculate when $\overline{\text{WAIT}}$ must be asserted to add wait states, is contained within the individual descriptions of the modes in the next section (4.2 RIAS Functional Description).

Programmed wait states delay when $\overline{WAIT}$ must be asserted since programmed wait states are inserted before $\overline{WAIT}$ is tested to see if any more wait states should be added. $\overline{LOCK}$ prevents local accesses of Data Memory. If $\overline{LOCK}$ is asserted a half T-state before T1 of a BCP instruction cycle, further local accesses will be prevented by waiting the Timing Control Unit. The Timing Control Unit (TCU) is the BCP CPU sub system responsible for timing each instruction. For a more detailed description of the operation of $\overline{LOCK}$, refer to the CPU Timing section. [LOR] allows the BCP to prevent remote accesses. Once [LOR], located in {ACR}, is set high, further remote accesses are waited by XACK remaining low.

Though the BCP CPU runs independently of RIAS there is some interaction between the two systems. [LOR] is one such interaction. In addition, two bits allow the BCP CPU to keep track of remote accesses. These bits are the Remote Write bit [RW] and the Remote Read bit [RR], and are located in {CCR[6–5]}. Each bit goes high when its respective remote access to DMEM reaches its Termination Phase. Once one of these bits has been set, it will remain high until a ''1'' is written to that bit to reset it low.

4.2 RIAS FUNCTIONAL DESCRIPTION

In this section, the operation of the Remote Arbitration State Machine (RASM), is described in detail. Discussed, among other things, are the sequence of events in a remote access, arbitration of the data buses, timing of external signals, when inputs are sampled, and when wait states are added. Each of the five Interface Modes is described in functional state machine form. Although each interface mode is broken out in a separate flow chart, they are all part of a single state machine (RASM). Thus the first state in each flow chart is actually the same state.

The functional state machine form is similar to a flow chart, except that transitions to a new state (states are denoted as rectangular boxes) can only occur on the rising edge of the internal CPU clock (CPU-CLK). CPU-CLK is high during the first half of its cycle. A state box can specify several actions, and each action is separated by a horizontal line. A signal name listed in a state box indicates that that pin will be asserted high when RASM has entered that state. Signals not listed are assumed low.

Note: This sometimes necessitates using the inversion of the external pin name.

This same rule applies to the A and AD buses. By default, these buses are active. The A bus will have the upper byte of the last used data address. The AD bus will display {RIC}. When one of these buses appears in a state box, the condition specified will be in effect only during that state. Decision blocks are shown as diamonds and their meaning is the same as in a flow chart. The hexagon box is used to denote a conditional state—not synchronous with the clock. When the path following a decision block encounters a conditional state, the action specified inside the hexagon box is executed immediately.

Also provided is a memory arbitration example in the form of a timing diagram for each of the five modes. These examples show back to back local accesses punctuated by a remote access. Both the state of RASM and the Timing Control Unit are listed for every clock at the top of each timing diagram. The RASM states listed correspond to the flow charts. The Timing Control Unit states are described in Section 2.2.2, Timing portion of the data sheet.

4.2.1 Buffered Read

The unique feature of this mode is the extension of the read until $\overline{REM\text{-}RD}$ is deasserted high. The complete flow chart for the Buffered Read mode is shown in *Figure 4-14.* Until a Remote Read is initiated (RAE*REM-RD true), the state machine (RASM) loops in state RS_{A1}. If a Remote Read is initiated and [LOR] is set high, RASM will move to state RS_{A2}. Likewise, if a Remote Read is initiated while the buses have been granted locally (i.e., Local Bus Request = 1), RASM will move to state RS_{A2}. The state machine will loop in state RS_{A2} as long as [LOR] is set high or the buses are granted locally. If the BCP CPU needs to access Data Memory while in either RS_A state (and $\overline{LOCK}$ is high), it can still do so. A local access is requested by the Timing Control Unit asserting the Local Bus Request (LCL-BREQ) signal. A local bus grant will be given by RASM if the buses are not being used (as is the case in the RS_A states).

XACK is taken low as soon as RAE*REM-RD is true, regardless of an ongoing local access. If [LOR] is low, RASM will move into RS_B on the next clock after RAE*REM-RD is true and there is no local bus request. No further local bus requests will be granted until the remote access is complete and RASM returns to RS_A. Half a T-state after entering RS_B the A bus (and AD bus if the access is to Data Memory) goes into TRI-STATE.

On the next CPU-CLK, RASM enters RS_C and $\overline{LCL}$ is taken high while XACK remains low. The wait state counters, i_{IW} and i_{DW}, are loaded in this state from [IW1–0] and [DW2–0], respectively, in {DCR}. The A bus (and AD if the access is to Data Memory) remains in TRI-STATE and the Access Phase begins.

The state machine can move into one of several states, depending on the state of CMD and [MS1–0], on the next clock. XACK remains low and $\overline{LCL}$ remains high in all the possible next states. If CMD is high, the access is to {RIC} and the next state will be RS_{D1}. Since the default state of AD is {RIC}, it will not transition in this state.

The five other next states all have CMD low and depend on the Memory Select bits. If [MS1–0] is 10 or 11 the state machine will enter either RS_{D2} or RS_{D3} and the low or high bytes of the Program Counter, respectively, will be read.

[MS1–0] = 00 designates a Data Memory access and moves RASM into RS_{D4}. $\overline{READ}$ will be asserted in this state and A and AD continue to be in TRI-STATE. This allows the Remote Processor to drive the Data Memory address for the read. Since DMEM is subject to wait states, RS_{D4} is looped upon until all the wait states have been inserted.

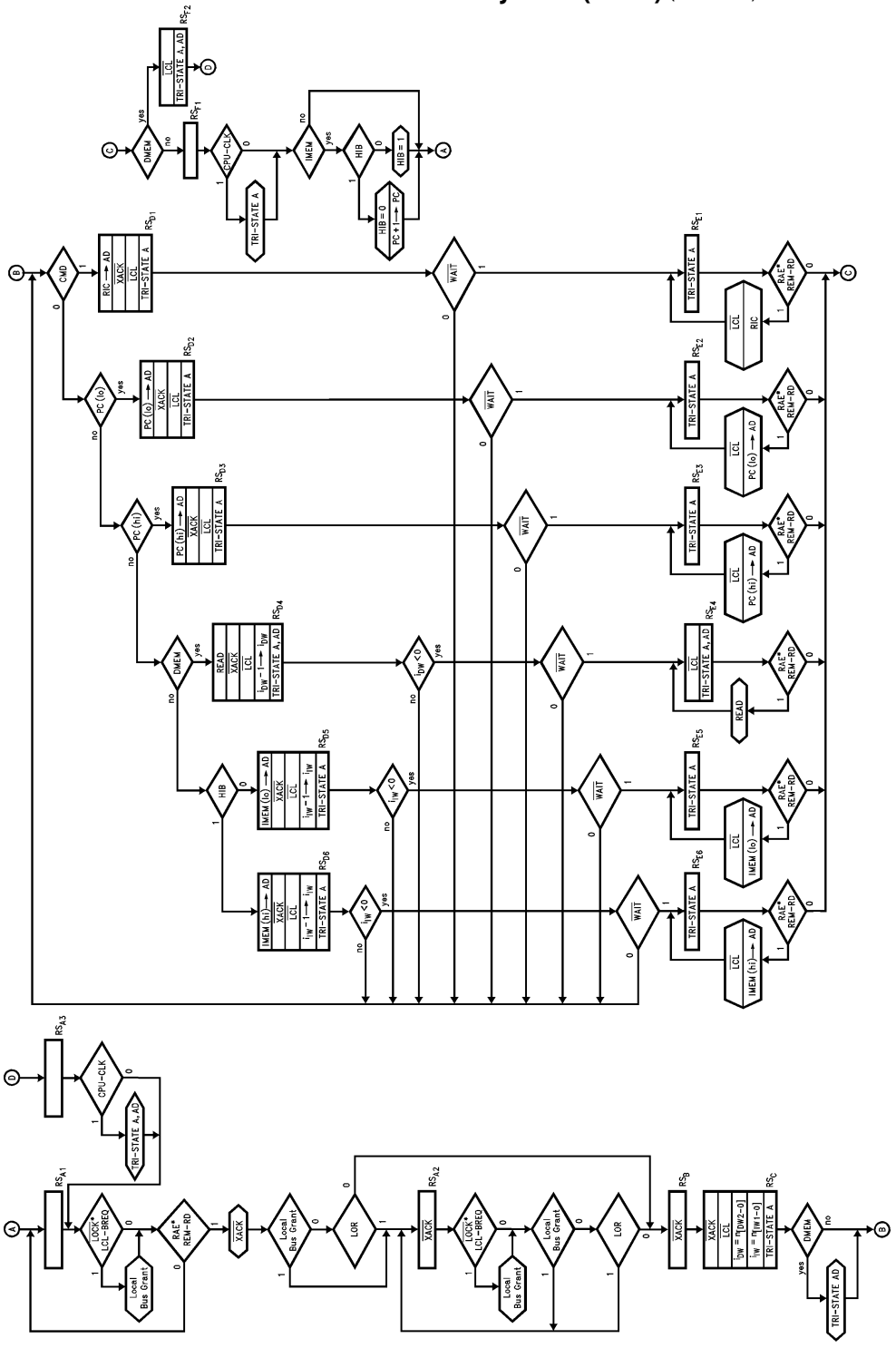
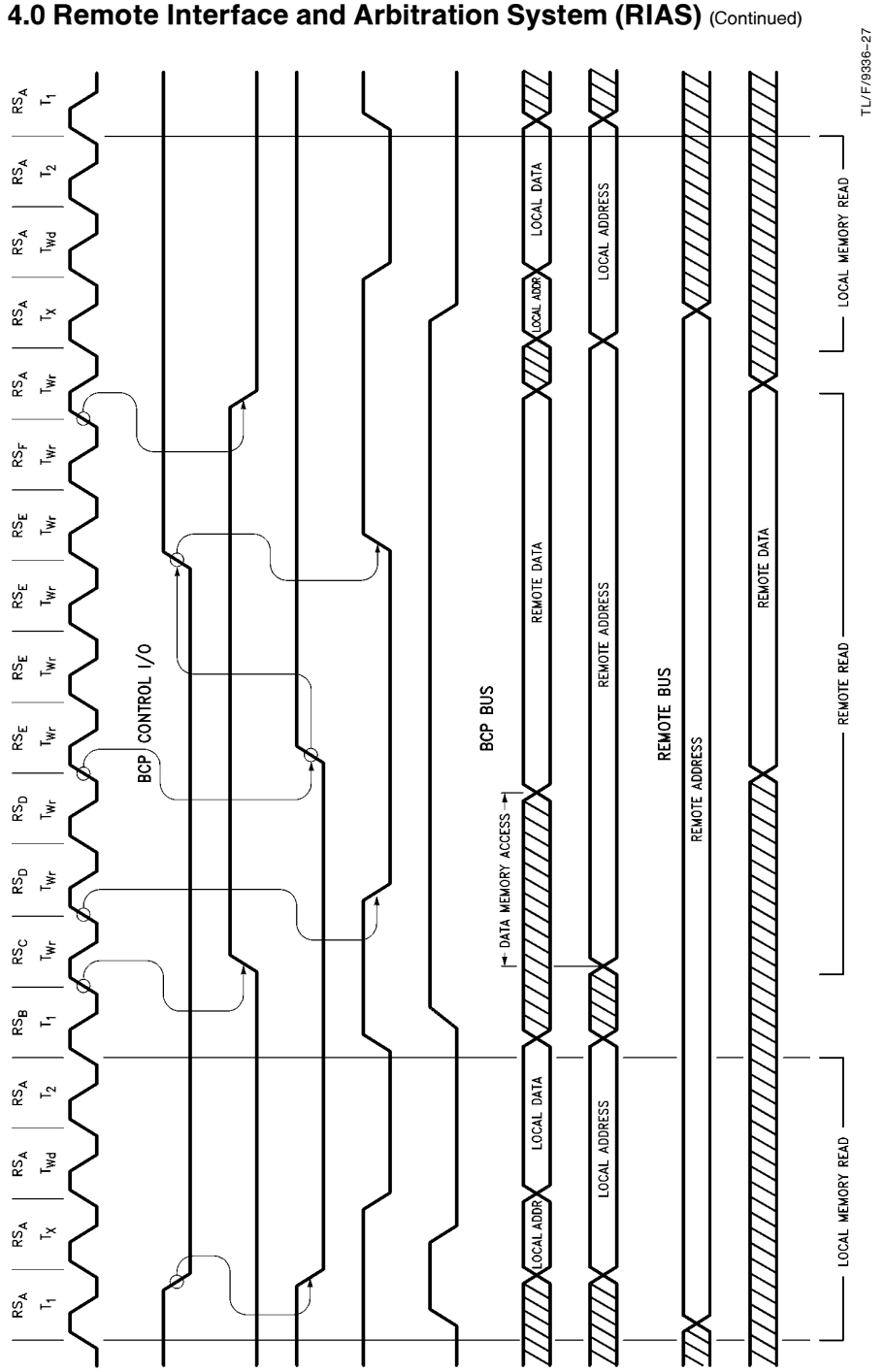


FIGURE 4-14. Flow Chart of Buffered Read Mode



Register Configuration:
—One Wait-State Programmed for Data-Memory
—Zero Wait-States Programmed for Instruction-Memory
—{RIC} Contents: XXX0X100
—[LOR] = 0

Other BCP Control Signals:

$\overline{RAE}$	= 0
CMD	= 0
$\overline{REM\text{-}WR}$	= 1
$\overline{LOCK}$	= 1

FIGURE 4-15. Buffered Read of Data Memory by Remote Processor

The last possible Memory Selection is Instruction Memory, [MS1–0] = 01. The two possible next states for an IMEM access depend on if RASM is expecting the low byte or high byte. Instruction words are accessed low byte then high byte and RASM powers up expecting the low Instruction byte. The internal flag that keeps track of the next expected Instruction byte is called the High Instruction Byte flag (HIB). If HIB is low, the next state is RS_{D5} and the low instruction byte is MUXed to the AD bus. If HIB is high, the high instruction byte is MUXed to AD and RS_{D6} is entered. An IMEM access, like a DMEM access, is subject to wait states and these states will be looped on until all programmed instruction memory wait states have been inserted.

Note: Resetting the BCP will reset HIB (i.e., HIB = 0). Writing 01 to the Memory Select bits in {RIC} (i.e., [MS1–0] = 01, pointing to IMEM) will also force HIB to zero. This way the instruction word boundary can be reset without resetting the BCP.

After all of the programmed wait states are inserted in the RS_D states, more wait states may be added by asserting $\overline{WAIT}$ low a half T-state before the end of the last programmed wait state. If there are no programmed wait states, $\overline{WAIT}$ must be asserted low a half T-state before the end of RS_D to add wait states. If $\overline{WAIT}$ remains low, the remote access is extended indefinitely. All the RS_D states move to their corresponding RS_E states on the CPU-CLK after the programmed wait state conditions are met and $\overline{WAIT}$ is high. The RS_E states are looped upon until RAE* REM-RD is deasserted. $\overline{LCL}$ remains high in all RS_E states and A remains in TRI-STATE. AD will also stay in TRI-STATE if the access was to DMEM. XACK is taken back high to indicate that data is now valid on the read. If XACK is connected to a Remote Processor wait pin, it is no longer waited and can now terminate its read cycle. This state begins the Termination Phase. The action specified in the conditional box is only executed while RAE*REM-RD is asserted—a clock edge is not necessary. In all RS_E states except RS_{E4} (DMEM) LCL will fall a propagation delay after RAE*REM-RD is deasserted. In RS_{E4}, $\overline{LCL}$ remains high through the whole state.

On the CPU-CLK after RAE*REM-RD is deasserted. RASM, enters RS_{F1} from every RS_E state except RS_{E4} (DMEM). In RS_{F1}, $\overline{LCL}$ remains low and A remains in TRI-STATE while CPU-CLK is high (i.e., for the first half T-state of RS_{F1}).

From RS_{E4}, RASM enters RS_{F2} on the CPU-CLK after RAE*REM-RD is deasserted. In RS_{F2}, $\overline{LCL}$ remains high while both A and AD remain in TRI-STATE.

From RS_{F1}, the next clock will return the state machine back to state RS_{A1} where it will loop until another Remote Access is initiated. If the access was to IMEM, then the last action of the remote access before returning to RS_A is to switch HIB and increment the PC if the high byte was read.

From RS_{F2}, the next CPU-CLK returns to state RS_{A3} where $\overline{LCL}$ returns low, but A and AD remain TRI-STATE for the first half T-state of RS_{A3}. If no Remote Access is initiated the next state will be RS_{A1} where it will loop until another Remote Access is initiated.

The example in *Figure 4-15* shows the BCP executing the first of two consecutive Data Memory reads when $\overline{REM\text{-}RD}$ goes low. In response, XACK goes low waiting the remote processor. At the end of the first instruction, although the BCP begins its second read by taking ALE high, the RASM now takes control of the bus and takes $\overline{LCL}$ high at the end of T_1. A one T-state delay is built into this transfer to ensure that $\overline{READ}$ has been deasserted before the data bus is switched. The Timing Control Unit is now waited, inserting remote access wait states, T_{Wr}, as RASM takes over.

The remote address is permitted one T-state to settle on the BCP address bus before $\overline{READ}$ goes low, XACK then returns high one T-state plus the programmed Data Memory wait state, T_{Wd} later, having satisfied the memory access time. The Remote Processor will respond by deasserting $\overline{REM\text{-}RD}$ high to which the BCP in turn responds by deasserting $\overline{READ}$ high. Following $\overline{READ}$ being deasserted high, the BCP waits till the end of the next T-state before taking $\overline{LCL}$ low, again ensuring that the read cycle has concluded before the bus is switched. Control is then returned to the Timing Control Unit and the local memory read continues.

4.2.2 Latched Read

This mode differs from the Buffered Read mode in the way the access is terminated. A latched Read cycle ends after the data being read is valid and the termination doesn't wait for the trailing edge of $\overline{REM\text{-}RD}$. Therefore the Arbitration and Access Phases of the Latched Read mode are the same as for the Buffered Read mode. The complete flow chart for the Latched Read mode is shown in *Figure 4-16*.

4.0 Remote Interface and Arbitration System (RIAS) (Continued)

Until a Remote Read is initiated (RAE*REM-RD true), the state machine (RASM) loops in state RS_{A1}. If a Remote Read is initiated and [LOR] is set high, RASM will move to state RS_{A2}. Likewise, if a Remote Read is initiated while the buses have been granted locally (i.e., Local Bus Grant = 1), RASM will move to state RS_{A2}. The state machine will loop in state RS_{A2}, as long as [LOR] is set high or the buses are granted locally. If the BCP CPU needs to access Data Memory while in either RS_A state (and $\overline{LOCK}$ is high), it can still do so. A local access is requested by the Timing Control Unit asserting the Local Bus Request (LCL-BREQ) signal. A local bus grant will be given by RASM if the buses are not being used (as is the case in RS_A).

XACK is taken low as soon as RAE*REM-RD is true, regardless of an ongoing local access. If [LOR] is low, RASM will move into RS_B on the next clock after RAE*REM-RD is asserted and there is no local bus request. No further local bus requests will be granted until RASM enters the Termination Phase. If the BCP CPU initiates a Data Memory access after RS_A, the Timing Control Unit will be waited and the BCP CPU will remain in state T_{Wr} until the remote access reaches the Termination Phase. Half a T-state after entering RS_B the A bus (and AD bus if the access is to Data Memory) goes into TRI-STATE.

On the next clock, RASM enters RS_C and $\overline{LCL}$ is taken high while XACK remains low. The wait state counters, i_{IW} and i_{DW}, are loaded in this state from [IW1–0] and [DW2–0], respectively, in {DCR}. The A bus (and AD if the access is to Data Memory) now remains TRI-STATE and the Access Phase begins.

The state machine can move into one of several states, depending on the state of CMD and [MS1–0], on the next clock. XACK remains low and $\overline{LCL}$ remains high in all the possible next states. If CMD is high, the access is to {RIC} and the next state will be RS_{D1}. Since the default state of AD is {RIC}, it will not transition in this state. The five other next states all have CMD low and depend on the Memory Select bits. If [MS1–0] is 10 or 11 the state machine will enter either RS_{D2} or RS_{D3} and the low or high bytes of the Program Counter, respectively, will be read.

[MS1–0] = 00 designates a Data Memory access and moves RASM into RS_{D4}. $\overline{READ}$ will be asserted low in this state and A and AD continue to be tri-stated. This allows the Remote Processor to drive the Data Memory address for the read. Since DMEM is subject to wait states, RS_{D4} is looped upon until all the wait states have been inserted.

The last possible Memory Selection is Instruction Memory, [MS1–0] = 01. The two possible next states for the IMEM access depend on if RASM is expecting the low byte or high byte. Instruction words are accessed low byte then high byte and RASM powers up expecting the low Instruction byte. The internal flag that keeps track of the next expected Instruction byte is called the High Instruction Byte flag (HIB). If HIB is low, the next state is RS_{D5} and the low instruction byte is MUXed to the AD bus. If HIB is high, the high instruction byte is MUXed to AD and RS_{D6} is entered. An IMEM access, like a DMEM access, is subject to wait states and these states will be looped on until all programmed instruction memory wait states have been inserted.

Note: Resetting the BCP will reset HIB (i.e., HIB = 0). Writing 01 to the Memory Select bits in {RIC} (i.e., [MS1–0] = 01, pointing to IMEM) will also force HIB to zero. This way the instruction word boundary can be reset without resetting the BCP.

After all of the programmed wait states are inserted in the RS_D states, more wait states may be added by asserting $\overline{WAIT}$ low a half T-state before the end of the last programmed wait state. If there are no programmed wait states $\overline{WAIT}$ must be asserted low a half T-state before the end of RS_D to add wait states. If $\overline{WAIT}$ remains low, the remote access is extended indefinitely. All the RS_D states move to their corresponding RS_E states on the CPU-CLK after the programmed wait state conditions are met and $\overline{WAIT}$ is high. $\overline{LCL}$ remains high in all RS_E states and A remains in TRI-STATE (and AD if the access is to Data Memory). XACK returns high in this state, indicating that data is valid so that it can be externally latched. The action specific to each RS_D state remains in effect during the first half of the RS_E cycle (i.e. $\overline{READ}$ is asserted in the first half of RS_{E4}). This half T-state of hold time is provided to guarantee data is latched when XACK goes high. This state begins the Termination Phase.

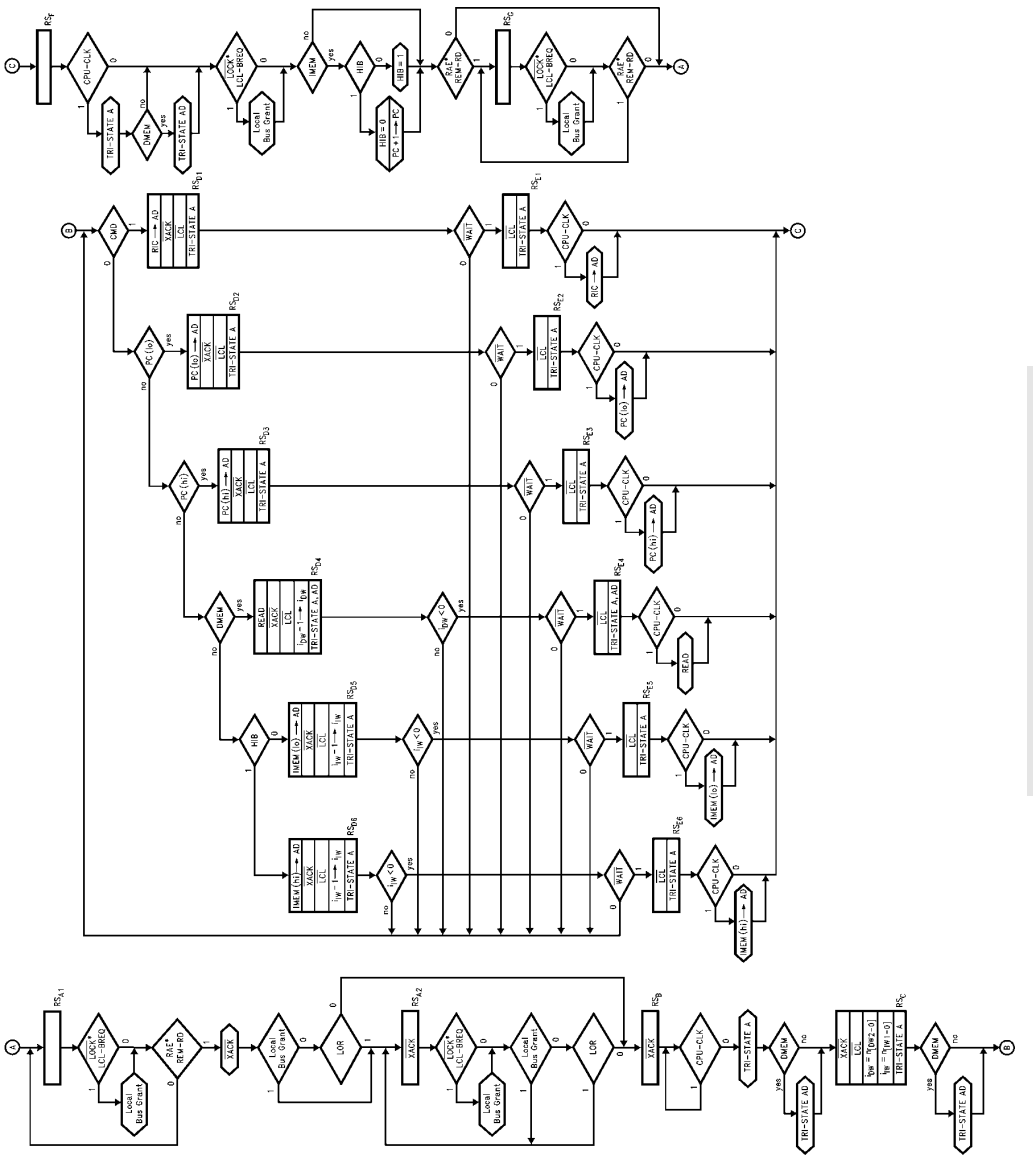
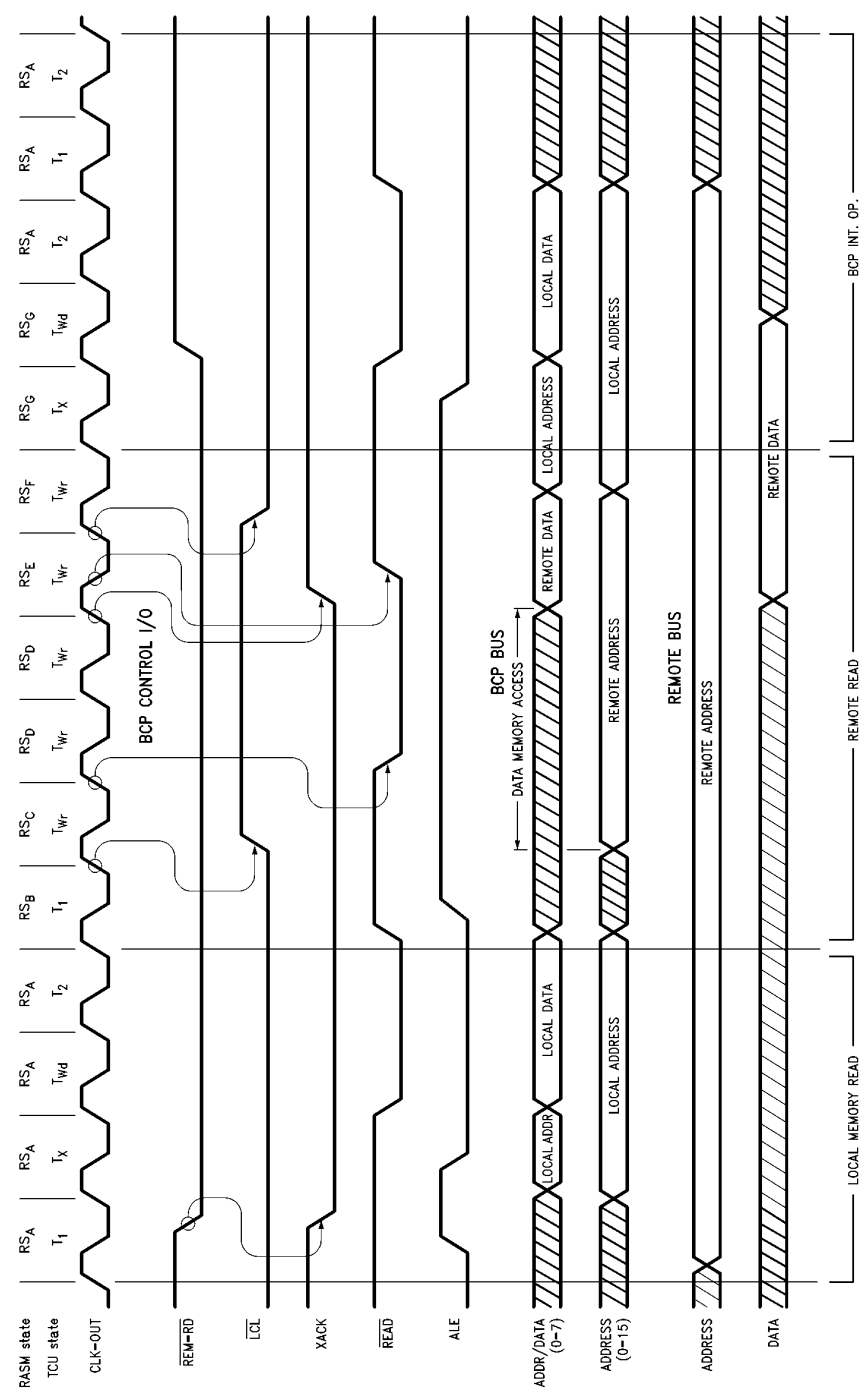


FIGURE 4-16. Flow Chart of Latched Read Mode



Register Configuration:
—One Wait-State Programmed for Data-Memory
—Zero Wait-States Programmed for Instruction-Memory
—{RIC} Contents: XXX1X100
—[LOR] = 0

Other BCP Control Signals:

$\overline{RAE}$	= 0
CMD	= 0
$\overline{REM\text{-}WR}$	= 1
$\overline{LOCK}$	= 1

FIGURE 4-17. Latched Read of Data Memory by Remote Processor

On the next clock the state machine will enter RS_F and $\overline{LCL}$ will return low. The A bus (and AD bus if the access is to data memory) remains in TRI-STATE for the first half T-state of RS_F. After the first half of RS_F, the Remote Processor is no longer using the buses and the BCP CPU will be granted the buses if LCL-BREQ is asserted. If a local bus request is made, a local bus grant will be given to the Timing Control Unit. If the preceding access was a read of IMEM, then HIB is switched and if the access was to the high byte of IMEM then the PC is incremented. If RAE* REM-RD is deasserted at this point, the next clock will bring RASM back to RS_A where it will loop until another Remote Access is initiated. RS_G is entered if RAE*REM-RD is still true. RASM will loop in RS_G until RAE*REM-RD is no longer active at which time the state machine will return to RS_A.

In *Figure 4-17*, the BCP is executing the first of two Data Memory reads when $\overline{REM\text{-}RD}$ goes low. In response, XACK goes low, waiting the Remote Processor. At the end of the first instruction, although the BCP begins its second write by taking ALE high, the RASM now takes control of the bus and deasserts $\overline{LCL}$ high at the end of T_1. A one T-state delay is built into this transfer to ensure that $\overline{READ}$ has been deasserted high before the data bus is switched. The Timing Control Unit is now waited, inserting remote access wait states, T_{Wr}, as RASM takes over.

The remote address is permitted one T-state to settle on the BCP address bus before $\overline{READ}$ goes low, XACK then returns high one T-state plus the programmed Data Memory wait state, T_{Wd} later, having satisfied the memory access time. $\overline{READ}$ returns high a half T-state later, ensuring sufficient hold time, followed by $\overline{LCL}$ being reasserted low after an additional half T-state, transferring bus control back to the BCP. The Remote Processor responds to XACK returning high by deasserting $\overline{REM\text{-}RD}$ high, although by this time the BCP is well into its own memory read.

4.2.3 Slow Buffered Write

The timing for this mode is the same as the Buffered Read mode. The complete flow chart for the Slow Buffered Write mode is shown in *Figure 4-18*. Until a Remote Write is initiated (RAE*REM-WR true), the state machine (RASM) loops in state RS_{A1}. If a Remote Write is initiated and [LOR] is set high, RASM will move to state RS_{A2}. Likewise, if a Remote Write is initiated while the buses have been granted locally (i.e., Local Bus Grant = 1), RASM will move to state RS_{A2}. The state machine will loop in state RS_{A2} as long as [LOR] is set high or the buses are granted locally. If the BCP CPU needs to access Data Memory while in either RS_A state (and $\overline{LOCK}$ is high), it can still do so. A local access is requested by the Timing Control Unit asserting the Local Bus Request (LCL-BREQ) signal. A local bus grant will be given by RASM if the buses are not being used (as is the case in the RS_A state).

XACK is taken low as soon as RAE*REM-WR is true, regardless of an ongoing local access. RASM will move into RS_B on the next clock after RAE*REM-WR is asserted and there is no local bus request and [LOR] = 0. No further local bus requests will be granted until the remote access is complete and RASM returns to RS_A. If the BCP CPU initiates a Data Memory access after RS_A, the Timing Control Unit will be waited and the BCP CPU will remain in state T_{Wr} until completion of the remote access. Half a T-state after entering RS_B the A and AD buses go into TRI-STATE.

On the next CPU-CLK, RASM enters RS_C and $\overline{LCL}$ is taken high while XACK remains low. The wait state counters, i_{IW} and i_{DW}, are loaded in this state from [IW1–0] and [DW2–0], respectively, in {DCR}. The A and AD buses now remain in TRI-STATE and the Access Phase begins. If the Remote Access is to IMEM and the high instruction byte flag is set (i.e., HIB = 1), then $\overline{IWR}$ is asserted low in RS_C. The state machine can move into one of several states, depending on the state of CMD and [MS1–0], on the next clock. XACK remains low and $\overline{LCL}$ remains high in all the possible next states. If CMD is high, the access is to {RIC} and the next state will be RS_{D1}. The path from AD to {RIC} opens in this state. Any remote access mode changes made by this write will not take effect until one T-state after the completion of the present write.

The five other next states all have CMD low and depend on the Memory Select bits. If [MS1–0] is 10 or 11, the state machine will enter either RS_{D2} or RS_{D3} and the low or high bytes of the Program Counter, respectively, will be written.

[MS1–0] equal to 00 designates a Data Memory access and moves RASM into RS_{D4}. $\overline{WRITE}$ will be asserted in this state and A and AD continue to be tri-stated. This allows the Remote Processor to drive the Data Memory address and data buses for the write. Since DMEM is subject to wait states, RS_{D4} is looped upon until all the programmed data memory wait states have been inserted.

The last possible Memory Selection is Instruction Memory, [MS1–0] = 01. The two possible next states for IMEM depend on whether RASM is expecting the low byte or high byte. Instruction words are accessed low byte, then high byte and RASM powers up expecting the low Instruction byte. The internal flag that keeps track of the next expected Instruction byte is called the High Instruction Byte flag (HIB). If HIB is low, the next state is RS_{D5} and the low instruction byte is written into the holding register, ILAT. If HIB is high, the high instruction byte is moved to I15–8 and the value in ILAT is moved to I7–0. At the same time, $\overline{IWR}$ is asserted low, beginning the write to instruction memory. An IMEM access, like a DMEM access, is subject to wait states and these states will be looped on until all programmed Instruction Memory wait states have been inserted.

Note: Resetting the BCP will reset HIB (i.e., HIB = 0). Writing 01 to the Memory Select bits in {RIC} (i.e., [MS1–0] = 01, pointing to IMEM) will also force HIB to zero. This way the instruction word boundary can be reset without resetting the BCP.

After all of the programmed wait states are inserted in the RS_D states, more wait states may be added by asserting $\overline{WAIT}$ low a half T-state before the end of the last programmed wait state. If there are no programmed wait states, $\overline{WAIT}$ must be asserted low a half T-state before the end of RS_D to add wait states. If $\overline{WAIT}$ remains low, the remote access is extended indefinitely. All the RS_D states move to their corresponding RS_E states on the CPU-CLK after the programmed wait state conditions are met and $\overline{WAIT}$ is high. The RS_E states are looped upon until RAE* REM-WR is deasserted. $\overline{LCL}$ remains high in all RS_E states, but XACK is taken back high to indicate that the remote access can be terminated. If XACK is connected to a Remote Processor wait pin, it can now terminate its write cycle. This state begins the Termination Phase. The action specified in the conditional box is only executed while RAE*REM-WR is asserted—a clock edge is not necessary.

On the CPU-CLK after RAE*REM-WR is deasserted, RASM enters RS_F, where $\overline{LCL}$ remains high and the BCP A and AD buses are still in TRI-STATE. The next CPU-CLK causes RASM to move to RS_{A3}. If the access was to IMEM, then

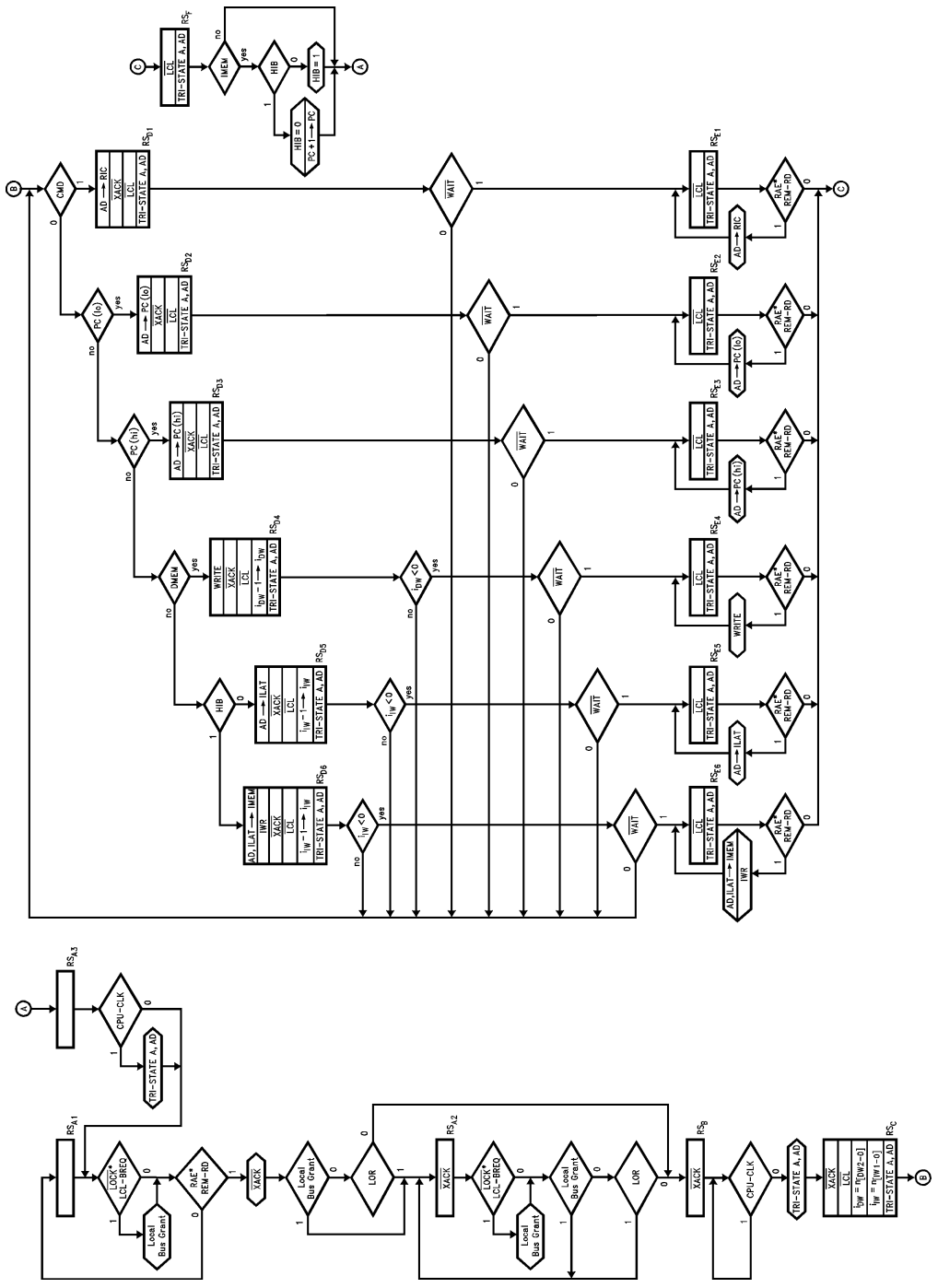


FIGURE 4-18. Flow Chart of Slow Buffered Write Mode

the last action of the remote access before moving to RS_{A3} is to switch HIB and increment the PC if the high byte was written. In RS_{A3}, $\overline{LCL}$ goes low while A and AD remain in TRI-STATE for the first half of RS_{A3}. If no new Remote access is initiated the next clock brings the state machine back to RS_{A1} where it will loop until a Remote Access is initiated.

In *Figure 4-19,* the BCP is executing the first of two consecutive Slow Buffered Writes to Data Memory when $\overline{REM\text{-}WR}$ goes low. In response, XACK goes low, waiting the Remote Processor. At the end of the first instruction, although the BCP begins its second write by taking ALE high, RASM now Takes control of the bus and deasserts $\overline{LCL}$ high at the end of T_1. A one T-state delay is built into this transfer to ensure that $\overline{WRITE}$ has been deasserted high before the data bus is switched. The Timing Control Unit is now waited, inserting remote access wait states, T_{Wr}, as RASM takes over.

The remote address is permitted one T-state to settle on the BCP address bus before $\overline{WRITE}$ goes low, XACK then returns high one T-state plus the programmed Data Memory wait state, T_{Wd} later, having satisfied the memory access time. The Remote Processor will respond by deasserting $\overline{REM\text{-}WR}$ high to which the BCP in turn responds by deasserting $\overline{WRITE}$ high. Following $\overline{WRITE}$ being deasserted high, the BCP waits till the end of the next T-state before asserting $\overline{LCL}$ low, again ensuring that the write cycle has concluded before the bus is switched. Control is then returned to the Timing Control Unit and the local memory write continues.

4.2.4 Fast Buffered Write

The timing for the Fast Buffered Write mode is very similar to the timing of the Latched Read. The major difference is the additional half clock that AD is active in the Latched Read mode that is not present in the Fast Buffered Write mode. The Fast Buffered Write cycle ends after the data is written and the termination doesn't wait for the trailing edge of $\overline{REM\text{-}WR}$. Therefore the Arbitration and Access Phases of the Fast Buffered Write mode are the same as for the Latched Read mode.

The complete flow chart for the Fast Buffered Write mode is shown in *Figure 4-20*. Until a Remote Write is initiated (RAE*REM-WR true), the state machine (RASM) loops in state RS_{A1}. If a Remote Write is initiated and [LOR] is set high, RASM will move to state RS_{A2}. Likewise, if a Remote Write is initiated while the buses have been granted locally (i.e., Local Bus Grant = 1), RASM will move to state RS_{A2}. The state machine will loop in state RS_{A2} as long as [LOR] is set high or the buses are granted locally. If the BCP CPU needs to access Data Memory while in either RS_A state (and $\overline{LOCK}$ is high), it can still do so. A local access is requested by the Timing Control Unit asserting the Local Bus Request (LCL-BREQ) signal. A local bus grant will be given by RASM if the buses are not being used (as is the case in the RS_A states).

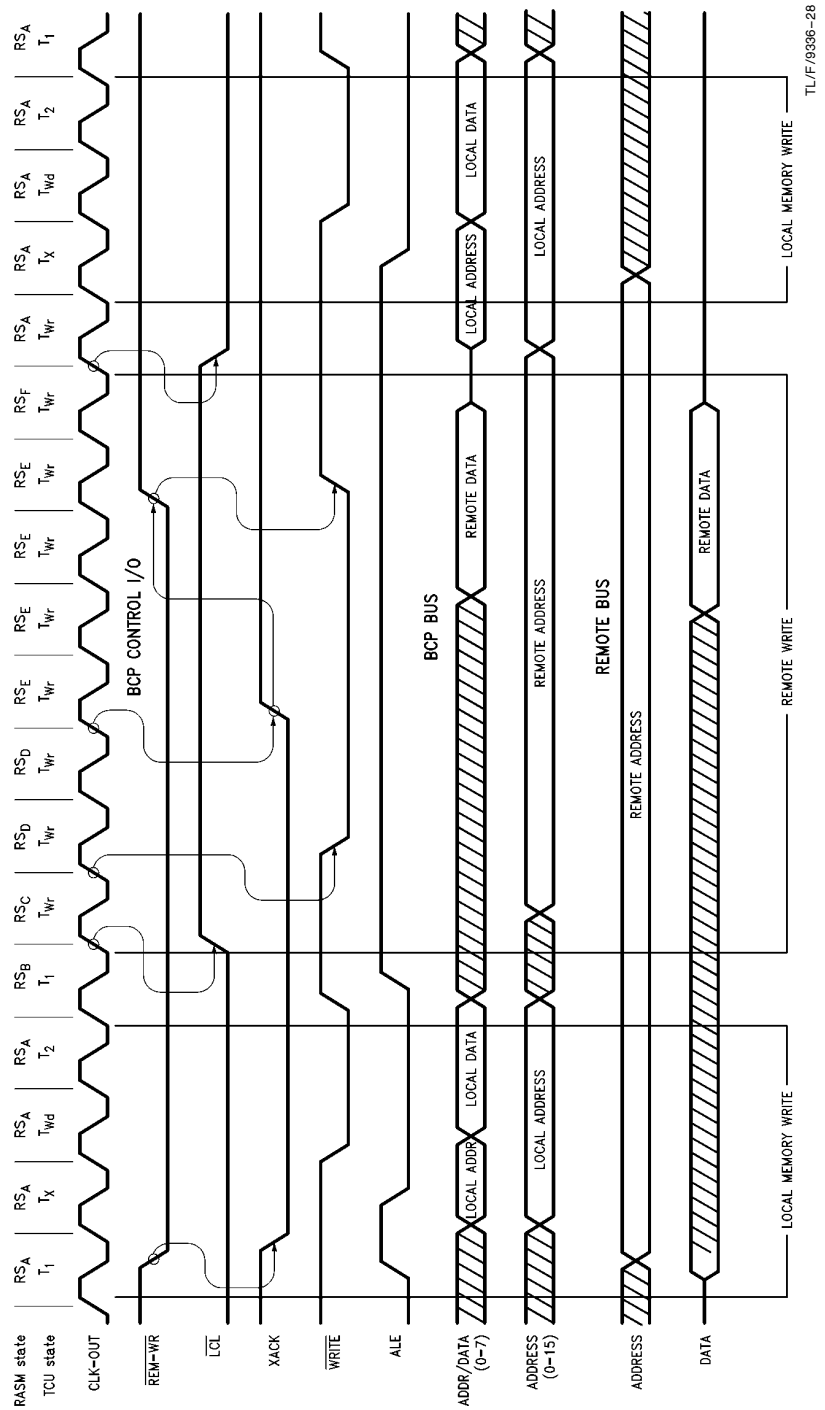
XACK is taken low as soon as RAE*REM-WR is true, regardless of an ongoing local access. If [LOR] is low, RASM will move into RS_B on the next clock after RAE*REM-WR is asserted and there is no local bus request. No further local bus requests will be granted until the BCP enters the Termination Phase. If the BCP CPU initiates a Data Memory access after RS_A, the Timing Control Unit will be waited and the BCP CPU will remain in state T_{Wr} until the remote access reaches the Termination Phase. Half a T-state after entering RS_B the A and AD buses go into TRI-STATE.

On the next CPU-CLK, RASM enters RS_C and $\overline{LCL}$ is taken high while XACK remains low. The wait state counters, i_{IW} and i_{DW}, are loaded in this state from [IW1–0] and [DW2–0], respectively, in {DCR}. The A and AD buses remain in TRI-STATE and the Access Phase begins. If the Remote Access is to IMEM and the high instruction byte flag is set (i.e., HIB = 1), then $\overline{IWR}$ is asserted low in RS_C.

The state machine can move into one of several states depending on the state of CMD and [MS1–0] on the next clock. XACK and $\overline{LCL}$ in all the possible next states. If CMD is high, the access is to {RIC} and the next state will be RS_{D1}. The path from AD to {RIC} opens in this state. Any remote access mode changes made by this write will not take effect until one T-state after the completion of the present write.

The five other next states all have CMD low and depend on the Memory Select bits. If [MS1–0] is 10 or 11 the state machine will enter either RS_{D2} or RS_{D3} and the low or high bytes of the Program Counter, respectively, will be written.

[MS1–0] = 00 designates a Data Memory access and moves RASM into RS_{D4}. $\overline{WRITE}$ will be asserted in this



Register Configuration:
—One Wait-State Programmed for Data-Memory
—Zero Wait-States Programmed for Instruction-Memory
—{RIC} Contents: XX0X0100
—[LOR] = 0

Other BCP Control Signals:

$\overline{RAE}$	= 0
CMD	= 0
$\overline{REM\text{-}RD}$	= 1
$\overline{LOCK}$	= 1

FIGURE 4-19. Slow Buffered Write to Data Memory by Remote Processor

state and A and AD continue to be tri-stated. This allows the Remote Processor to drive the Data Memory address and data buses for the write. Since DMEM is subject to wait states, RS_{D4} is looped upon until all the programmed Data Memory wait states have been inserted.

The last possible Memory Selection is Instruction Memory, [MS1–0] = 01. The two possible next states for IMEM depend on whether RASM is expecting the low byte or high byte. Instruction words are accessed low byte then high byte and RASM powers up expecting the low Instruction byte. The internal flag that keeps track of the next expected Instruction byte is called the High Instruction Byte flag (HIB). If HIB is low, the next state is RS_{D5} and the low instruction byte is written into the holding register, ILAT. If HIB is high, the high instruction byte is moved to I15–8 and ILAT is moved to I7–0. At the same time $\overline{IWR}$ is asserted low, beginning the write to instruction memory. An IMEM access, like a DMEM access, is subject to wait states and these states will be looped on until all programmed instruction memory wait states have been inserted.

Note: Resetting the BCP will reset HIB (i.e., HIB = 0). Writing 01 to the Memory Select bits in {RIC} (i.e., [MS1–0] = 01, pointing to IMEM) will also force HIB to zero. This way the instruction word boundary can be reset without resetting the BCP.

After all of the programmed wait states are inserted into RS_D states, more wait states may be added by asserting $\overline{WAIT}$ low a half T-state before the end of the last programmed wait state. If there are no programmed wait states $\overline{WAIT}$ must be asserted low a half T-state before the end of RS_D to add wait states. If $\overline{WAIT}$ remains low, the remote access is extended indefinitely. All the RS_D states converge to state RS_E on the next CPU-CLK after the programmed wait state conditions are met and $\overline{WAIT}$ is high. $\overline{LCL}$ remains high in all RS_E states and A and AD remain in TRI-STATE as well. XACK returns high in this state, indicating that the data is written and the cycle can be terminated by the RP. This state begins the Termination Phase.

On the next clock the state machine will enter RS_F and $\overline{LCL}$ will return low. The A and AD buses remain in TRI-STATE for the first half T-state of RS_F. After the first half of RS_F, the Remote Processor is no longer using the buses and the BCP CPU can make an access to Data Memory by asserting LCL-BREQ. If a local bus request is made, a local bus grant will be given to the Timing Control Unit. If the preceding access was a write of IMEM, then HIB is switched and if the access was to the high byte of IMEM then the PC is incremented. If RAE*REM-WR is deasserted at this point, the next clock will bring RASM back to RS_A where it will loop until another remote access is initiated. RS_G is entered if RAE*REM-WR is still true. RASM will loop in RS_G until RAE*REM-WR is no longer active at which time the state machine will return to RS_A.

In *Figure 4-21*, the BCP is executing the first of two Data Memory writes when $\overline{REM\text{-}WR}$ goes low. In response, XACK goes low, waiting the Remote Processor. At the end of the first instruction, although the BCP begins its second write by taking ALE high, RASM now takes control of the bus and deasserts $\overline{LCL}$ high at the end of T_1. A one T-state delay is built into this transfer to ensure that $\overline{WRITE}$ has been deasserted high before the data bus is switched. The Timing Control Unit is now waited, inserting remote access wait states, T_{Wr}, as RASM takes over.

The remote access is permitted one T-state to settle on the BCP address bus before $\overline{WRITE}$ goes low, XACK then returns high one T-state plus the programmed Data Memory wait state, T_{Wd} later, having satisfied the memory access time. $\overline{WRITE}$ returns high at the same time, and one T-state later $\overline{LCL}$ returns low, transferring bus control back to the BCP. The remote processor responds to XACK returning high by deasserting $\overline{REM\text{-}WR}$ high, although by this time the BCP is well into its own memory write.

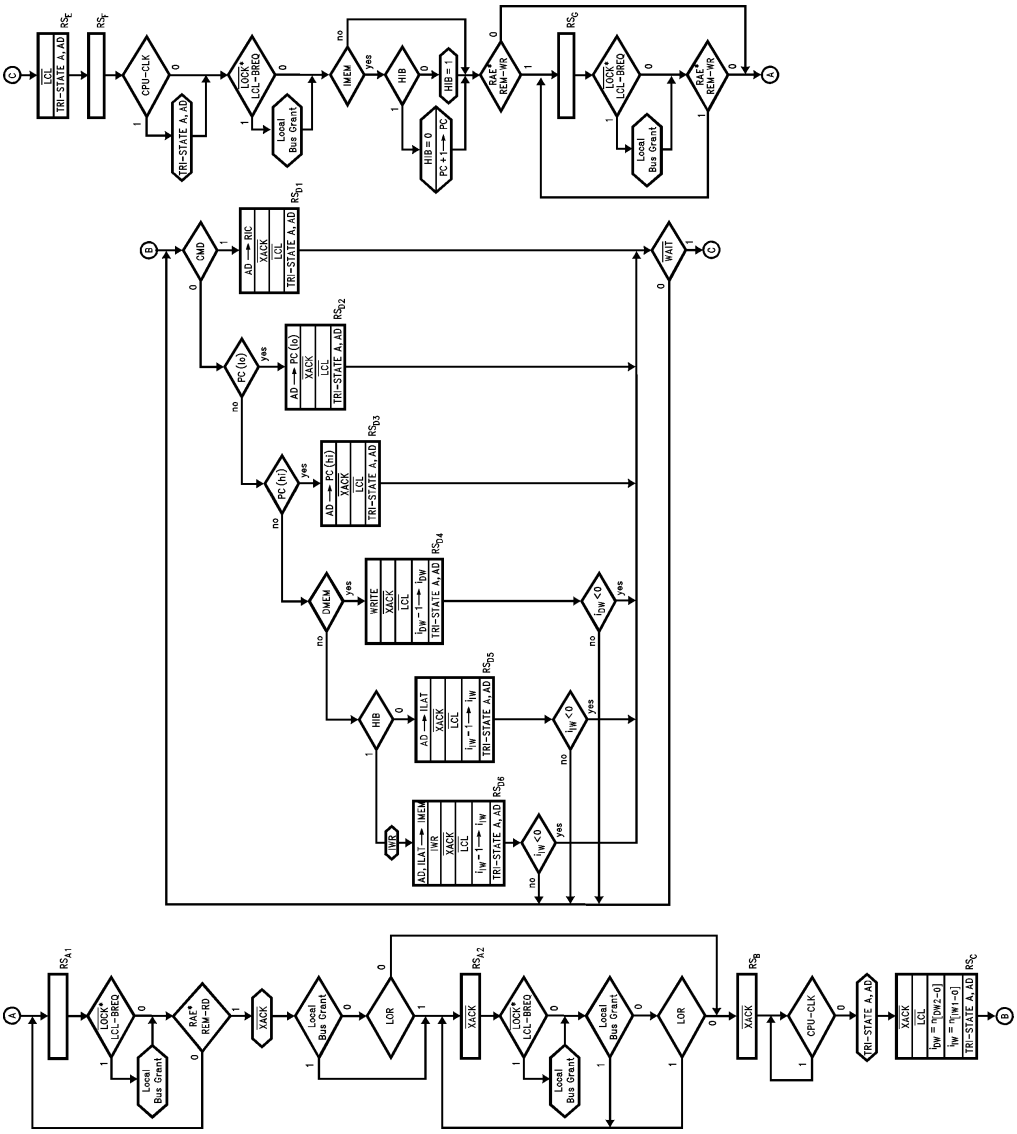
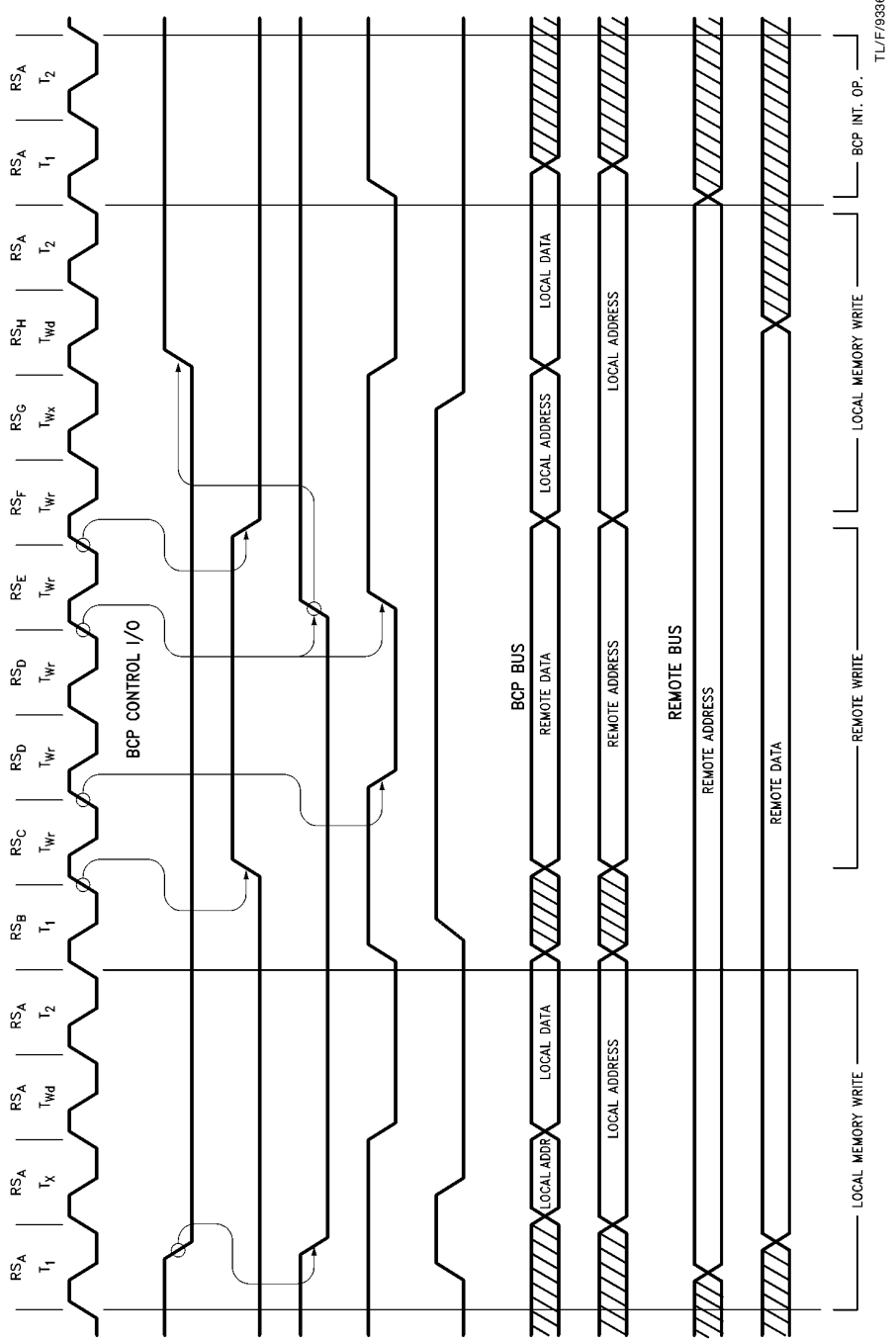


FIGURE 4-20. Flow Chart of Fast Buffered Write Mode



Register Configuration:
—One Wait-State Programmed for Data-Memory
—Zero Wait-States Programmed for Instruction-Memory
—{RIC} Contents: XX1X0100
—[LOR] = 0

Other BCP Control Signals:

$\overline{RAE}$	= 0
CMD	= 0
$\overline{REM\text{-}RD}$	= 1
$\overline{LOCK}$	= 1

FIGURE 4-21. Fast Buffered Write to Data Memory by Remote Processor

4.2.5 Latched Write

This mode executes a write without waiting the Remote Processor—XACK isn't normally taken low. The complete flow chart for the Latched Write mode is shown in *Figure 4-22*. Until a Remote Write is initiated (RAE*REM-WR true), the state machine (RASM) loops in state RS_A. If the BCP CPU needs to access Data Memory at this time (and $\overline{LOCK}$ is high), it can still do so. A local access is requested by the Timing Control Unit asserting the Local Bus Request (LCL-BREQ) signal. A local bus grant will be given by RASM if the buses are not being used (as is the case in RS_A).

RASM will move into RS_B on the next clock after RAE*REM-WR is asserted. XACK is not taken low and therefore the RP is not waited. The state machine will loop in RS_B until the RP terminates its write cycle—until RAE*REM-WR is no longer true. The external address and data latches are typically latched on the trailing edge of $\overline{REM\text{-}WR}$. A local bus request will still be serviced in this state.

Next, RASM enters RS_C and $\overline{WR\text{-}PEND}$ is asserted to prevent overwrite of the external latches. Since the RP has completed its write cycle, another write or read can happen at any time. Any Remote Read cycle (RAE*REM-RD) or Remote Write cycle (RAE*REM-WR) occurring after the state machine enters RS_C will take XACK low. A local access initiated before or during this state must be completed before RASM can move to RS_D. Once RS_D is entered, though, no further local bus requests will be granted until RASM enters the Termination Phase. If the BCP CPU initiates a Data Memory access after RS_C, the Timing Control Unit will be waited and the BCP CPU will remain in state T_{Wr} until the RASM enters RS_H. Half a T-state after entering RS_B the A and AD buses go into TRI-STATE.

On the next clock, the state machine enters RS_E and $\overline{LCL}$ is taken high. $\overline{WR\text{-}PEND}$ continues to be asserted low in this state and the data and instruction wait state counters, i_{DW} and i_{IW}, are loaded from [DW2–0] and [IW1–0], respectively, in {DCR}. The A and AD buses remain in TRI-STATE and the Access Phase begins. Any remote accesses now occurring will take XACK low and wait the Remote Processor. If the Remote Access is to IMEM and the high instruction byte flag is set (i.e., HIB = 1), then $\overline{IWR}$ is asserted low in RS_E.

The state machine will move into one of several states on the next clock, depending on the state of CMD and [MS1–0]. $\overline{WR\text{-}PEND}$ remains low and $\overline{LCL}$ remains high in all the possible next states. If CMD is high, the access is to {RIC} and the next state will be RS_{F1}. The path from AD to {RIC} opens in this state. Any remote access mode changes made by this write will not take effect until one T-state after the completion of the present write.

The five other next states all have CMD low and depend on the Memory Select bits. If [MS1–0] is 10 or 11 the state machine will enter either RS_{F2} or RS_{F3} and the low or high bytes of the Program Counter, respectively, will be loaded.

[MS1–0] = 00 designates a Data Memory access and moves RASM into RS_{F4}. $\overline{WRITE}$ will be asserted low in this state and A and AD continue to be tri-stated. This allows the Remote Processor to drive the Data Memory address and data for the write. Since DMEM is subject to wait states, RS_{F4} is looped upon until all the programmed Data Memory wait states have been inserted.

The last possible Memory Selection is Instruction Memory, [MS1–0] = 01. The two possible next states for IMEM depend on if RASM is expecting the low byte or high byte. Instruction words are accessed low byte then high byte and RASM powers up expecting the low Instruction byte. The internal flag that keeps track of the next expected Instruction byte is called the High Instruction Byte flag (HIB). If HIB is low, the next state is RS_{F5} and the low instruction byte is written into the holding register, ILAT. If HIB is high, the high instruction byte is moved to I15–8 and the value in ILAT is moved to I7–0. At the same time, $\overline{IWR}$ is asserted low and the write to Instruction Memory is begun. An IMEM access, like a DMEM access, is subject to wait states and these states will be looped on until all programmed instruction memory wait states have been inserted.

Note: Resetting the BCP will reset HIB (i.e., HIB = 0). Writing 01 to the Memory Select bits in {RIC} (i.e., [MS1–0] = 01, pointing to IMEM) will also force HIB to zero. This way the instruction word boundary can be reset without resetting the BCP.

All the RS_F states converge to a single decision box that tests $\overline{WAIT}$. If $\overline{WAIT}$ is low then the state machine loops back to RS_F, otherwise RASM will move on to RS_G. $\overline{LCL}$ remains high and $\overline{WR\text{-}PEND}$ remains low in this state but the actions specific to the RS_F states have ended (i.e. $\overline{WRITE}$ will no longer be asserted low).

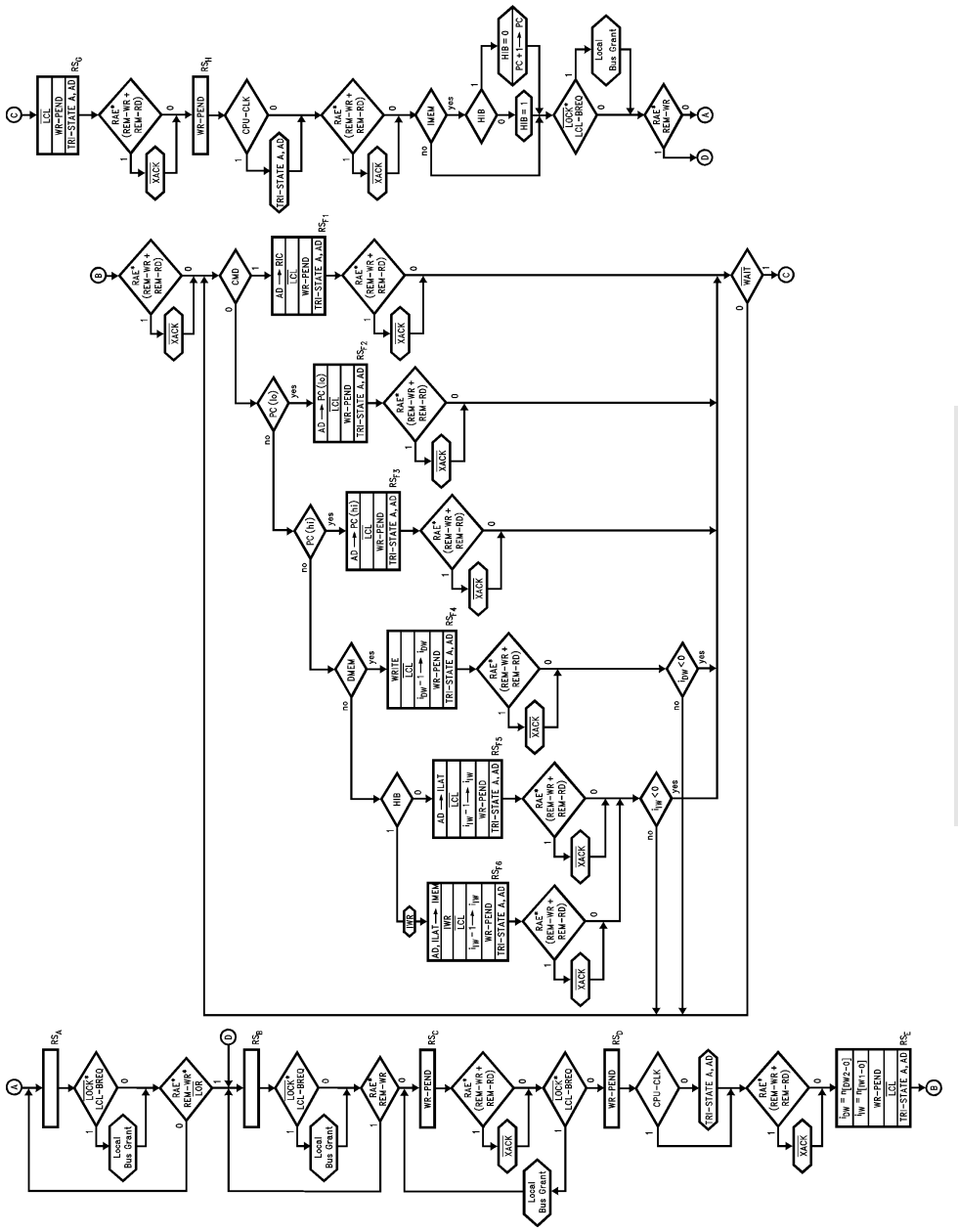
The next CPU-CLK moves RASM into RS_H, the last state in the state machine. $\overline{LCL}$ returns low but $\overline{WR\text{-}PEND}$ is still low. The A and AD buses remain in TRI-STATE for the first half of RS_H. XACK will be taken low if a Remote Access is initiated. If the just completed access was to IMEM, HIB will be switched. Also, the PC will be incremented if the high byte was written. A local access will be granted if LCL-BREQ is asserted in this state.

If another Remote Write is pending, the state machine takes the path to RS_B where that write will be processed. A pending Remote Read will return to the RS_A in either the Buffered or Latched Read sections (not shown in *Figure 4-22*) of the state machine. And if no Remote Access is pending, the machine will loop in RS_A until the next access is initiated.

In *Figure 4-23*, the BCP is executing the first of two Data Memory writes when $\overline{REM\text{-}WR}$ goes low. The BCP takes no action until $\overline{REM\text{-}WR}$ goes back high, latching the data and making a remote access request. The BCP responds to this by taking $\overline{WR\text{-}PEND}$ low. At the end of the first instruction, although the BCP begins its second write by taking ALE high, RASM now takes control of the bus and deasserts $\overline{LCL}$ high at the end of T_1. A one T-state delay is built into this transfer to ensure that $\overline{WRITE}$ has been deasserted high before the data bus is switched. Timing Control Unit is now waited, inserting remote access wait states, T_{Wr}, as RASM takes over.

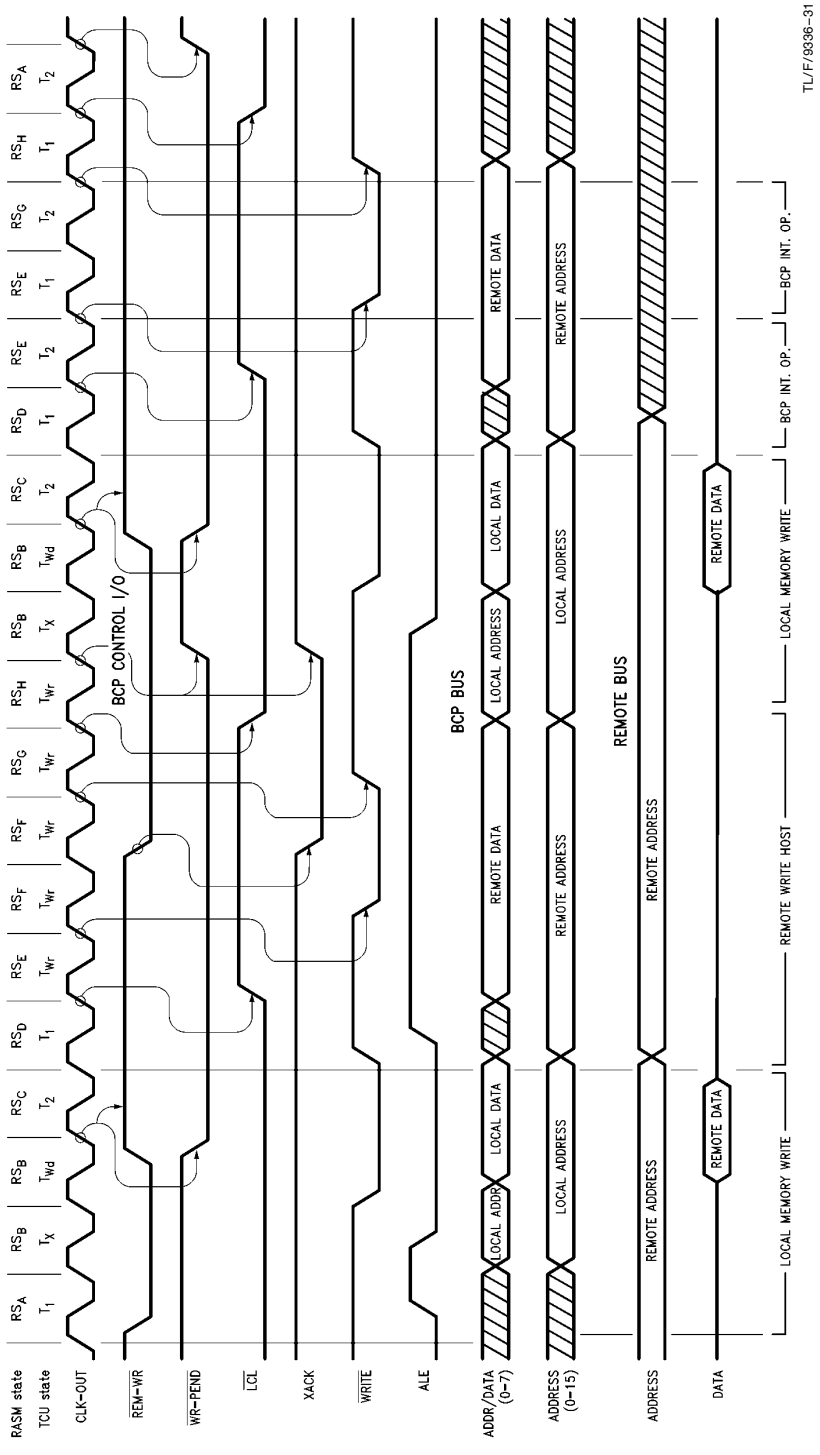
The remote address is permitted one T-state to settle on the BCP address bus before $\overline{WRITE}$ goes low. $\overline{WRITE}$ then returns high one T-state plus the programmed Data Memory wait state, T_{Wd} later, having satisfied the memory access time, and one T-state later $\overline{LCL}$ is reasserted low, transferring bus control back to the BCP.

In this example, $\overline{REM\text{-}WR}$ goes low again during the remote write cycle which, since $\overline{WR\text{-}PEND}$ is still low, causes XACK to go low to wait the Remote Processor. Then $\overline{LCL}$ goes low, allowing the second data byte to be latched on the next trailing edge of $\overline{REM\text{-}WR}$. One T-state later. XACK and $\overline{WR\text{-}PEND}$ go back high at the same time.



TL/F/9336–A1

FIGURE 4-22. Flow Chart of Latched Write Mode



Register Configuration:
—One Wait-State Programmed for Data-Memory
—Zero Wait-States Programmed for Instruction-Memory
—{RIC} Contents: XXXX1100
—[LOR] = 0

Other BCP Control Signals:
$\overline{RAE}$ = 0
CMD = 0
$\overline{REM\text{-}RD}$ = 1
$\overline{LOCK}$ = 1

FIGURE 4-23. Latched Write to Data Memory by Remote Processor

The BCP is now shown executing a local memory write, with remote data still pending in the latch. At the end of this instruction, the BCP begins executing a series of internal operations which do not require the bus. RASM therefore takes over and, without waiting the Timing Control Unit, executes the Remote Write.

4.2.6 Remote Rest Time

For the BCP to operate properly, remote accesses to the BCP must be separated by a minimal amount of time. This minimal amount of time has been termed ''rest time''.

There are two causes for remote rest time. The first cause is implied in the functional state machine forms for remote accesses and can be explained as follows: At the beginning of every T-state the validity of a remote access is sampled for that T-state. To guarantee that the BCP recognizes the end of a remote cycle, the time between remote accesses must be a minimum of one T-state plus set up and hold times.

In the case of Latched Read and Fast Buffered Write, the validity of a remote access is not sampled on the first rising edge of the CPU-CLK following XACK rising. However, on all subsequent rising edges of the CPU-CLK the validity of the remote access is sampled. As a result, if the remote processor can terminate its remote access quickly after XACK rises (within a T-state), up to a T-state may be added to the above equation for Latched Read and Fast Buffered Write modes (i.e., a second remote access should not begin for two T-states plus set up and hold times after XACK rises in Latched Read and Fast Buffered Write modes). On the other hand, if the remote processor does not terminate its remote access within a T-state of XACK rising, the above equation (one T-state plus set up and hold times between remote accesses) remains valid for Latched Read and Fast Buffered Write modes.

If these specifications are not adhered to, the BCP may sample the very end of one valid remote access and one T-state later sample the very beginning of a second remote access. Thus, the BCP will treat the second access as a continuation of the first remote access and will not perform the second read/write. The second access will be ignored. (Reference *Figure 4-24* for the timing diagrams which demonstrate how two remote accesses can be mistaken as one.)

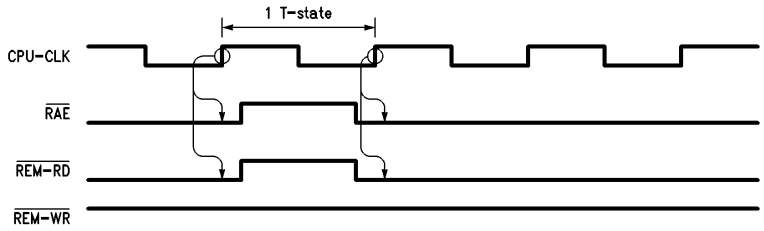
The second source of remote rest time is due to the manner in which the BCP samples the CMD signal. CMD is sampled once at the beginning of each remote access. Due to the manner in which CMD is sampled, CMD will not be sampled again if a second remote access begins within 1.5 T-states plus a hold time, after the BCP recognizes the end of the first remote access. If this happens, the BCP will use the value of CMD from the previous remote access during the second remote access. If the value of CMD is the same for both accesses, the second access will proceed as intended. However, if the value of CMD is different for the two remote accesses, the second remote access will read/write the wrong location.

The reader should note that the timing of the second source of rest time begins at the same time that the BCP first samples the end of the previous remote access. Thus when the first source of rest time ends, the second source of rest time begins. (Reference *Figure 4-25* for timing diagrams for rest time in all modes except Latched Write mode).

Latched Write Mode

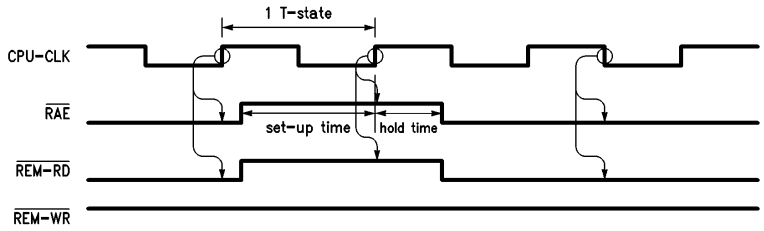
Latched Write mode is a special case of rest time and needs to be discussed separately from the other modes. The first cause of rest time affects every mode including Latched Write. In regards to the second source of rest time, Latched Write mode was designed to allow a second remote access to start while a write is still pending (i.e., $\overline{\text{WR-PEND}} = 0$). Thus, when $\overline{\text{WR-PEND}}$ rises (signaling the end of the previous write) the value of CMD is sampled for the second remote access. This allows Latched Write to avoid the second cause of rest time discussed above.

However, if a remote access begins within one half a T-state after $\overline{\text{WR-PEND}}$ rises, CMD will not be sampled again. For this case, if the value of CMD changes just after $\overline{\text{WR-PEND}}$ rose and at the same time the remote access begins, the BCP will read/write the wrong location. (Reference *Figure 4-26* for timing diagrams of rest time for latched write mode.)



TL/F/9336–G5

(a) This timing diagram shows two remote accesses within one T-state. The first set of arrows shows the BCP sampling a valid remote read. The next time the BCP samples the validity of the remote access is shown by the second set of arrows (1 T-state later). In this case, it will sample the second remote access and mistake it as a continuation of the first remote access.

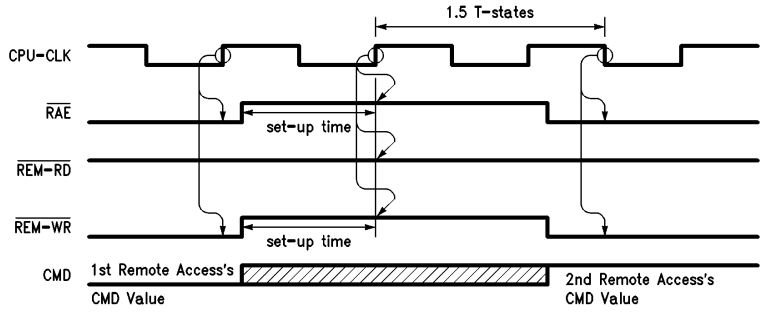


TL/F/9336–G6

(b) This timing diagram shows the timing necessary for the BCP to recognize both accesses as separate accesses. The first set of arrows shows the BCP sampling a valid remote read. One T-state later at the second set of arrows the BCP will sample the end of the first remote access. Another T-state later at the third set of arrows the BCP will sample the beginning of the second remote access.

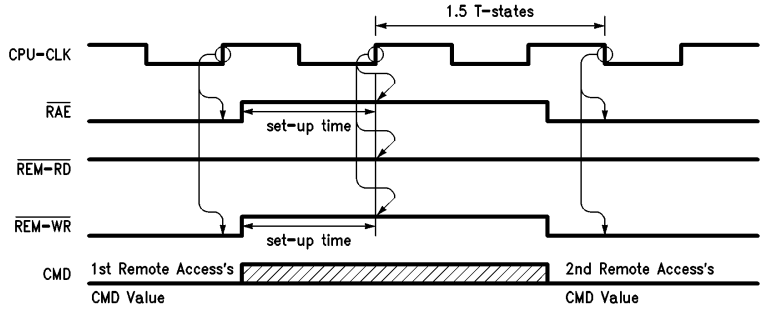
FIGURE 4-24. Mistaking Two Remote Accesses as Only One

4.0 Remote Interface and Arbitration System (RIAS) (Continued)



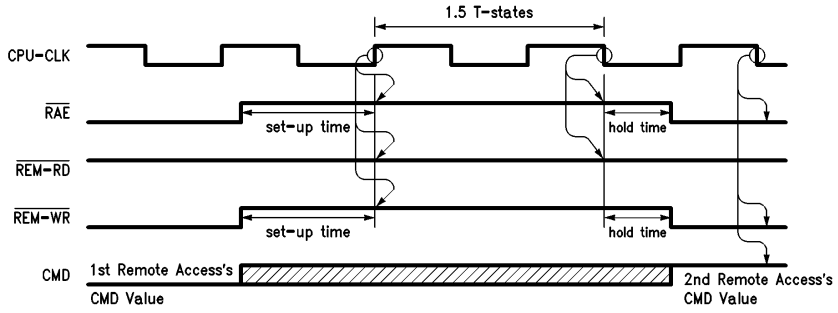
TL/F/9336–G7

(a) This timing diagram shows the second remote access violating rest time. The first set of arrows shows the BCP sampling a valid remote write. The second set of arrows (1 T-state later), shows the BCP sampling the end of the first remote access. If a second remote access starts before the position of the third set of arrows (another 1.5 T-states later), the value of CMD will not be sampled. The value of CMD has changed from the first remote access, so the BCP will write to the wrong location during the second access.



TL/F/9336–G8

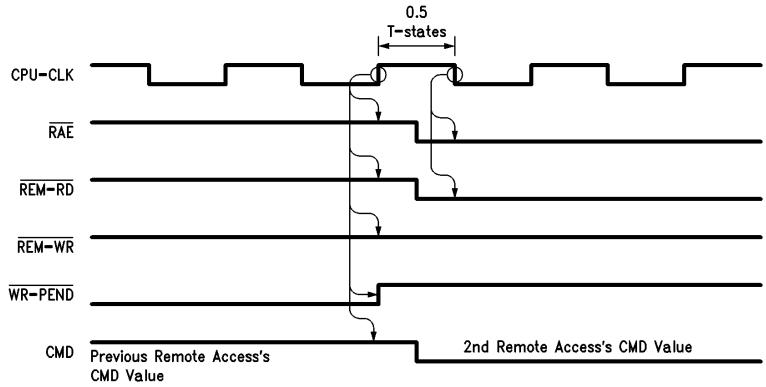
(b) This timing diagram shows the second remote access violating rest time. The first set of arrows shows the BCP sampling a valid remote write. The second set of arrows (1 T-state later), shows the BCP sampling the end of the first remote access. If a second remote access starts before the position of the third set of arrows (another 1.5 T-states later), the value of CMD will not be sampled. The value of CMD does not change from the first remote access, so the BCP will write to the intended location during the second remote access.



TL/F/9336–G9

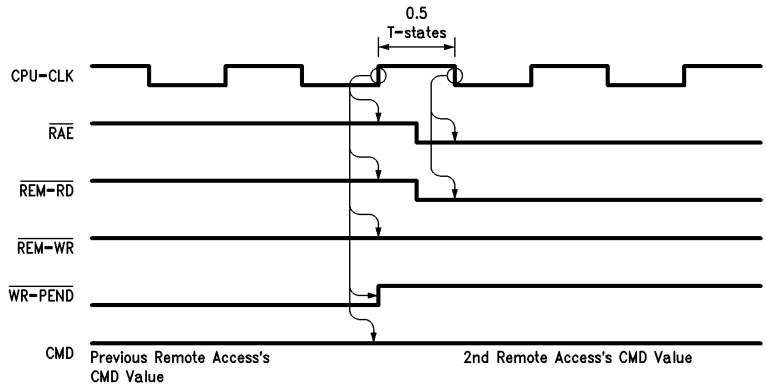
(c) This timing diagram shows the timing needed to avoid violating rest time for all modes except latched write. The first set of arrows shows the BCP sampling the end of the first remote access. The second set of arrows (1.5 T-states later), shows the BCP recognizing no remote access has started and the value of CMD will be sampled for the next remote access. The third set of arrows shows the BCP sampling the correct value of CMD for the second remote access.

FIGURE 4-25. Remote Rest Time for All Modes except Latched Write



TL/F/9336–H1

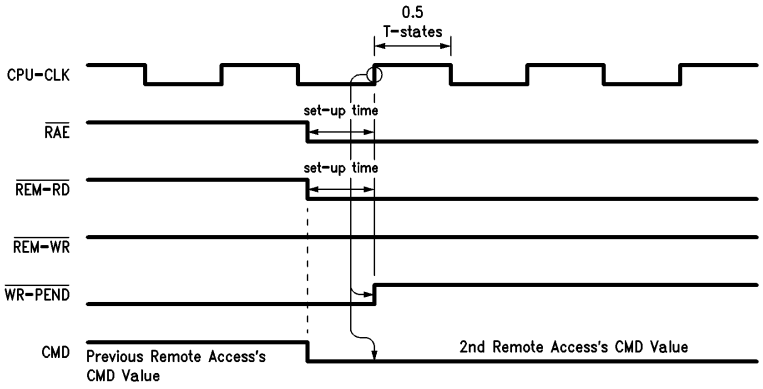
(a) This timing diagram shows a remote access violating remote rest time. The first set of arrows shows the BCP sampling the value of CMD when WR-PEND rises. If a remote access begins after WR-PEND rises and before the position of the second set of arrows (0.5 T-states later), the value of CMD will not be sampled again. The value of CMD has changed since WR-PEND rose, so the BCP will read the wrong location.



TL/F/9336–H2

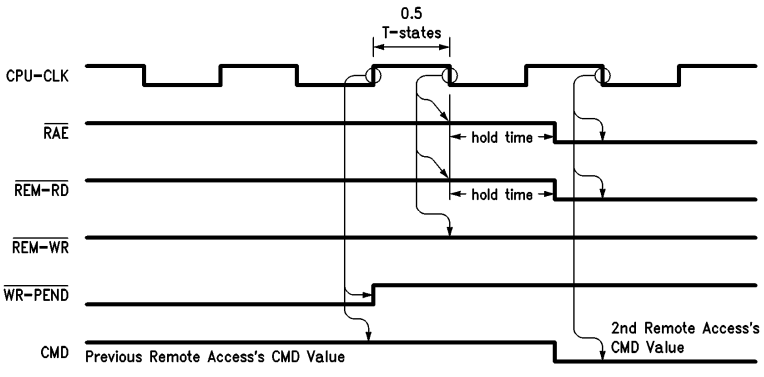
(b) This timing diagram shows a remote access violating remote rest time. The first set of arrows shows the BCP sampling the value of CMD when WR-PEND rises. If a remote access begins after WR-PEND rises and before the position of the second set of arrows (0.5 T-states later), the value of CMD will not be sampled again. The value of CMD has not changed since WR-PEND rose, so the BCP will read the intended location.

FIGURE 4-26. Rest Time for Latched Write Mode



TL/F/9336–H3

(c) This timing diagram shows a remote access setting up in time for WR-PEND rising to latch in the proper value of CMD. The only set of arrows shows the BCP sampling the second remote access's CMD value when WR-PEND rises. The value of CMD will not be sampled again. The BCP will carry out the second remote access as it was intended.



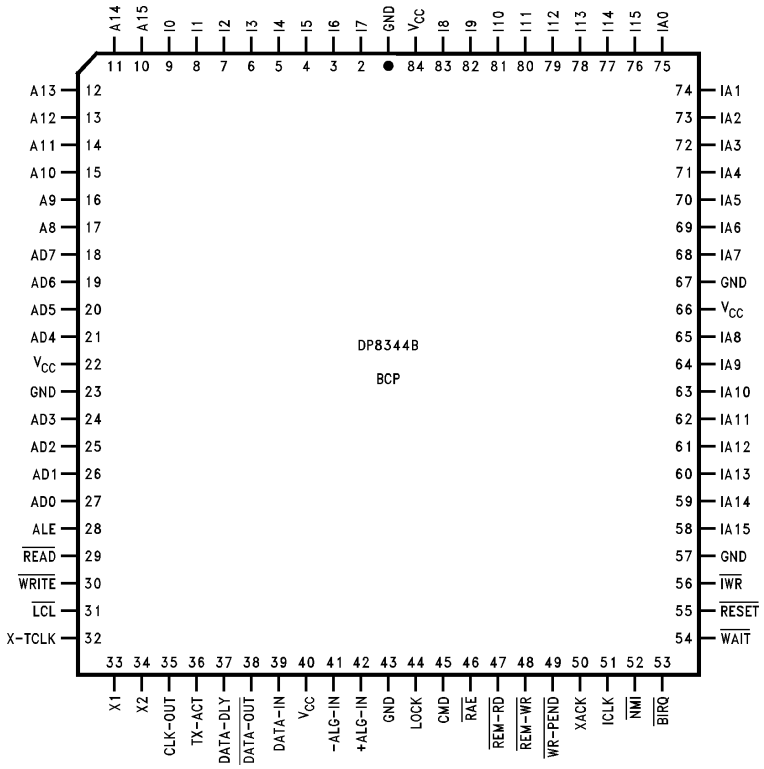
TL/F/9336–H4

(d) This timing diagram shows a remote access starting after a half T-state plus a hold time since WR-PEND rose. The first set of arrows shows the BCP sampling the value of CMD when WR-PEND rises. The second set of arrows shows the BCP recognizing that no remote access has started and the value of CMD will be sampled for the next remote access. The third set of arrows shows the BCP sampling the correct value of CMD for the second remote access. The BCP will carry out the second remote access as it was intended.

FIGURE 4-26. Rest Time for Latched Write Mode (Continued)

5.0 Device Specifications

Plastic Chip Carrier



TL/F/9336–2

FIGURE 5-1. Top View

Order Number DP8344B
See NS Package Number V84A

5.1 PIN DESCRIPTIONS

Signal	In/Out	Pin	Reset State	Description
5.1.1 TIMING/CONTROL SIGNALS				
X1 X2	In Out	33 34	X $\overline{X1}$	Input and output of the on-chip crystal oscillator amplifier. Connect a crystal across these pins, or apply an external clock to X1, with X2 left open.
CLK-OUT	Out	35	X1	Buffered **CL**oc**K** oscillator **OUT**put, at the crystal frequency.
X-TCLK	In	32	X	E**X**ternal **T**ransceiver **CL**oc**K** input.
$\overline{WAIT}$	In	54	X	CPU **WAIT**. When active, waits processor and remote interface controller.
$\overline{RESET}$	In	55	0	Master **RESET**. Parallel reset to all sections of the chip.
5.1.2 INSTRUCTION MEMORY INTERFACE **Instruction Address Bus:**				
IA15 (MSB) IA14 IA13 IA12 IA11 IA10	Out Out Out Out Out Out	58 59 60 61 62 63	0 0 0 0 0 0	16-bit **I**nstruction memory **A**ddress bus.

Signal	In/Out	Pin	Reset State	Description
5.1.2 INSTRUCTION MEMORY INTERFACE (Continued) **Instruction Address Bus:** (Continued)				
IA9	Out	64	0	16-bit **I**nstruction memory **A**ddress bus.
IA8	Out	65	0	
IA7	Out	68	0	
IA6	Out	69	0	
IA5	Out	70	0	
IA4	Out	71	0	
IA3	Out	72	0	
IA2	Out	73	0	
IA1	Out	74	0	
IA0 (LSB)	Out	75	0	
Instruction Bus:				
I15 (MSB)	In/Out	76	In	16-bit **I**nstruction memory data bus.
I14	In/Out	77	In	
I13	In/Out	78	In	
I12	In/Out	79	In	
I11	In/Out	80	In	
I10	In/Out	81	In	
I9	In/Out	82	In	
I8	In/Out	83	In	
I7	In/Out	2	In	
I6	In/Out	3	In	
I5	In/Out	4	In	
I4	In/Out	5	In	
I3	In/Out	6	In	
I2	In/Out	7	In	
I1	In/Out	8	In	
I0 (LSB)	In/Out	9	In	
Timing Control:				
$\overline{IWR}$	Out	56	1	**I**nstruction **WR**ite. Instruction memory write strobe.
ICLK	Out	51	0	**I**nstruction **CL**oc**K**. Delimits instruction fetch cycles. Rises during the first half of T1, signifying the start of an instruction cycle, and falls when the next instruction address is valid.
5.1.3 DATA MEMORY INTERFACE **Address Bus:**				
A15 (MSB)	Out	10	X	High byte of 16-bit memory **A**ddress.
A14	Out	11	X	
A13	Out	12	X	
A12	Out	13	X	
A11	Out	14	X	
A10	Out	15	X	
A9	Out	16	X	
A8	Out	17	X	
Multiplexed Address/Data Bus:				
AD7	In/Out	18	1	Low byte of 16-bit data memory **A**ddress, multiplexed with 8-bit **D**ata bus.
AD6	In/Out	19	0	
AD5	In/Out	20	0	
AD4	In/Out	21	0	
AD3	In/Out	24	0	
AD2	In/Out	25	0	
AD1	In/Out	26	0	
AD0 (LSB)	In/Out	27	1	

5.0 Device Specifications (Continued)

Signal	In/Out	Pin	Reset State	Description
5.1.3 DATA MEMORY INTERFACE (Continued) **Timing/Control:**				
ALE	Out	28	0	**A**ddress **L**atch **E**nable. Demultiplexes AD bus. Address should be latched on the falling edge.
$\overline{\text{READ}}$	Out	29	1	Data memory **READ** strobe. Data is latched on the rising edge.
$\overline{\text{WRITE}}$	Out	30	1	Data memory **WRITE** strobe. Data is presented on the rising edge.
5.1.4 TRANSCEIVER INTERFACE				
DATA-IN	In	39	X	Logic level serial **DATA IN**put.
+ ALG-IN	In	42	X	Non-inverting **A**na**L**o**G IN**put for biphase serial data.
− ALG-IN	In	41	X	Inverting **A**na**L**o**G IN**put for biphase serial data.
$\overline{\text{DATA-OUT}}$	Out	38	1	Biphase serial **DATA OUT**put (inverted).
DATA-DLY	Out	37	1	Biphase serial **DATA** output **D**e**L**a**Y**ed by one-quarter bit time.
TX-ACT	Out	36	0	**T**ransmitter **ACT**ive. Normally low, goes high to indicate serial data is being transmitted. Used to enable external line drive circuitry.
5.1.5 REMOTE INTERFACE				
$\overline{\text{RAE}}$	In	46	X	**R**emote **A**ccess **E**nable. A ''chip-select'' input to allow host access of BCP functions and memory.
CMD	In	45	X	**C**o**M**man**D** input. When high, remote accesses are directed to the Remote Interface Configuration register {RIC}. When low, remote accesses are directed to data-memory, instruction-memory or program counter as determined by {RIC}.
$\overline{\text{REM-RD}}$	In	47	X	**REM**ote **R**ea**D**. When active along with $\overline{\text{RAE}}$, a remote read cycle is requested; serviced by the BCP when the data bus becomes available.
$\overline{\text{REM-WR}}$	In	48	X	**REM**ote **WR**ite. When active along with $\overline{\text{RAE}}$, a remote write cycle is requested; serviced by the BCP when the data bus becomes available.
XACK	Out	50	1	Transfer **ACK**nowledge. Normally high, goes low on $\overline{\text{REM-RD}}$ or $\overline{\text{REM-WR}}$ going low (if $\overline{\text{RAE}}$ low), returning high when the transfer is complete. Normally used as a ''wait'' signal to a remote processor.
$\overline{\text{WR-PEND}}$	Out	49	1	**WR**ite **PEND**ing. In a system configuration where remote write cycles are latched, indicates when the latches contain valid data which is yet to be serviced by the BCP.
$\overline{\text{LOCK}}$	In	44	X	The remote processor uses this input to **LOCK** out local (BCP) accesses to data-memory. Once the remote processor has been granted the bus, $\overline{\text{LOCK}}$ gives it sole access to the bus and BCP accesses are ''waited''.
$\overline{\text{LCL}}$	Out	31	0	**L**o**C**a**L**. Normally low, goes high when the BCP relinquishes the data and address bus to service a Remote Access.
5.1.6 EXTERNAL INTERRUPTS				
$\overline{\text{BIRQ}}$	In/Out	53	In	**B**i-directional **I**nterrupt **R**e**Q**uest. As an input, can be used as an active low interrupt input (maskable and level-sensitive). As an output, can be used to generate remote system interrupts, reset via {RIC}.
$\overline{\text{NMI}}$	In	52	X	**N**on-**M**askable **I**nterrupt. Negative edge sensitive interrupt input.

5.0 Device Specifications (Continued)

5.2 ABSOLUTE MAXIMUM RATINGS (Notes 1 & 2)

If Military/Aerospace specified devices are required, please contact the National Semiconductor Sales Office/Distributors for availability and specifications.

Supply Voltage (V_{CC})	−0.5V to +7.0V
DC Input Voltage (V_{IN}) or DC Input Diode Current	−0.5V to V_{CC} + 0.5V ±20 mA
DC Output Voltage (V_{OUT}) or DC Output Current, per Pin (I_{OUT})	−0.5V to V_{CC} + 0.5V ±20 mA
DC V_{CC} or GND Current, per Pin	±50 mA
Storage Temperature Range (T_{STG})	−65°C to +150°C
Power Dissipation (PD)	500 mW
Lead Temperature (Soldering, 10 sec)	260°C
ESD Tolerance: C_{ZAP} = 120 pF, R_{ZAP} = 1500Ω	2.0 kV

5.3 OPERATING CONDITIONS

	Min	Max	Units
Supply Voltage (V_{CC})	4.5	5.5	V
DC Input or Output Voltage (V_{IN}, V_{OUT})	0.0	V_{CC}	V
Operating Temp. Range (T_A)	0	70	°C
Input Rise or Fall Times (t_r, t_f)		500	ns
Oscillator Crystal R_S		20	Ω
V_{CC} Power Up Ramp	6		ms

DC ELECTRICAL CHARACTERISTICS V_{CC} = 5V ±10% (unless otherwise specified)

Symbol	Parameter	Conditions	Guaranteed Limits 0–70°C	Units
V_{IH}	Minimum High Level Input Voltage X1 (Note 3) All Other Inputs Except −ALG-IN, +ALG-IN		 3.5 2.0	 V V
V_{IL}	Maximum Low Level Input Voltage X1 (Note 3) All Other Inputs Except −ALG-IN, +ALG-IN		 1.7 0.8	 V V
$V_{IH}-V_{IL}$	Minimum DATA-IN Hysteresis		0.1	V
V_{SENS}	Minimum Analog Input IN+, IN− Differential Sensitivity	*Figure 5-8b*	20	mV
V_{BIAS}	Common Mode Analog Input Bias Voltage	User Provided Bias Voltage	Min 2.25 Max 2.75	V V
V_{OH}	Minimum High Level Output Voltage IA, A, AD All Other Outputs	V_{IN} = V_{IH} or V_{IL} $\|I_{OUT}\|$ = 20 μA $\|I_{OUT}\|$ = 4.0 mA, V_{CC} = 4.5V $\|I_{OUT}\|$ = 1.0 mA, V_{CC} = 4.5V	 V_{CC} − 0.1 3.5 3.5	 V V V
V_{OL}	Maximum Low Level Output Voltage IA, A, AD All Other Outputs	V_{IN} = V_{IH} or V_{IL} $\|I_{OUT}\|$ = 20 μA $\|I_{OUT}\|$ = 4.0 mA, V_{CC} = 4.5V $\|I_{OUT}\|$ = 1.0 mA, V_{CC} = 4.5V	 0.1 0.4 0.4	 V V V
I_{IN}	Maximum Input Current	V_{IN} = V_{CC} or GND −ALG-IN, +ALG-IN X1 (Note 3) All Others	 ±10 ±20 ±10	 μA μA μA
I_{OZ}	Maximum TRI-STATE® Output Leakage Current	V_{OUT} = V_{CC} or GND	±10	μA
I_{CC}	Maximum Operating Supply Current Total to 4 V_{CC} Pins (Note 4)	V_{IN} = V_{CC} or GND TCLK = 8 MHz, CPU-CLK = 16 MHz Xcvr and CPU Operating Xcvr Idle, CPU Waited V_{IN} = V_{CC} or GND TCLK = 20 MHz, CPU-CLK = 20 MHz Xcvr and CPU Operating Xcvr Idle, CPU Waited	 61 29 71 31	 mA mA mA mA

Note 1: Absolute Maximum Ratings are those values beyond which damage to the device may occur.

Note 2: Unless otherwise specified, all voltages are referenced to ground.

Note 3: X2 is an internal node with ESD protection. Do not use other than with crystal oscillator application.

Note 4: No DC loading, with X1 driven, no crystal. AC load per Test Circuit for Output Tests.

5.0 Device Specifications (Continued)

5.5 SWITCHING CHARACTERISTICS

The following specifications apply for V_{CC} = 4.5V to 5.5V, T_A = 0°C to 70°C.

5.5.1 Definitions

The timing specifications for the BCP are provided in the following tables and figures. The tables consist of five sections which are the following: the timing parameter symbol, the parameter ID#, the parameter description, the formula for the parameter, and the timing specification for the parameter. Below each table is a figure containing the waveforms for the parameters in the table.

The parameter symbol is composed of the type of timing specification and the signal or signals involved. Note that the symbols are unique only within a given table. The following symbol conventions are used for the type of timing specification.

- t_W — Pulse width specification
- t_{PD} — Propagation delay specification
- t_H — Hold time specification
- t_{SU} — Setup time specification
- t_{ZA} — High impedance to active delay specification (enable time)
- t_{AZ} — Active to high impedance delay specification (disable time)
- t_{ACC} — Access time specification
- t_T — Clock period specification

The parameter ID# is used to cross reference the timing parameter to the appropriate timing relationship in the accompanying figure. The waveforms in the figures are shown with the CPU clock running full speed ([CCS] = 0). For this case, CPU-CLK and CLK-OUT are equivalent. If CPU-CLK/2 is selected ([CCS] = 1), the effect on the waveforms with CLK-OUT is for CLK-OUT to double in frequency. The same is true for waveforms with X1. Note that CLK-OUT is always running at the crystal frequency and it is the CPU-CLK that is changing to half speed.

The parameter description defines the timing relationship being specified. BCP pin references are capitalized in the description.

Many of the timing specifications are dependent on variables such as operating frequency and number of programmed wait states. The formula for the parameter allows an accurate timing specification to be calculated for any combination of these variables. The formula represents the part of the timing specification that is synchronized to the internal CPU clock. This value is calculated and then added to the value specified under the Min or Max column to create the minimum or maximum guaranteed timing specification for the parameter.

The following acronyms are used in the tables:

DMEM refers to data memory

IMEM refers to instruction memory

RIC refers to the Remote Interface Control register

PC refers to the BCP Program Counter

T refers to the CPU clock period in ns

T_H refers to first half pulse width (high time) of the CPU clock in ns

T_L refers to second half pulse width (low time) of the CPU clock in ns.

C refers to the transceiver clock period in ns

n_{IW} is the number of instruction memory wait states programmed in DCR

n_{DW} is the number of data memory wait states programmed in DCR

n_{LW} is the number of remote wait states due to a BCP local data memory access

n_{RW} is the number of CPU wait states due to a remote access

MAX(A,B) means take the greater value of A or B

The following table is an example of the format used for the timing specifications. In this example, $t_{W\text{-}RD}$ indicates a pulse width specification for the output pin $\overline{READ}$. The ID# for locating the parameter in the timing waveforms is 10. The formula for this specification involves data and instruction memory wait states and the CPU clock period. For the case of 3 data memory wait states and 0 instruction memory wait states and a CPU clock period of 50 ns, the READ low minimum pulse width would be calculated as:

$(MAX(3,0-1)+1)T+(-10) = 4T - 10 = 190$ ns

For the case of 1 data memory wait state and 3 instruction memory wait states and a CPU clock period of 50 ns, the $\overline{READ}$ low minimum pulse width would be calculated as:

$(MAX(1,3-1)+1)T+(-10) = 3T - 10 = 140$ ns

To calculate n_{LW} the following two equations are needed:

n_{LW} (min) = 0

n_{LW} (max) = MAX(n_{DW}, n_{IW}−1) + Data Memory Access Cycle

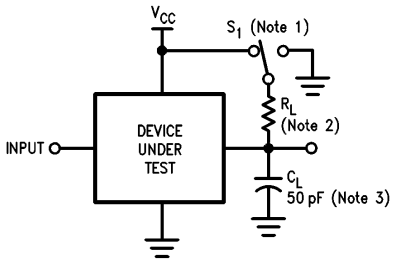
Data Memory Access Cycle is normally 3 T-states if [4TR] = 0 and 4 T-states if [4TR] = 1. Keep in mind that both [LOR] and $\overline{WAIT}$ can extend n_{LW}.

Symbol	ID#	Parameter	Formula	Min	Max	Units
$t_{W\text{-}RD}$	10	$\overline{Read}$ Low	$(MAX(n_{DW},n_{IW}-1)+1)T+$	−10	10	ns

5.0 Device Specifications (Continued)

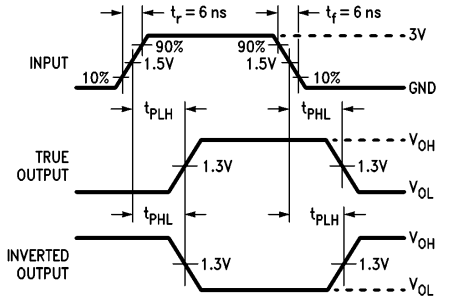
Note 1: S_1 = V_{CC} for t_{PZL}, and t_{PLZ} measurements
S_1 = GND for t_{PZH}, and t_{PHZ} measurements
S_1 = Open for push pull outputs
Note 2: R_L = 1.1k for 4 mA outputs
R_L = 4.4k for 1 mA outputs
Note 3: C_L includes scope and jig capacitance.

Test Circuit for Output Tests



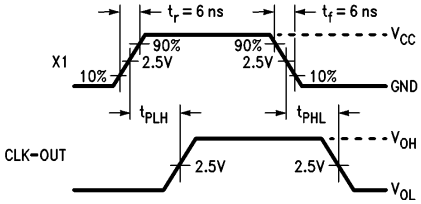
TL/F/9336–A2

Propagation Delay Waveforms Except for Oscillator



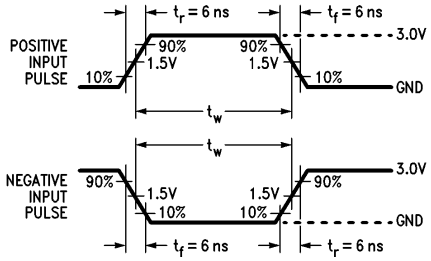
TL/F/9336–A3

Propagation Delay Waveform for Oscillator



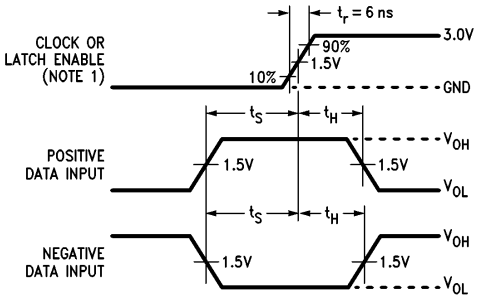
TL/F/9336–A4

Input Pulse Width Waveforms



TL/F/9336–A5

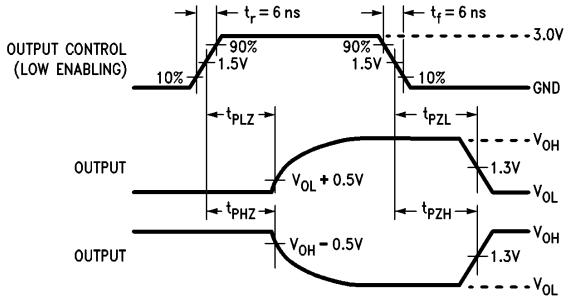
Setup and Hold Time Waveforms



TL/F/9336–A6

Note 1: Waveform for negative edge sensitive circuits will be inverted.

TRI-STATE Output Enable and Disable Waveforms



TL/F/9336–A7

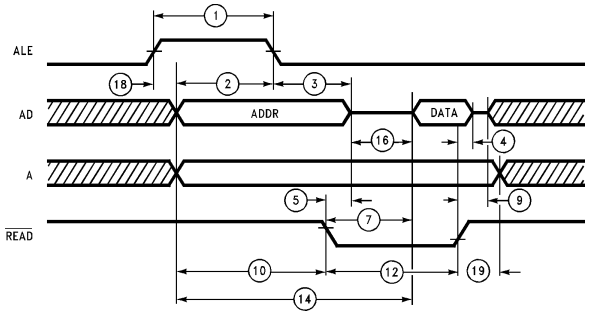
FIGURE 5-2. Switching Characteristic Measurement Waveforms

5.0 Device Specifications (Continued)

TABLE 5-3. Data Memory Read Timing (Note 1)

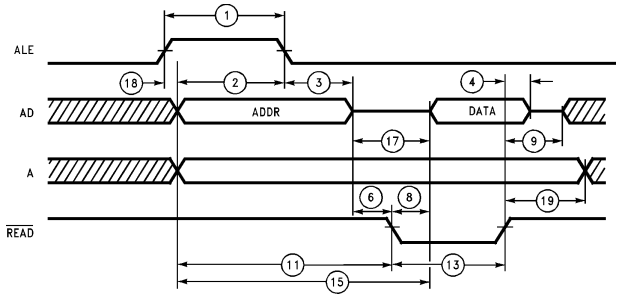
Symbol	ID#	Parameter	Formula	Min	Max	Units
$t_{W\text{-}ALE}$	1	ALE High	$(n_{RW}+1)T+$	−10	12	ns
$t_{PD\text{-}AAD\text{-}ALE}$	2	A, AD (Data Address) Valid to ALE Falling	$T+$	−22		ns
$t_{PD\text{-}ALE\text{-}AD}$	3	ALE Falling to AD (Data Address) Invalid	T_L+	−2		ns
$t_{H\text{-}RD\text{-}DATA}$	4	Data Valid after $\overline{READ}$ Rising		0		ns
$t_{AZ\text{-}RD\text{-}AD}$	5	$\overline{READ}$ Falling to AD Disabled ([4TR] = 0)			20	ns
$t_{AZ\text{-}AD\text{-}RD}$	6	AD Disabled before $\overline{READ}$ Falling ([4TR] = 1)	T_H+	−20		ns
$t_{SU\text{-}RD\text{-}DATA}$	7	$\overline{READ}$ Falling to AD (Data) Setup ([4TR] = 0)	$(MAX(n_{DW},n_{IW}-1)+1)T+$		−22	ns
$t_{SU\text{-}RD\text{-}DATA}$	8	$\overline{READ}$ Falling to AD (Data) Setup ([4TR] = 1)	$(MAX(n_{DW}-1,n_{IW}-1)+1)T+T_L+$		−21	ns
$t_{ZA\text{-}RD\text{-}AD}$	9	$\overline{READ}$ Rising to AD Enabled	T_H+	−2		ns
$t_{PD\text{-}AAD\text{-}RD}$	10	A, AD (Data Address) Valid before $\overline{READ}$ Falling ([4TR] = 0)	$T+T_L+$	−27		ns
$t_{PD\text{-}AAD\text{-}RD}$	11	A, AD (Data Address) Valid before $\overline{READ}$ Falling ([4TR] = 1)	$2T+$	−27		ns
$t_{W\text{-}RD}$	12	$\overline{READ}$ Low ([4TR] = 0)	$(MAX(n_{DW},n_{IW}-1)+1)T+$	−10	10	ns
$t_{W\text{-}RD}$	13	$\overline{READ}$ Low ([4TR] = 1)	$(MAX(n_{DW}-1,n_{IW}-1)+1)T+T_L+$	−10	10	ns
$t_{ACC\text{-}D}$	14	Data Memory Read Time ([4TR]) = 0)	$(MAX(n_{DW},n_{IW}-1)+2)T+T_L+$		−40	ns
$t_{ACC\text{-}D}$	15	Data Memory Read Time ([4TR]) = 1)	$(MAX(n_{DW}-1,n_{IW}-1)+3)T+T_L+$		−40	ns
$t_{SU\text{-}AD\text{-}DATA}$	16	AD Disabled to AD (Data) Setup ([4TR] = 0)	$(MAX(n_{DW},n_{IW}-1)+1)T+$	−33		ns
$t_{SU\text{-}AD\text{-}DATA}$	17	AD Disabled to AD (Data) Setup ([4TR] = 1)	$(MAX(n_{DW}-1,n_{IW}-1)+2)T+$	−33		ns
$t_{PD\text{-}ALE\text{-}AAD}$	18	ALE Rising to A, AD (Data Address) Valid	$(n_{RW})T+$		24	ns
$t_{PD\text{-}RD\text{-}A}$	19	$\overline{READ}$ Rising to A Invalid	T_H+	0		ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.



TL/F/9336–52

(a) Read Timing with ([4TR] = 0)



TL/F/9336–H7

(b) Read Timing with ([4TR] = 1)

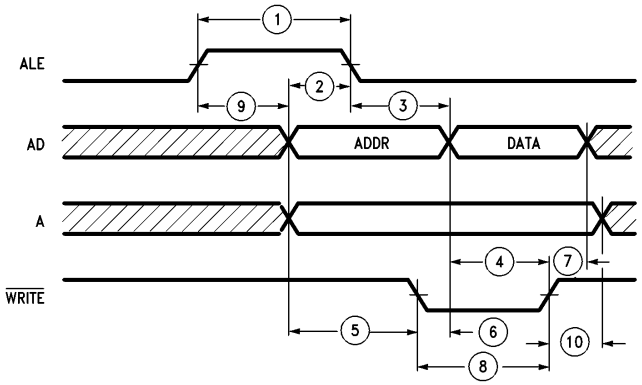
FIGURE 5-3. Data Memory Read Timing

5.0 Device Specifications (Continued)

TABLE 5-4. Data Memory Write Timing (Note 1)

Symbol	ID#	Parameter	Formula	Min	Max	Units
$t_{W\text{-}ALE}$	1	ALE High	$(n_{RW}+1)T+$	−10	12	ns
$t_{PD\text{-}AAD\text{-}ALE}$	2	A, AD (Data Address) Valid to ALE Falling	$T+$	−22		ns
$t_{PD\text{-}ALE\text{-}AD}$	3	ALE Falling to AD (Data Address) Invalid	T_L+	−2		ns
$t_{PD\text{-}DATA\text{-}WR}$	4	AD (Data) Valid to $\overline{\text{WRITE}}$ Rising	$(\text{MAX}(n_{DW},n_{IW}-1)+1)T+$	−20		ns
$t_{PD\text{-}AAD\text{-}WR}$	5	A, AD (Data Address) Valid to $\overline{\text{WRITE}}$ Falling	$1.5T+$	−28		ns
$t_{PD\text{-}WR\text{-}DATA}$	6	$\overline{\text{WRITE}}$ Falling to AD (Data) Valid			19	ns
$t_{PD\text{-}WR\text{-}DATAz}$	7	$\overline{\text{WRITE}}$ Rising to AD (Data) Invalid	T_H+	−4		ns
$t_{W\text{-}WR}$	8	$\overline{\text{WRITE}}$ Low	$(\text{MAX}(n_{DW},n_{IW}-1)+1)T+$	−10	10	ns
$t_{PD\text{-}ALE\text{-}AAD}$	9	ALE Rising to A, AD (Data Address) Valid	$(n_{RW})T+$		24	ns
$t_{PD\text{-}WR\text{-}A}$	10	$\overline{\text{WRITE}}$ Rising to A Invalid	T_H+	−2		ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.



TL/F/9336–53

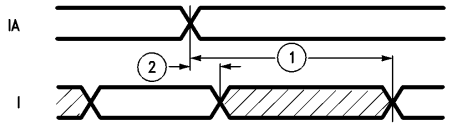
FIGURE 5-4. Data Memory Write Timing

5.0 Device Specifications (Continued)

TABLE 5-5. Instruction Memory Read Timing (Note 1)

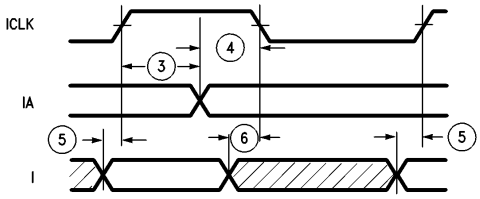
Symbol	ID#	Parameter	Formula	Min	Max	Units
t_{ACC-I}	1	Instruction Memory Read Time	$(n_{IW}+1)T+T_L+$		−19	ns
t_{H-IA-I}	2	IA Invalid to I Invalid		0		ns
$t_{PD-ICLK-IA}$	3	ICLK Rising to IA Invalid	T_H+	−13		ns
$t_{PD-IA-ICLK}$	4	Next IA Valid before ICLK Falling	T_L+	−12		ns
$t_{PD-IAz-ICLK}$		IA Invalid before ICLK Falling			17	ns
$t_{SU-I-ICLK}$	5	I Valid before ICLK Rising		20		ns
$t_{H-I-ICLK}$	6	I Invalid before ICLK Falling	T_L+		0	ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.



TL/F/9336–A9

(a) Instruction Memory Read Timing



TL/F/9336–54

(b) Instruction ICLK Timing

FIGURE 5-5. Instruction Memory Timing

5.0 Device Specifications (Continued)

TABLE 5-6. Clock Timing (Note 1)

Symbol	ID #	Parameter	Formula	Min	Max	Units
$t_{T\text{-}X1}$	1	X1 Period (Note 2)		50	500	ns
$t_{PD\text{-}X1\text{-}CO}$	2	X1 to CLK-OUT (Note 2)			37	ns
$t_{PD\text{-}CO\text{-}ICLKr}$	3	CLK-OUT Rising to ICLK Rising			15	ns
$t_{PD\text{-}CO\text{-}ICLKf}$	4	CLK-OUT Rising to ICLK Falling (Note 3)			15	ns
$t_{T\text{-}XT}$	5	X-TCLK Period (Note 4)		50	500	ns
$t_{W\text{-}X1HL}$	6	X1 High and Low time Pulse Widths (Note 5)		21		ns
$t_{W\text{-}XTHL}$	7	XTCLK High and Low Time Pulse Widths		15		ns

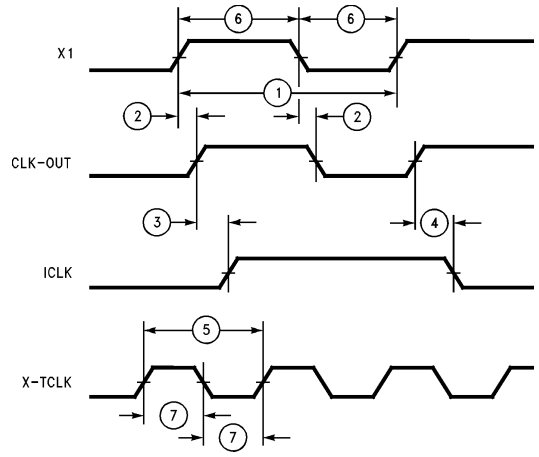
Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

Note 2: Measurement thresholds at 2.5V.

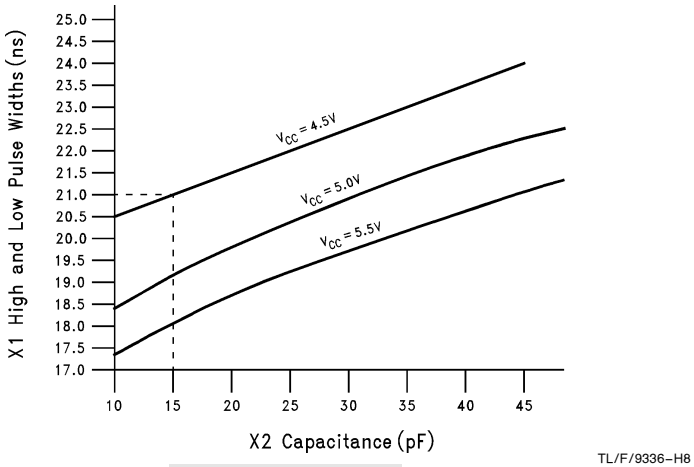
Note 3: The falling edge of ICLK occurs only after the next IA becomes valid. The CLK-OUT cycle in which this occurs depends on the instruction being executed and the number of programmed instruction wait states.

Note 4: There is no relationship between X1 and X-TCLK. X-TCLK is fully asynchronous.

Note 5: External loading on pin X2 equal to 15 pF. See *Figure 5-6b* for affect of X2 loading in non-crystal applications (i.e., an external oscillator driving X1).



TL/F/9336–55



TL/F/9336–H8

FIGURE 5-6. Clock Timing

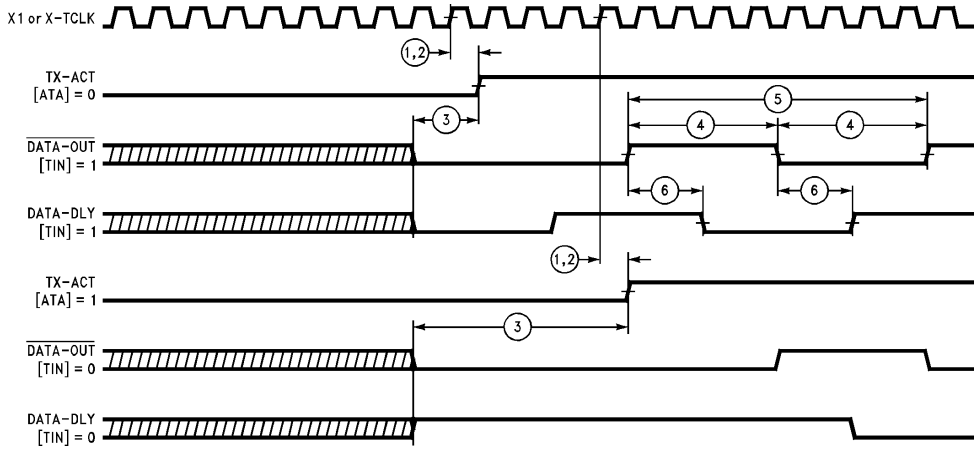
5.0 Device Specifications (Continued)

TABLE 5-7. Transceiver Timing (Note 1)

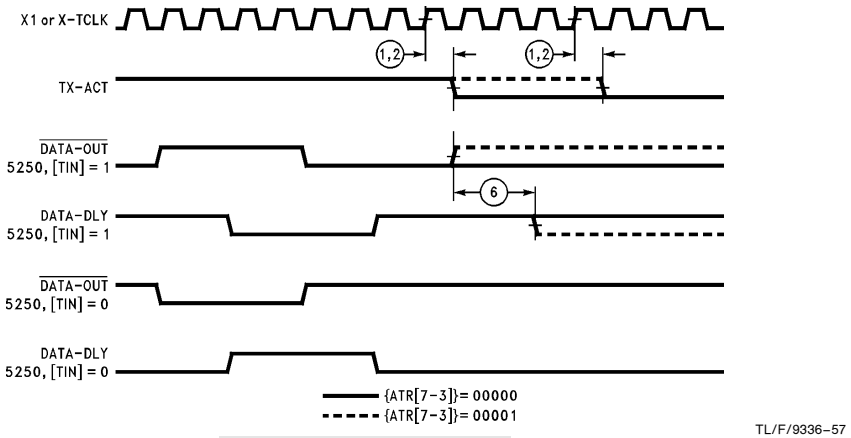
Symbol	ID #	Parameter	Formula	Min	Max	Units
$t_{PD-X1-TA}$	1	X1 Rising to TX-ACT Rising/Falling		10	65	ns
$t_{PD-XTCLK-TA}$	2	X-TCLK Rising to TX-ACT Rising/Falling		7	49	ns
$t_{PD-DODD-TA}$	3	$\overline{\text{DATA-OUT}}$, DATA-DLY Valid to TX-ACT Rising	C+	16		ns
$t_{W-DO-HB}$	4	$\overline{\text{DATA-OUT}}$ Half Bit Cell Width	4C+	−10	10	ns
$t_{W-DO-FB}$	5	$\overline{\text{DATA-OUT}}$ Full Bit Cell Width	8C+	−10	10	ns
$t_{PD-DO-DD}$	6	$\overline{\text{DATA-OUT}}$ Falling/Rising to DATA-DLY Rising/Falling (Note 3)	2C+	−10	10	ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

Note 2: When [ATA] = 1, TX-ACT is delayed by 4C and an additional line quiescent is generated resulting in 5½ line quiescent pulses after the line interface logic. The additional delay relative to a message with [ATA] = 0 is 8C (one bit time).



(a) Transmission Beginning Timing (Note 2)



(b) Transmission Ending Timing

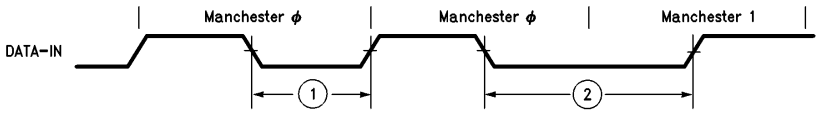
FIGURE 5-7. Transceiver Timing

5.0 Device Specifications (Continued)

TABLE 5-8. Analog and DATA-IN Timing (Note 1)

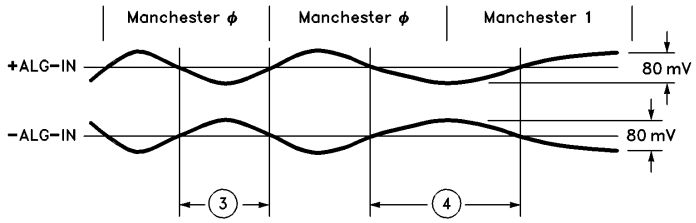
Symbol	ID #	Parameter	Formula	Min	Max	Units
$t_{W\text{-}DI\text{-}hb}$	1	DATA-IN Data, Half Bit Width	3C +	12		ns
			5C +		−12	ns
$t_{W\text{-}DI\text{-}fb}$	2	DATA-IN Data, Full Bit Width	7C +	12		ns
			9C +		−12	ns
$t_{W\text{-}AI\text{-}hb}$	3	Analog Data, Half Bit Width (−ALG-IN or +ALG-IN)	3C +	20		ns
			5C +		−20	ns
$t_{W\text{-}AI\text{-}fb}$	4	Analog Data, Full Bit Width (−ALG-IN or +ALG-IN)	7C +	20		ns
			9C +		−20	ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.



TL/F/9336–58

(a) DATA-IN Jitter Timing (3270)



TL/F/9336–59

(b) Analog Jitter Timing (3270)

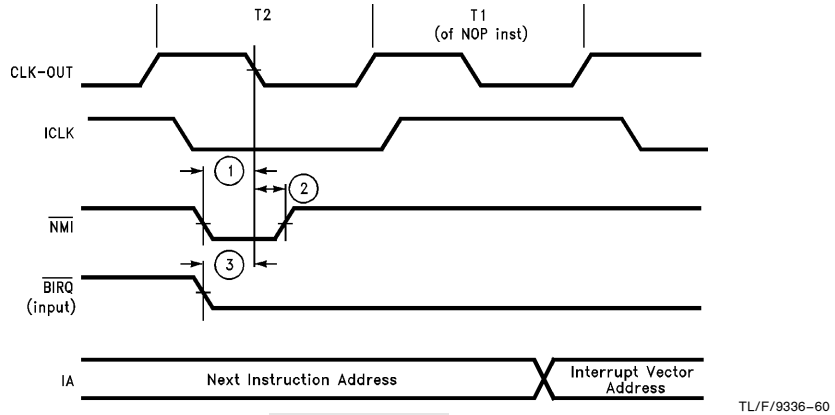
FIGURE 5-8. Analog and DATA-IN Timing

5.0 Device Specifications (Continued)

TABLE 5-9. Interrupt Timing (Note 1)

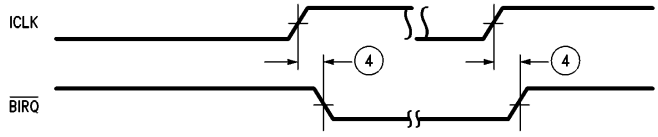
Symbol	ID #	Parameter	Formula	Min	Max	Units
$t_{SU-NMI-CO}$	1	$\overline{NMI}$ Falling before CLK-OUT Falling		12		ns
$t_{H-NMI-CO}$	2	$\overline{NMI}$ Hold after CLK-OUT Falling		8		ns
$t_{SU-BQ-CO}$	3	$\overline{BIRQ}$ (Input) Falling before CLK-OUT Falling		13		ns
$t_{PD-ICLK-BQ}$	4	ICLK Rising to $\overline{BIRQ}$ (Output) Rising/Falling			24	ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.



TL/F/9336–60

(a) Interrupt Timing



TL/F/9336–61

(b) $\overline{BIRQ}$ Output Timing

FIGURE 5-9. Interrupt Timing

5.0 Device Specifications (Continued)

TABLE 5-10. Control Pin Timing (Note 1)

Symbol	ID #	Parameter	Formula	Min	Max	Units
$t_{W\text{-}RST}$	1	$\overline{RESET}$ Low	5T +	0		ns
$t_{PD\text{-}RST\text{-}ICLK}$	2	$\overline{RESET}$ Rising to ICLK Rising	4T +		0	ns
$t_{SU\text{-}ALE\text{-}WT}$	3	$\overline{WAIT}$ Low after ALE High to Extend Cycle	$(MAX(n_{DW},n_{IW}-1)+1)T+$		−21	ns
$t_{H\text{-}WT\text{-}ALE}$	4	$\overline{WAIT}$ Rising after ALE Falling (Note 2)		0		ns
			$(MAX(n_{DW},n_{IW}-1)+1)T+$		−28	ns
$t_{PD\text{-}WT\text{-}RDWR}$	5	$\overline{WAIT}$ Rising to $\overline{READ}$ or $\overline{WRITE}$ Rising	$T+T_L+$	−22		ns
			$2T+T_L+$		2	ns
$t_{SU\text{-}RRW\text{-}RST}$	6	$\overline{REM\text{-}RD}$, $\overline{REM\text{-}WR}$ Low to $\overline{RESET}$ Rising for BCP to Start		15		ns
$t_{H\text{-}RST\text{-}RRW}$	7	$\overline{REM\text{-}RD}$, $\overline{REM\text{-}WR}$ Low after $\overline{RESET}$ Rising for BCP to Start		5		ns
$t_{SU\text{-}LK\text{-}ICLK}$	8	$\overline{LOCK}$ Low before ICLK High (Note 3)	T_L+	19		ns
$t_{PD\text{-}LK\text{-}ALE}$	9	$\overline{LOCK}$ High to ALE Low	T +	−2		ns
			3T +		20	ns
$t_{SU\text{-}WT\text{-}ICLK}$	10	$\overline{WAIT}$ Low after ICLK Rising to Extend Cycle (Note 4)	$(MAX(n_{DW},n_{IW}-1))T+T_H+$		−22	ns
$t_{H\text{-}WT\text{-}ICLK}$	11	$\overline{WAIT}$ High after ICLK Rising (Notes 2, 4)	$(MAX(n_{DW},n_{IW}-1))T+T_H+$	2		ns
			$(MAX(n_{DW},n_{IW}-1)+1)T+T_H+$		−20	ns
$t_{H\text{-}LK\text{-}ICLK}$	12	$\overline{LOCK}$ Rising after ICLK High	T_H+	2		ns
$t_{PD\text{-}AD\text{-}ALE}$	13	AD to ALE Falling after $\overline{LOCK}$ Rising	T +	−33		ns
$t_{SU\text{-}WT\text{-}ALEf}$	14	$\overline{WAIT}$ Low before ALE Falling to Extend Cycle		23		ns

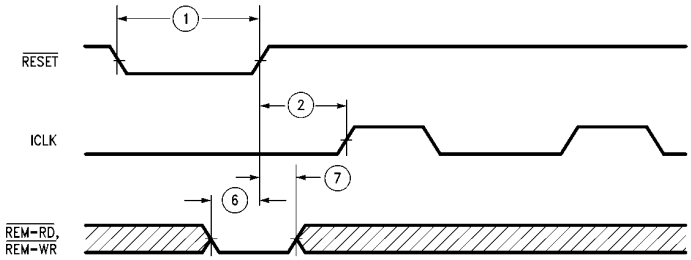
Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

Note 2: The maximum value for this parameter is the lastest $\overline{WAIT}$ can be removed without adding an additional T-state. The formula assumes a minimum externally generated wait of one T-state.

Note 3: If $t_{SU\text{-}LK\text{-}ICLK}$ is not met, the maximum time from $\overline{LOCK}$ low till no more local accesses is $(MAX(n_{DW}, n_{IW}-1)+3)T$.

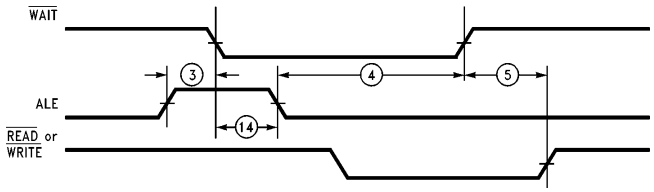
Note 4: The formula(s) apply to a 2 T-state instruction. For a 3 T-state instruction, add one T-state; for a 4 T-state instruction, add two T-states.

5.0 Device Specifications (Continued)



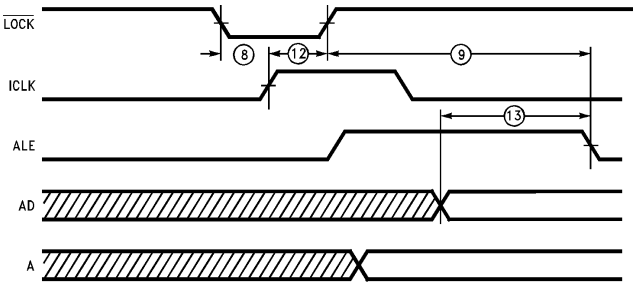
TL/F/9336–62

(a) Reset Timing



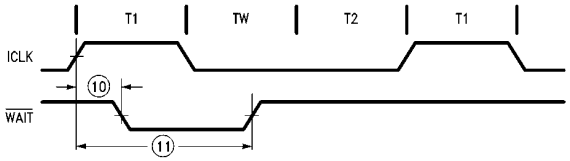
TL/F/9336–63

(b) BCP Access $\overline{\text{WAIT}}$ Timing



TL/F/9336–64

(c) $\overline{\text{LOCK}}$ Timing



TL/F/9336–A8

(d) Instruction $\overline{\text{WAIT}}$ Timing

FIGURE 5-10. Control Pin Timing

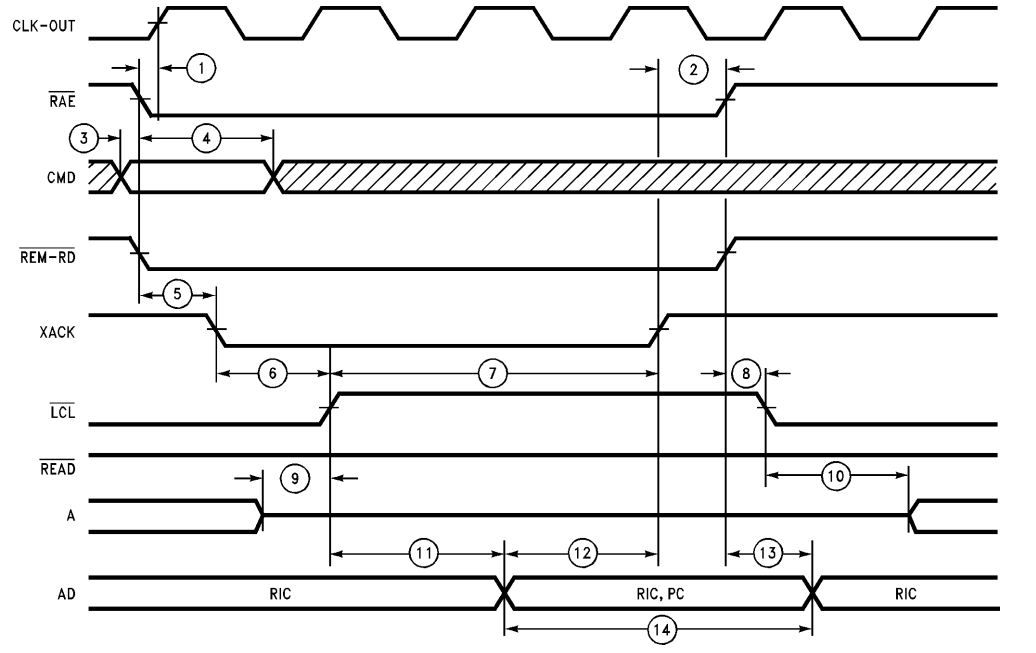
5.0 Device Specifications (Continued)

TABLE 5-11. Buffered Read of PC, RIC (Note 1)

Symbol	ID #	Parameter	Formula	Min	Max	Units
$t_{SU-RRR-CO}$	1	$\overline{RAE}$, $\overline{REM-RD}$ Falling before CLK-OUT Rising		22		
$t_{H-RRR-X}$	2	$\overline{RAE}$, $\overline{REM-RD}$ Rising after XACK Rising (Note 2)		0		ns
			2T+		−34	ns
$t_{SU-CMD-RRR}$	3	CMD Valid before $\overline{RAE}$, $\overline{REM-RD}$ Falling		0		ns
$t_{H-CMD-RRR}$	4	CMD Invalid after $\overline{RAE}$, $\overline{REM-RD}$ Falling	T+	26		ns
$t_{PD-RRR-X}$	5	$\overline{RAE}$, $\overline{REM-RD}$ Falling to XACK Falling			26	ns
$t_{PD-X-LCL}$	6	XACK Falling to $\overline{LCL}$ Rising	$(n_{LW}+1)T+$	−5		ns
$t_{PD-LCL-X}$	7	$\overline{LCL}$ Rising to XACK Rising	2T+	−10	8	ns
$t_{PD-RRR-LCL}$	8	$\overline{RAE}$, $\overline{REM-RD}$ Rising to $\overline{LCL}$ Falling		3		ns
$t_{AZ-A-LCL}$	9	A Disabled before $\overline{LCL}$ Rising	T_L+	−18		ns
$t_{ZA-LCL-A}$	10	A Enabled after $\overline{LCL}$ Falling	T_H+	15		ns
$t_{PD-LCL-PC}$	11	$\overline{LCL}$ Rising to AD (PC) Valid	T+		22	ns
$t_{PD-PC-X}$	12	AD (PC, RIC) Valid before XACK Rising	T+	−24		ns
$t_{PD-PC-RRR}$	13	$\overline{RAE}$, $\overline{REM-RD}$ Rising to AD (PC) Invalid		6		ns
t_{W-PC}	14	AD (PC, RIC) Valid Time	T+	−2		ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

Note 2: The maximum value for this parameter is the latest $\overline{RAE}$, $\overline{REM-RD}$ can be removed without adding a T-state to the remote access.



TL/F/9336–65

FIGURE 5-11. Buffered Read of PC, RIC

5.0 Device Specifications (Continued)

TABLE 5-12. Buffered Read of DMEM (Note 1)

Symbol	ID#	Parameter	Formula	Min	Max	Units
$t_{SU-RRR-CO}$	1	$\overline{RAE}$, $\overline{REM-RD}$ Falling before CLK-OUT Rising		22		ns
$t_{H-RRR-X}$	2	$\overline{RAE}$, $\overline{REM-RD}$ Rising after XACK Rising (Note 2)		0		ns
			T+		−32	ns
$t_{SU-CMD-RRR}$	3	CMD Valid before $\overline{RAE}$, $\overline{REM-RD}$ Falling		0		ns
$t_{H-CMD-RRR}$	4	CMD Invalid after $\overline{RAE}$, $\overline{REM-RD}$ Falling	T+	26		ns
$t_{PD-RRR-X}$	5	$\overline{RAE}$, $\overline{REM-RD}$ Falling to XACK Falling			26	ns
$t_{PD-X-LCL}$	6	XACK Falling to $\overline{LCL}$ Rising	$(n_{LW}+1)T+$	−5		ns
$t_{PD-LCL-X}$	7	$\overline{LCL}$ Rising to XACK Rising	$(n_{DW}+2)T+$	−10	8	ns
$t_{PD-RRR-LCL}$	8	$\overline{RAE}$, $\overline{REM-RD}$ Rising to $\overline{LCL}$ Falling	T+	3		ns
$t_{PD-LCL-RD}$	9	$\overline{LCL}$ Rising to $\overline{READ}$ Falling	T+	−5	16	ns
$t_{PD-RD-X}$	10	$\overline{READ}$ Falling to XACK Rising	$(n_{DW}+1)T+$	−15		ns
$t_{PD-RRR-RD}$	11	$\overline{RAE}$, $\overline{REM-RD}$ Rising to $\overline{READ}$ Rising		1	28	ns
$t_{AZ-AAD-LCL}$	12	A, AD Disabled before $\overline{LCL}$ Rising	T_L+	−20		ns
$t_{ZA-LCL-AAD}$	13	A, AD Enabled after $\overline{LCL}$ Falling	T_H+	−10		ns
t_{W-RD}	14	Read Low	$(n_{DW}+1)T+$	−4		ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

Note 2: The maximum value for this parameter is the latest $\overline{RAE}$, $\overline{REM-RD}$ can be removed without adding a T-state to the remote access.

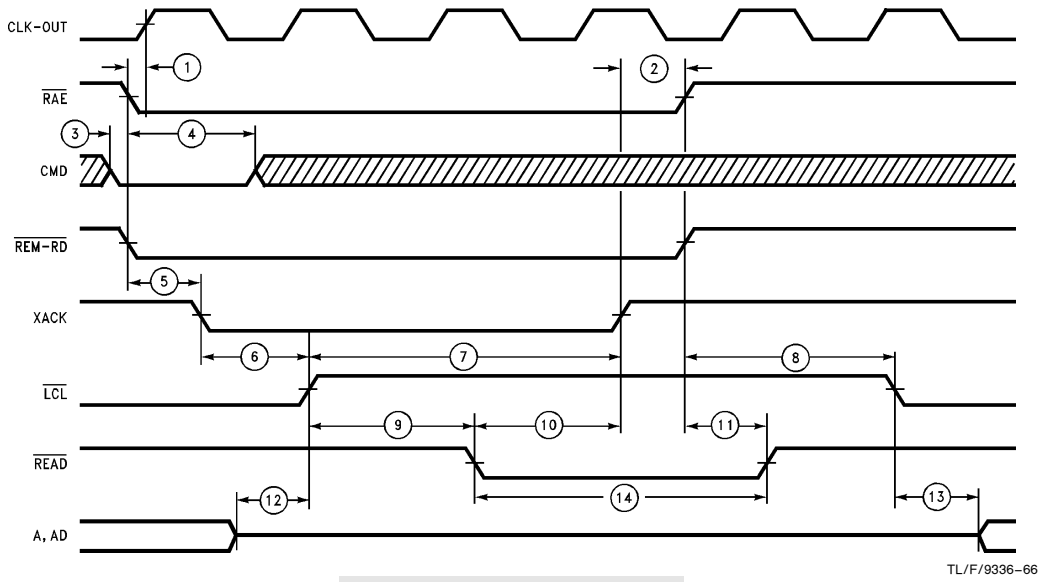


FIGURE 5-12. Buffered Read of DMEM

5.0 Device Specifications (Continued)

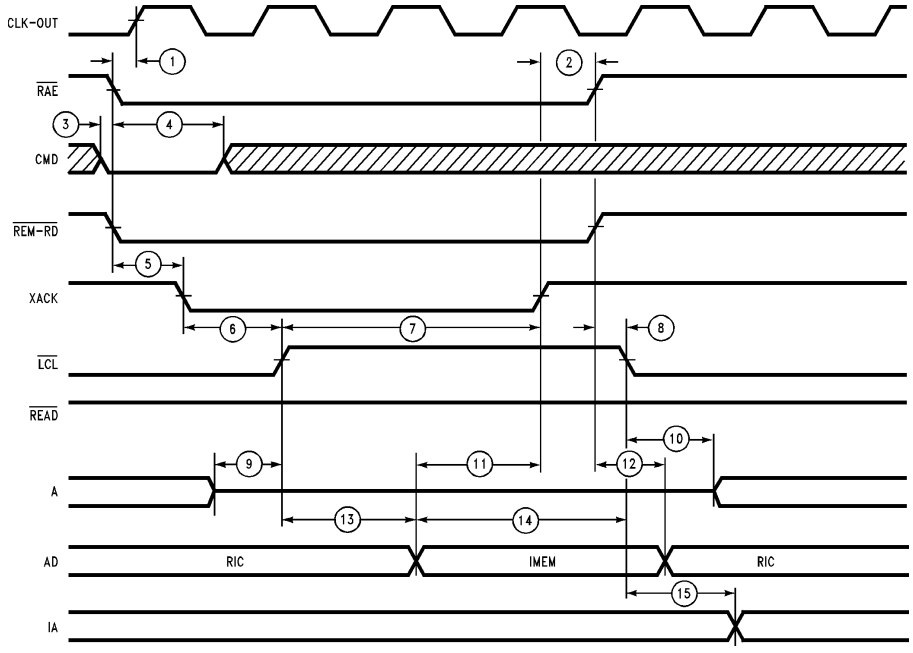
TABLE 5-13. Buffered Read of IMEM (Note 1)

Symbol	ID#	Parameter	Formula	Min	Max	Units
$t_{SU-RRR-CO}$	1	$\overline{RAE}$, $\overline{REM-RD}$ Falling before CLK-OUT Rising		22		ns
$t_{H-RRR-X}$	2	$\overline{RAE}$, $\overline{REM-RD}$ Rising after XACK Rising (Note 2)		0		ns
			T+		−32	ns
$t_{SU-CMD-RRR}$	3	CMD Valid before $\overline{RAE}$, $\overline{REM-RD}$ Falling		0		ns
$t_{H-CMD-RRR}$	4	CMD Invalid after $\overline{RAE}$, $\overline{REM-RD}$ Falling	T+	26		ns
$t_{PD-RRR-X}$	5	$\overline{RAE}$, $\overline{REM-RD}$ Falling to XACK Falling			26	ns
$t_{PD-X-LCL}$	6	XACK Falling to $\overline{LCL}$ Rising	T+	−5		ns
$t_{PD-LCL-X}$	7	$\overline{LCL}$ Rising to XACK Rising	$(n_{IW}+2)T+$	−10	8	ns
$t_{PD-RRR-LCL}$	8	$\overline{RAE}$, $\overline{REM-RD}$ Rising to $\overline{LCL}$ Falling		3		ns
$t_{AZ-LCL-A}$	9	A Disabled after $\overline{LCL}$ Rising	T_L+	−18		ns
$t_{ZA-A-LCL}$	10	A Enabled before $\overline{LCL}$ Falling	T_H+	15		ns
$t_{PD-IMEM-X}$	11	AD (IMEM) Valid before XACK Rising	$(n_{IW}+1)T+$	−25		ns
$t_{PD-RRR-IMEM}$	12	AD (IMEM) Invalid after $\overline{RAE}$, $\overline{REM-RD}$ Rising		10		ns
$t_{PD-LCL-IMEM}$	13	$\overline{LCL}$ Rising to AD (IMEM) Valid	T+		22	ns
t_{W-IMEM}	14	(IMEM) Valid	$(n_{IW}+1)T+$	0		ns
$t_{PD-LCL-IA}$	15	$\overline{LCL}$ Falling to Next IA Valid (Note 3)	T_H+	8		ns
			$T+T_H+$		44	ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

Note 2: The maximum value for this parameter is the latest $\overline{RAE}$, $\overline{REM-RD}$ can be removed without adding a T-state to the remote access.

Note 3: Two remote reads from instruction memory are necessary to read a 16-bit instruction word from IMEM—low byte followed by high byte. The timing for the two reads are the same except that IA is incremented after the high instruction memory byte is read.



TL/F/9336–67

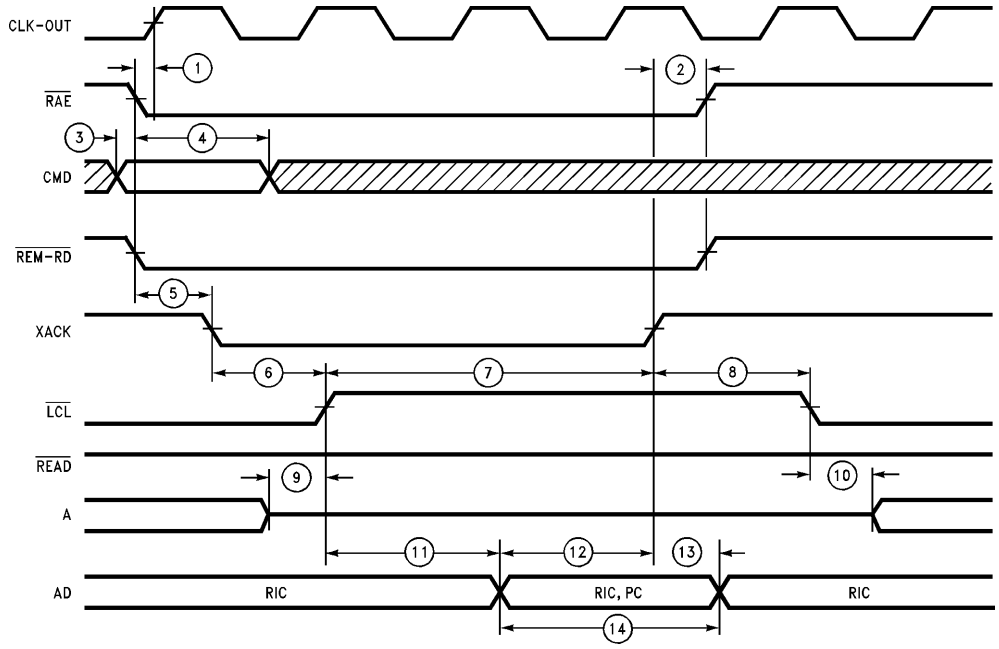
FIGURE 5-13. Buffered Read of IMEM

5.0 Device Specifications (Continued)

TABLE 5-14. Latched Read of PC, RIC (Note 1)

Symbol	ID #	Parameter	Formula	Min	Max	Units
$t_{SU\text{-}RRR\text{-}CO}$	1	$\overline{RAE}$, $\overline{REM\text{-}RD}$ Falling before CLK-OUT Rising		22		ns
$t_{H\text{-}RRR\text{-}X}$	2	$\overline{RAE}$, $\overline{REM\text{-}RD}$ Rising after XACK Rising		0		ns
$t_{SU\text{-}CMD\text{-}RRR}$	3	CMD Valid before $\overline{RAE}$, $\overline{REM\text{-}RD}$ Falling		0		ns
$t_{H\text{-}CMD\text{-}RRR}$	4	CMD Invalid after $\overline{RAE}$, $\overline{REM\text{-}RD}$ Falling	T+	26		ns
$t_{PD\text{-}RRR\text{-}X}$	5	$\overline{RAE}$, $\overline{REM\text{-}RD}$ Falling to XACK Falling			26	ns
$t_{PD\text{-}Xf\text{-}LCLr}$	6	XACK Falling to $\overline{LCL}$ Rising	$(n_{LW}+1)T+$	−5		ns
$t_{PD\text{-}LCL\text{-}X}$	7	$\overline{LCL}$ Rising to XACK Rising	2T+	−10	8	ns
$t_{PD\text{-}Xr\text{-}LCLf}$	8	XACK Rising to $\overline{LCL}$ Falling	T+	−11	11	ns
$t_{AZ\text{-}A\text{-}LCL}$	9	A Disabled before $\overline{LCL}$ Rising	T_L+	−18		ns
$t_{ZA\text{-}LCL\text{-}A}$	10	A Enabled after $\overline{LCL}$ Falling	T_H+	−12		ns
$t_{PC\text{-}LCL\text{-}PC}$	11	$\overline{LCL}$ Rising to AD (PC) Valid	T+		20	ns
$t_{PD\text{-}PC\text{-}X}$	12	AD (PC) Valid before XACK Rising	T+	−22		ns
$t_{PD\text{-}X\text{-}PC}$	13	XACK Rising to AD (PC) Invalid	T_H+	0		ns
$t_{W\text{-}PC}$	14	AD (PC, RIC) Valid	$T+T_H+$	−12		ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.



TL/F/9336–68

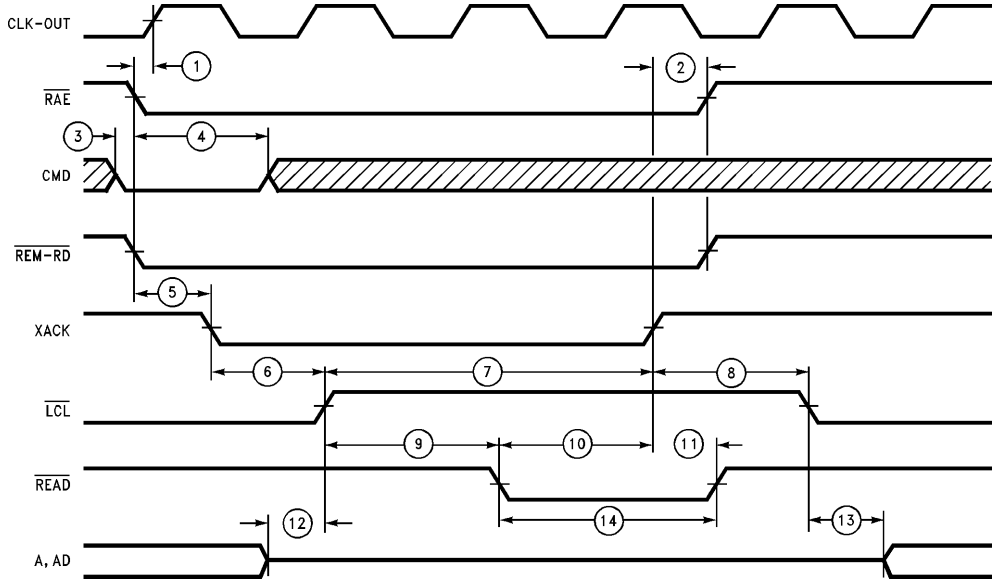
FIGURE 5-14. Latched Read of PC, RIC

5.0 Device Specifications (Continued)

TABLE 5-15. Latched Read of DMEM (Note 1)

Symbol	ID #	Parameter	Formula	Min	Max	Units
$t_{SU-RRR-CO}$	1	$\overline{RAE}$, $\overline{REM-RD}$ Falling before CLK-OUT Rising		22		ns
$t_{H-RRR-X}$	2	$\overline{RAE}$, $\overline{REM-RD}$ Rising after XACK Rising		0		ns
$t_{SU-CMD-RRR}$	3	CMD Valid before $\overline{RAE}$, $\overline{REM-RD}$ Falling		0		ns
$t_{H-CMD-RRR}$	4	CMD Invalid after $\overline{RAE}$, $\overline{REM-RD}$ Falling	T+	26		ns
$t_{PD-RRR-X}$	5	$\overline{RAE}$, $\overline{REM-RD}$ Falling to XACK Falling			26	ns
$t_{PD-Xf-LCLr}$	6	XACK Falling to $\overline{LCL}$ Rising	$(n_{LW}+1)T+$	−5		ns
$t_{PD-LCL-X}$	7	$\overline{LCL}$ Rising to XACK Rising	$(n_{DW}+2)T+$	−10	8	ns
$t_{PD-Xr-LCLf}$	8	XACK Rising to $\overline{LCL}$ Falling	T+	−11	11	ns
$t_{PC-LCL-RD}$	9	$\overline{LCL}$ Rising to $\overline{READ}$ Falling	T+	−5	16	ns
$t_{PD-RD-X}$	10	$\overline{READ}$ Falling before XACK Rising	$(n_{DW}+1)T+$	−15		ns
$t_{PD-X-RD}$	11	XACK Rising to $\overline{READ}$ Rising	T_H+	−7	12	ns
$t_{AZ-AAD-LCL}$	12	A, AD Disabled before $\overline{LCL}$ Rising	T_L+	−20		ns
$t_{ZA-LCL-AAD}$	13	A, AD Enabled after $\overline{LCL}$ Falling	T_H+	−10		ns
t_{W-RD}	14	$\overline{READ}$ Low	$(n_{DW}+1)T+T_H+$	−12		ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.



TL/F/9336–69

FIGURE 5-15. Latched Read of DMEM

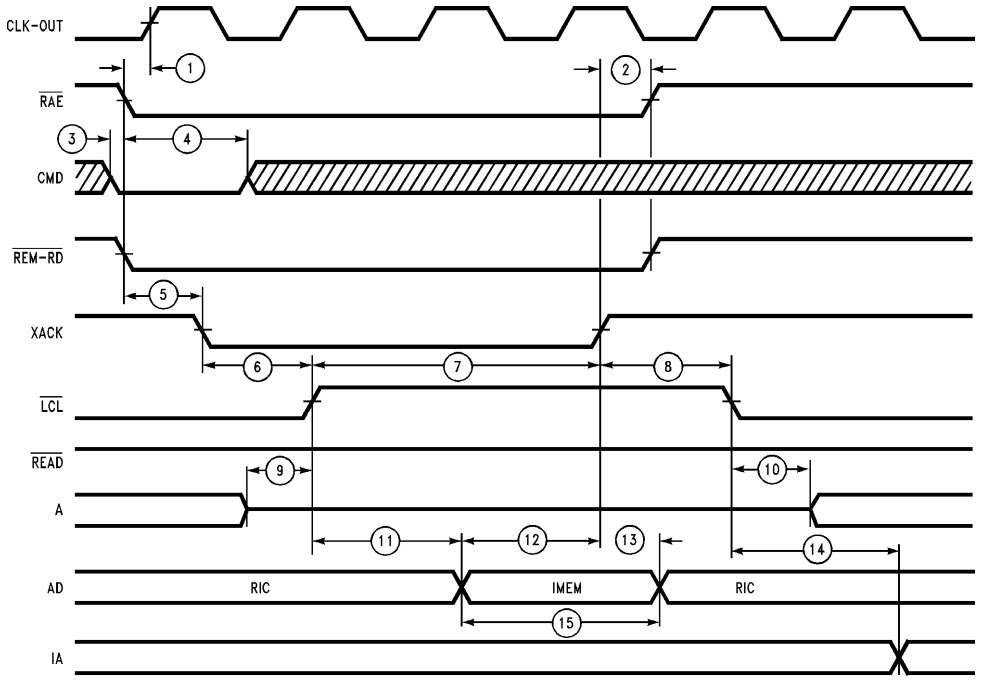
5.0 Device Specifications (Continued)

TABLE 5-16. Latched Read of IMEM (Note 1)

Symbol	ID #	Parameter	Formula	Min	Max	Units
$t_{SU\text{-}RRR\text{-}CO}$	1	$\overline{RAE}$, $\overline{REM\text{-}RD}$ Falling before CLK-OUT Rising		22		ns
$t_{H\text{-}RRR\text{-}X}$	2	$\overline{RAE}$, $\overline{REM\text{-}RD}$ Rising after XACK Rising		0		ns
$t_{SU\text{-}CMD\text{-}RRR}$	3	CMD Valid before $\overline{RAE}$, $\overline{REM\text{-}RD}$ Falling		0		ns
$t_{H\text{-}CMD\text{-}RRR}$	4	CMD Invalid after $\overline{RAE}$, $\overline{REM\text{-}RD}$ Falling	T+	26		ns
$t_{PD\text{-}RRR\text{-}X}$	5	$\overline{RAE}$, $\overline{REM\text{-}RD}$ Falling to XACK Falling			26	ns
$t_{PD\text{-}Xf\text{-}LCLr}$	6	XACK Falling to $\overline{LCL}$ Rising	T+	−5		ns
$t_{PD\text{-}LCL\text{-}X}$	7	$\overline{LCL}$ Rising to XACK Rising	$(n_{IW}+2)T+$	−10	8	ns
$t_{PD\text{-}Xr\text{-}LCLf}$	8	XACK Rising to $\overline{LCL}$ Falling	T+	−11	11	ns
$t_{AZ\text{-}A\text{-}LCL}$	9	A Disabled before $\overline{LCL}$ Rising	T_L+	−18		ns
$t_{ZA\text{-}LCL\text{-}A}$	10	A Enabled after $\overline{LCL}$ Falling	T_H+	−12		ns
$t_{PD\text{-}LCL\text{-}IMEM}$	11	$\overline{LCL}$ Rising to AD (IMEM) Valid	T+		22	ns
$t_{PD\text{-}IMEM\text{-}X}$	12	AD (IMEM) Valid to XACK Rising	$(n_{IW}+1)T+$	−23		ns
$t_{PD\text{-}X\text{-}IMEM}$	13	XACK Rising to AD (IMEM) Invalid	T_H+	1		ns
$t_{PD\text{-}LCL\text{-}IA}$	14	$\overline{LCL}$ Falling to Next IA Valid (Note 2)	$T+T_H+$	−19	5	ns
$t_{W\text{-}IMEM}$	15	IMEM Valid	$(n_{IW}+1)T+T_H+$	−9		ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

Note 2: Two remote reads from instruction memory are necessary to read a 16-bit instruction word from IMEM—low byte followed by high byte. The timing for the two reads are the same except that IA is incremented after the high instruction memory byte is read.



TL/F/9336–70

FIGURE 5-16. Latched Read of IMEM

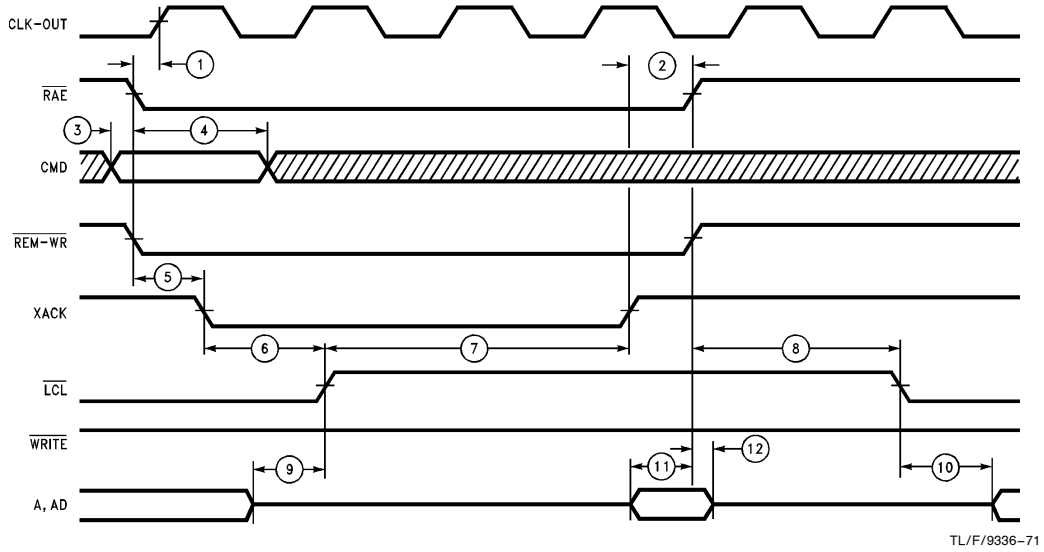
5.0 Device Specifications (Continued)

TABLE 5-17. Slow Buffered Write of PC, RIC (Note 1)

Symbol	ID#	Parameter	Formula	Min	Max	Units
$t_{SU-RRW-CO}$	1	$\overline{RAE}$, $\overline{REM-WR}$ Falling before CLK-OUT Rising		24		ns
$t_{H-RRW-X}$	2	$\overline{RAE}$, $\overline{REM-WR}$ Rising after XACK Rising (Note 2)		0		ns
			T+		−37	ns
$t_{SU-CMD-RRW}$	3	CMD Valid before $\overline{RAE}$, $\overline{REM-WR}$ Falling		0		ns
$t_{H-CMD-RRW}$	4	CMD Invalid after $\overline{RAE}$, $\overline{REM-WR}$ Falling	T+	26		ns
$t_{PD-RRW-X}$	5	$\overline{RAE}$, $\overline{REM-WR}$ Falling to XACK Falling			26	ns
$t_{PD-X-LCL}$	6	XACK Falling to $\overline{LCL}$ Rising	$(n_{LW}+1)T+$	−5		ns
$t_{PD-LCL-X}$	7	$\overline{LCL}$ Rising to XACK Rising	2T+	−10	8	ns
$t_{PD-RRW-LCL}$	8	$\overline{RAE}$, $\overline{REM-WR}$ Rising to $\overline{LCL}$ Falling	T+	5		ns
$t_{AZ-AAD-LCL}$	9	A, AD Disabled before $\overline{LCL}$ Rising	T_L+	−20		ns
$t_{ZA-LCL-AAD}$	10	A, AD Enabled after $\overline{LCL}$ Falling	T_H+	−10		ns
$t_{SU-RDAT-RRW}$	11	AD (Data) Valid before $\overline{RAE}$, $\overline{REM-WR}$ Rising		12		ns
$t_{H-RDAT-RRW}$	12	AD (Data) Invalid after $\overline{RAE}$, $\overline{REM-WR}$ Rising		10		ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

Note 2: The maximum value for this parameter is the latest $\overline{RAE}$, $\overline{REM-WR}$ can be removed without adding a T-state to the remote access.



TL/F/9336–71

FIGURE 5-17. Slow Buffered Write of PC, RIC

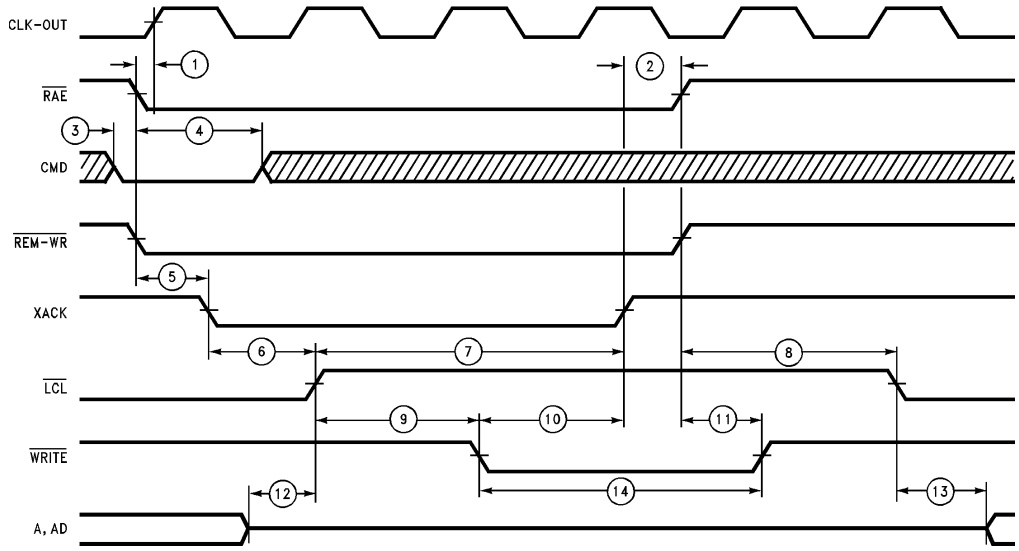
5.0 Device Specifications (Continued)

TABLE 5-18. Slow Buffered Write of DMEM (Note 1)

Symbol	ID #	Parameter	Formula	Min	Max	Units
$t_{SU\text{-}RRW\text{-}CO}$	1	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling before CLK-OUT Rising		24		ns
$t_{H\text{-}RRW\text{-}X}$	2	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Rising after XACK Rising (Note 2)		0		ns
			T+		−34	ns
$t_{SU\text{-}CMD\text{-}RRW}$	3	CMD Valid before $\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling		0		ns
$t_{H\text{-}CMD\text{-}RRW}$	4	CMD Invalid after $\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling	T+	26		ns
$t_{PD\text{-}RRW\text{-}X}$	5	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling to XACK Falling			26	ns
$t_{PD\text{-}X\text{-}LCL}$	6	XACK Falling to $\overline{LCL}$ Rising	$(n_{LW}+1)T+$	−5		ns
$t_{PD\text{-}LCL\text{-}X}$	7	$\overline{LCL}$ Rising to XACK Rising	$(n_{DW}+2)T+$	−10	8	ns
$t_{PD\text{-}RRW\text{-}LCL}$	8	$\overline{RAE}$, $\overline{REM\text{-}WR}$ to $\overline{LCL}$ Falling	T+	5		ns
$t_{PD\text{-}LCL\text{-}WR}$	9	$\overline{LCL}$ Rising to $\overline{WRITE}$ Falling	T+	−5		ns
$t_{PD\text{-}WR\text{-}X}$	10	$\overline{WRITE}$ Falling to XACK Rising	$(n_{DW}+1)T+$	−17		ns
$t_{PD\text{-}RRW\text{-}WR}$	11	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Rising to $\overline{WRITE}$ Rising		2	28	ns
$t_{AZ\text{-}AAD\text{-}LCL}$	12	A, AD Disabled before $\overline{LCL}$ Rising	T_L+	−20		ns
$t_{AZ\text{-}LCL\text{-}AAD}$	13	A, AD Enabled after $\overline{LCL}$ Falling	T_H+	−10		ns
$t_{W\text{-}WR}$	14	$\overline{WRITE}$ Low	$(n_{DW}+1)T+$	−3		ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

Note 2: The maximum value for this parameter is the latest $\overline{RAE}$, $\overline{REM\text{-}WR}$ can be removed without adding a T-state to the remote access.



TL/F/9336–72

FIGURE 5-18. Slow Buffered Write of DMEM

5.0 Device Specifications (Continued)

TABLE 5-19. Slow Buffered Write of IMEM (Notes 1, 2)

Symbol	ID#	Parameter	Formula	Min	Max	Units
$t_{SU\text{-}RRW\text{-}CO}$	1	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling before CLK-OUT Rising		24		ns
$t_{H\text{-}RRW\text{-}X}$	2	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Rising after XACK Rising (Note 3)		0		ns
			T+		−34	ns
$t_{SU\text{-}CMD\text{-}RRW}$	3	CMD Valid before $\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling		0		ns
$t_{H\text{-}CMD\text{-}RRW}$	4	CMD Invalid after $\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling	T+	26		ns
$t_{PD\text{-}RRW\text{-}X}$	5	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling to XACK Falling			26	ns
$t_{PD\text{-}X\text{-}LCL}$	6	XACK Falling to $\overline{LCL}$ Rising	T+	−5		ns
$t_{PD\text{-}LCL\text{-}X}$	7	$\overline{LCL}$ Rising to XACK Rising	$(n_{IW}+2)T+$	−10	8	ns
$t_{PD\text{-}RRW\text{-}LCL}$	8	$\overline{RAE}$, $\overline{REM\text{-}WR}$ to $\overline{LCL}$ Falling	T+	5		ns
$t_{AZ\text{-}AAD\text{-}LCL}$	9	A, AD Disabled before $\overline{LCL}$ Rising	T_L+	−20		ns
$t_{ZA\text{-}LCL\text{-}AAD}$	10	A, AD Enabled after $\overline{LCL}$ Falling	T_H+	−10		ns
$t_{PD\text{-}RDAT\text{-}I}$	11	AD (Data) Valid to I Valid			30	ns
$t_{H\text{-}RDAT\text{-}RRW}$	12	AD (Data) Invalid after $\overline{RAE}$, $\overline{REM\text{-}WR}$ Rising		14		ns
$t_{PD\text{-}LCL\text{-}IA}$	13	$\overline{LCL}$ Falling to next IA Valid	$T+T_H+$	−20	3	ns
$t_{PD\text{-}LCL\text{-}IWR}$	14	$\overline{LCL}$ Rising to $\overline{IWR}$ Falling		−3		ns
$t_{PD\text{-}IWR\text{-}X}$	15	$\overline{IWR}$ Falling before XACK Rising	$(n_{IW}+2)T+$	−19		ns
$t_{PD\text{-}RRW\text{-}IWR}$	16	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Rising to $\overline{IWR}$ Rising		5		ns
$t_{ZA\text{-}IWR\text{-}I}$	17	$\overline{IWR}$ Falling to I Enabled	T+	−2		ns
$t_{AZ\text{-}IWR\text{-}I}$	18	$\overline{IWR}$ Rising to I Disabled		22	52	ns
$t_{PD\text{-}I\text{-}IWR}$	19	I Valid before $\overline{IWR}$ Rising	$(n_{IW}+1)T+$	−10		ns
$t_{W\text{-}IWR}$	20	$\overline{IWR}$ Low	$(n_{IW}+2)T+$	−2		ns
$t_{PD\text{-}I\text{-}IA}$	21	I Disabled to IA Invalid	$2T+T_H+$	−64		ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

Note 2: Two remote writes to instruction memory are necessary to store a 16-bit instruction word to IMEM—low byte followed by high byte. The timing for the 2nd write is shown in the following diagram. The timing of the first write is the same as a write of the PC or RIC.

Note 3: The maximum value for this parameter is the latest $\overline{RAE}$, $\overline{REM\text{-}WR}$ can be removed without adding a T-state to the remote access.

5.0 Device Specifications (Continued)

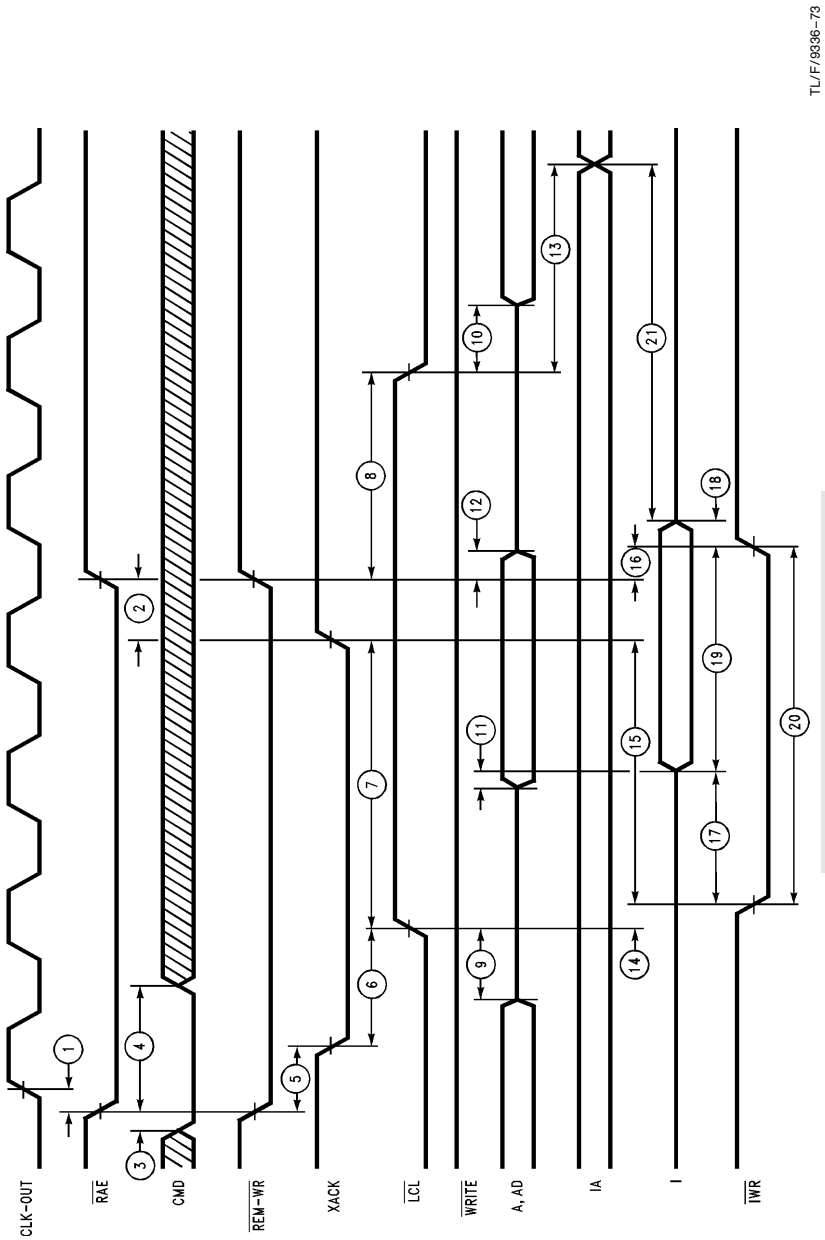


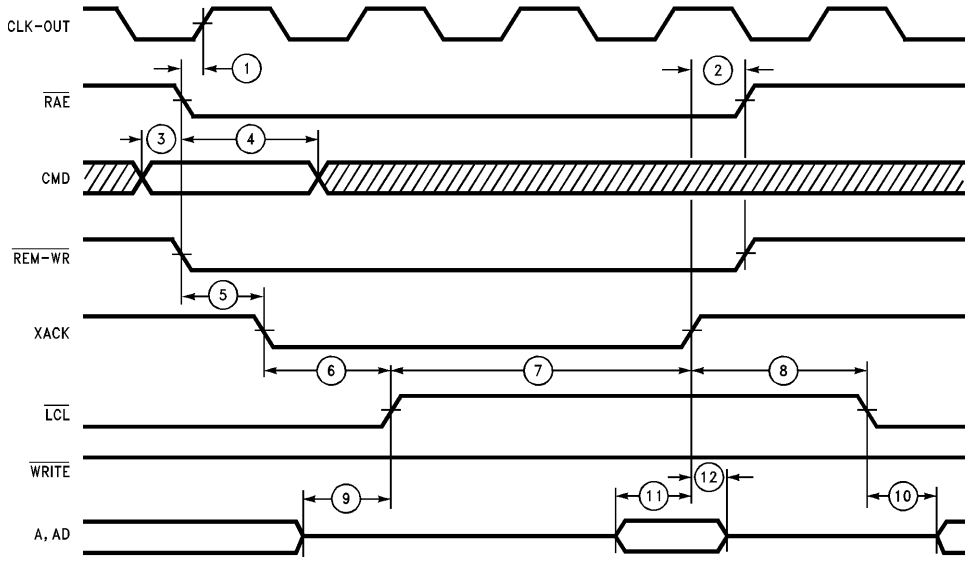
FIGURE 5-19. Slow Buffered Write of IMEM

5.0 Device Specifications (Continued)

TABLE 5-20. Fast Buffered Write of RIC, PC (Note 1)

Symbol	ID#	Parameter	Formula	Min	Max	Units
$t_{SU\text{-}RRW\text{-}CO}$	1	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling before CLK-OUT Rising		24		ns
$t_{H\text{-}RRW\text{-}X}$	2	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Rising after XACK Rising		0		ns
$t_{SU\text{-}CMD\text{-}RRW}$	3	CMD Valid before $\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling		0		ns
$t_{H\text{-}CMD\text{-}RRW}$	4	CMD Invalid after $\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling	T+	26		ns
$t_{PD\text{-}RRW\text{-}X}$	5	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling to XACK Falling			26	ns
$t_{PD\text{-}X\text{-}LCL}$	6	XACK Falling to $\overline{LCL}$ Rising	$(n_{LW}+1)T+$	−5		ns
$t_{PD\text{-}LCL\text{-}X}$	7	$\overline{LCL}$ Rising to XACK Rising	2T+	−10	8	ns
$t_{PD\text{-}Xr\text{-}LCLf}$	8	XACK Rising to $\overline{LCL}$ Falling	T+	−11	11	ns
$t_{AZ\text{-}AAD\text{-}LCL}$	9	A, AD Disabled before $\overline{LCL}$ Rising	T_L+	−20		ns
$t_{ZA\text{-}LCL\text{-}AAD}$	10	A, AD Enabled after $\overline{LCL}$ Falling	T_H+	−10		ns
$t_{SU\text{-}RDAT\text{-}X}$	11	AD (Data) Valid before XACK Rising		26		ns
$t_{H\text{-}RDAT\text{-}X}$	12	AD (Data) Invalid after XACK Rising		3		ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.



TL/F/9336–74

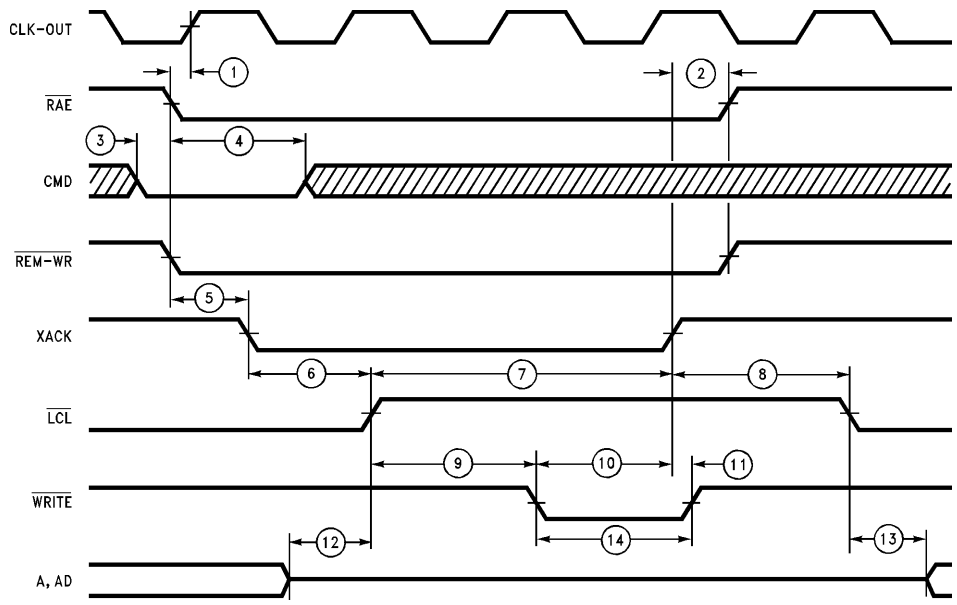
FIGURE 5-20. Fast Buffered Write of RIC, PC

5.0 Device Specifications (Continued)

TABLE 5-21. Fast Buffered Write of DMEM (Note 1)

Symbol	ID #	Parameter	Formula	Min	Max	Units
$t_{SU\text{-}RRW\text{-}CO}$	1	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling before CLK-OUT Rising		24		ns
$t_{H\text{-}RRW\text{-}X}$	2	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Rising after XACK Rising		0		ns
$t_{SU\text{-}CMD\text{-}RRW}$	3	CMD Valid before $\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling		0		ns
$t_{H\text{-}CMD\text{-}RRW}$	4	CMD Invalid after $\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling	T+	26		ns
$t_{PD\text{-}RRW\text{-}X}$	5	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling to XACK Falling			26	ns
$t_{PD\text{-}Xf\text{-}LCLr}$	6	XACK Falling to $\overline{LCL}$ Rising	$(n_{LW}+1)T+$	−5		ns
$t_{PD\text{-}LCL\text{-}X}$	7	$\overline{LCL}$ Rising to XACK Rising	$(n_{DW}+2)T+$	−10	8	ns
$t_{PD\text{-}Xr\text{-}LCLf}$	8	XACK Rising to $\overline{LCL}$ Falling	T+	−11	11	ns
$t_{PD\text{-}LCL\text{-}WR}$	9	$\overline{LCL}$ Rising to $\overline{WRITE}$ Falling	T+	−5		ns
$t_{PD\text{-}WR\text{-}X}$	10	$\overline{WRITE}$ Falling to XACK Rising	$(n_{DW}+1)T+$	−16		ns
$t_{PD\text{-}X\text{-}WR}$	11	XACK Rising to $\overline{WRITE}$ Rising		−4	13	ns
$t_{AZ\text{-}AAD\text{-}LCL}$	12	A, AD Disabled before $\overline{LCL}$ Rising	T_L+	−20		ns
$t_{ZA\text{-}LCL\text{-}AAD}$	13	A, AD Enabled after $\overline{LCL}$ Falling	T_H+	−10		ns
$t_{W\text{-}WR}$	14	$\overline{WRITE}$ Low	$(n_{DW}+1)T+$	−10		ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.



TL/F/9336–75

FIGURE 5-21. Fast Buffered Write of DMEM

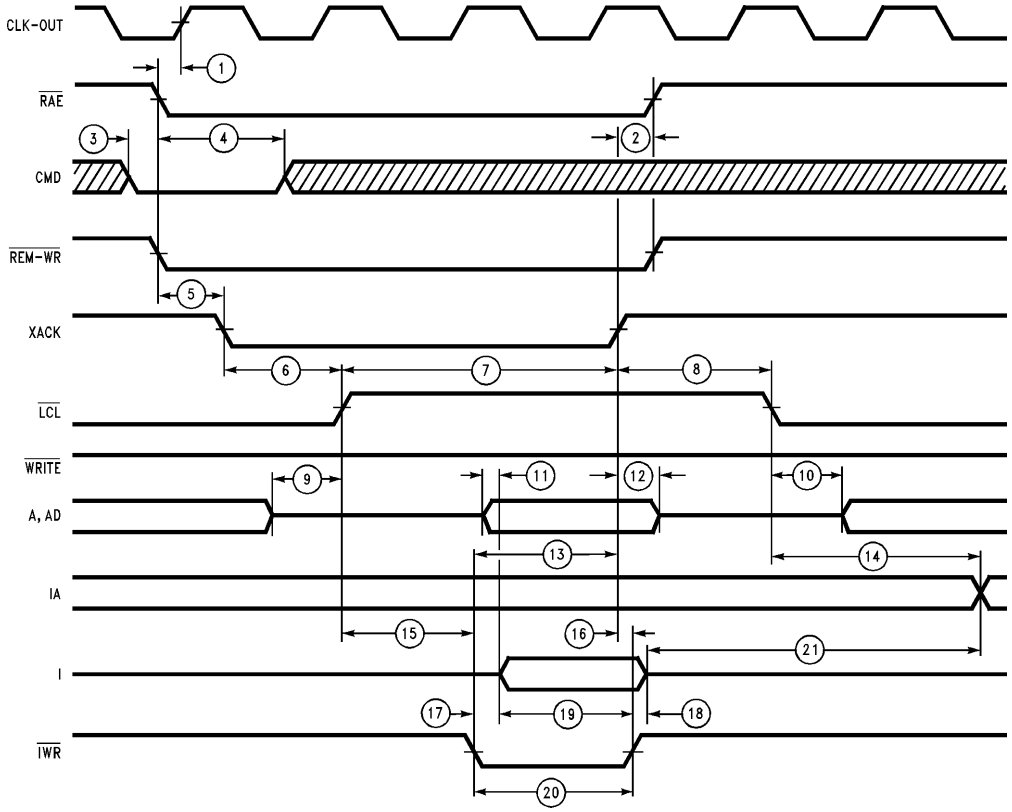
TABLE 5-22. Fast Buffered Write of IMEM (Notes 1, 2)

Symbol	ID #	Parameter	Formula	Min	Max	Units
$t_{SU\text{-}RRW\text{-}CO}$	1	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling before CLK-OUT Rising		24		ns
$t_{H\text{-}RRW\text{-}X}$	2	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Rising after XACK Rising		0		ns
$t_{SU\text{-}CMD\text{-}RRW}$	3	CMD Valid before $\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling		0		ns
$t_{H\text{-}CMD\text{-}RRW}$	4	CMD Invalid after $\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling	T+	26		ns
$t_{PD\text{-}RRW\text{-}X}$	5	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling to XACK Falling			26	ns
$t_{PD\text{-}Xf\text{-}LCLr}$	6	XACK Falling to $\overline{LCL}$ Rising	T+	−5		ns
$t_{PD\text{-}LCL\text{-}X}$	7	$\overline{LCL}$ Rising to XACK Rising	$(n_{IW}+2)T+$	−10	8	ns
$t_{PD\text{-}Xr\text{-}LCLf}$	8	XACK Rising to $\overline{LCL}$ Falling	T+	−11	11	ns
$t_{AZ\text{-}AAD\text{-}LCL}$	9	A, AD Disabled before $\overline{LCL}$ Rising	T_L+	−20		ns
$t_{ZA\text{-}LCL\text{-}AAD}$	10	A, AD Enabled after $\overline{LCL}$ Falling	T_H+	−10		ns
$t_{PD\text{-}RDAT\text{-}I}$	11	AD (Data) Valid to I Valid			30	ns
$t_{H\text{-}RDAT\text{-}X}$	12	AD (Data) Invalid after XACK Rising		3		ns
$t_{PD\text{-}IWR\text{-}X}$	13	$\overline{IWR}$ Falling before XACK Rising	$(n_{IW}+2)T+$	−19		ns
$t_{PD\text{-}LCL\text{-}IA}$	14	$\overline{LCL}$ Falling to next IA Valid	$T+T_H+$	−19	5	ns
$t_{PD\text{-}LCL\text{-}IWR}$	15	$\overline{LCL}$ Rising to $\overline{IWR}$ Falling		−3		ns
$t_{PD\text{-}X\text{-}IWR}$	16	XACK Rising to $\overline{IWR}$ Rising		−2		ns
$t_{ZA\text{-}IWR\text{-}I}$	17	$\overline{IWR}$ Falling to I Enabled	T+	−2		ns
$t_{AZ\text{-}IWR\text{-}I}$	18	$\overline{IWR}$ Rising to I Disabled		22	52	ns
$t_{PD\text{-}I\text{-}IWR}$	19	I Valid before $\overline{IWR}$ Rising	$(n_{IW}+1)T+$	−18		ns
$t_{W\text{-}IWR}$	20	$\overline{IWR}$ Low Time	$(n_{IW}+2)T+$	−10		ns
$t_{PD\text{-}I\text{-}IA}$	21	I Disabled to IA Invalid	$2T+T_H+$	−70		ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

Note 2: Two remote writes to instruction memory are necessary to store a 16-bit instruction word to IMEM—low byte followed by high byte. The timing of the 2nd write is shown in the following diagram. The timing of the first write is the same as a write of the PC or RIC as shown in *Figure 5-20*.

5.0 Device Specifications (Continued)



TL/F/9336–76

FIGURE 5-22. Fast Buffered Write of IMEM

5.0 Device Specifications (Continued)

TABLE 5-23. Latched Write of PC, RIC (Note 1)

Symbol	ID #	Parameter	Formula	Min	Max	Units
$t_{SU-RRW-CO}$	1	$\overline{RAE}$, $\overline{REM-WR}$ Falling before CLK-OUT Rising		24		ns
$t_{H-RRW-CO}$	2	$\overline{RAE}$, $\overline{REM-WR}$ Rising after CLK-OUT Rising (Note 2)	T_H+	6		ns
			T+		−20	ns
$t_{H-RRW-X}$	3	$\overline{RAE}$, $\overline{REM-WR}$ Rising after XACK Rising		0		ns
$t_{SU-CMD-RRW}$	4	CMD Valid before $\overline{RAE}$, $\overline{REM-WR}$ Falling		0		ns
$t_{H-CMD-RRW}$	5	CMD Invalid after $\overline{RAE}$, $\overline{REM-WR}$ Falling	T+	26		ns
$t_{PD-RRW-X}$	6	$\overline{RAE}$, $\overline{REM-WR}$ Falling to XACK Falling			26	ns
$t_{SU-RDAT-LCL}$	7	AD (Data) Valid after $\overline{LCL}$ Rising	2T+		−30	ns
$t_{H-RDAT-LCL}$	8	AD (Data) Invalid after $\overline{LCL}$ Rising	2T+	2		ns
$t_{AZ-AAD-LCL}$	9	A, AD Disabled before $\overline{LCL}$ Rising	T_L+	−20		ns
$t_{ZA-LCL-AAD}$	10	A, AD Enabled after $\overline{LCL}$ Falling	T_H+	−10		ns
$t_{PD-RRW-WPND}$	11	$\overline{RAE}$, $\overline{REM-WR}$ Rising to $\overline{WR-PEND}$ Falling		5		ns
			T+		34	ns
$t_{SU-CMD-WPND}$	12	CMD Valid before $\overline{WR-PEND}$ Rising		16		ns
$t_{H-CMD-WPND}$	13	CMD Invalid after $\overline{WR-PEND}$ Rising		4		ns
$t_{SU-RRWr-CO}$	14	$\overline{RAE}$, $\overline{REM-WR}$ Rising before CLK-OUT Rising		20		ns
$t_{PD-X-WPND}$	15	XACK Rising to $\overline{WR-PEND}$ Rising			13	ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

Note 2: The maximum value for this parameter is the latest $\overline{RAE}$, $\overline{REM-WR}$ can be removed without delaying the remote access by one T-state.

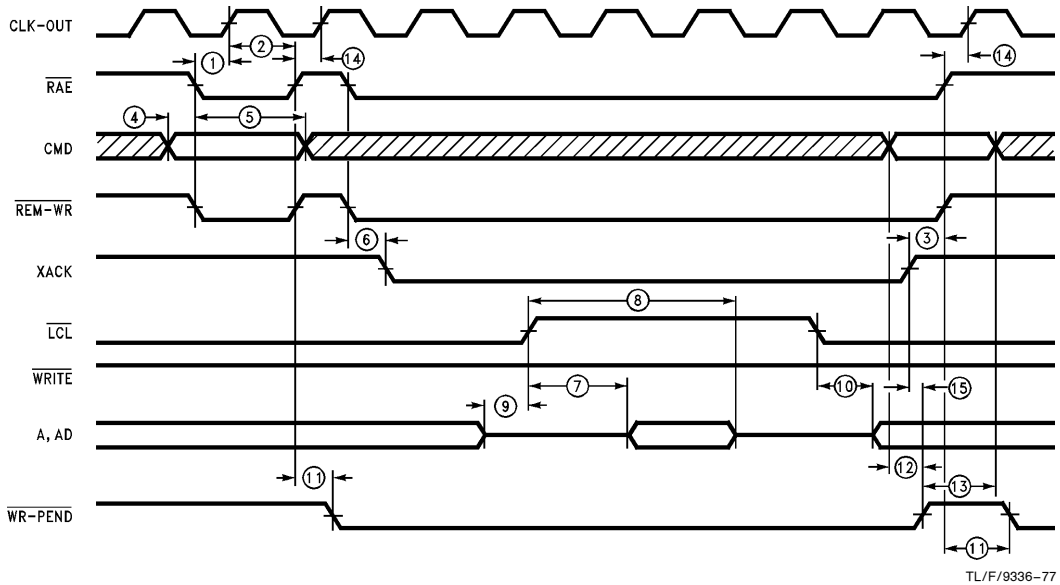


FIGURE 5-23. Latched Write of PC, RIC

5.0 Device Specifications (Continued)

TABLE 5-24. Latched Write of DMEM (Note 1)

Symbol	ID#	Parameter	Formula	Min	Max	Units
$t_{SU\text{-}RRW\text{-}CO}$	1	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling before CLK-OUT Rising		24		ns
$t_{H\text{-}RRW\text{-}CO}$	2	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Rising after CLK-OUT Rising (Note 2)	T_H+	6		ns
			$T+$		−20	ns
$t_{H\text{-}RRW\text{-}X}$	3	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Rising after XACK Rising		0		ns
$t_{SU\text{-}CMD\text{-}RRW}$	4	CMD Valid before $\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling		0		ns
$t_{H\text{-}CMD\text{-}RRW}$	5	CMD Invalid after $\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling	$T+$	26		ns
$t_{PD\text{-}RRW\text{-}X}$	6	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling to XACK Falling			26	ns
$t_{PD\text{-}LCL\text{-}WR}$	7	$\overline{LCL}$ Rising to $\overline{WRITE}$ Falling	$T+$	−5		ns
$t_{PD\text{-}WR\text{-}LCL}$	8	$\overline{WRITE}$ Rising to $\overline{LCL}$ Falling	$T+$	−11		ns
$t_{AZ\text{-}AAD\text{-}LCL}$	9	A, AD Disabled before $\overline{LCL}$ Rising	T_L+	−20		ns
$t_{ZA\text{-}LCL\text{-}AAD}$	10	A, AD Enabled after $\overline{LCL}$ Falling	T_H+	−10		ns
$t_{W\text{-}WR}$	11	$\overline{WRITE}$ Low Time	$(n_{DW}+1)T+$	−10		ns
$t_{PD\text{-}RRW\text{-}WPND}$	12	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Rising to $\overline{WR\text{-}PEND}$ Falling		5		ns
			$T+$		34	ns
$t_{SU\text{-}CMD\text{-}WPND}$	13	CMD Valid before $\overline{WR\text{-}PEND}$ Rising		16		ns
$t_{H\text{-}CMD\text{-}WPND}$	14	CMD Invalid after $\overline{WR\text{-}PEND}$ Rising		4		ns
$t_{SU\text{-}RRWr\text{-}CO}$	15	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Rising before CLK-OUT Rising		20		ns
$t_{PD\text{-}X\text{-}WPND}$	16	XACK Rising to $\overline{WR\text{-}PEND}$ Rising			13	ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

Note 2: The maximum value for this parameter is the latest $\overline{RAE}$, $\overline{REM\text{-}WR}$ can be removed without delaying the remote access by one T-state.

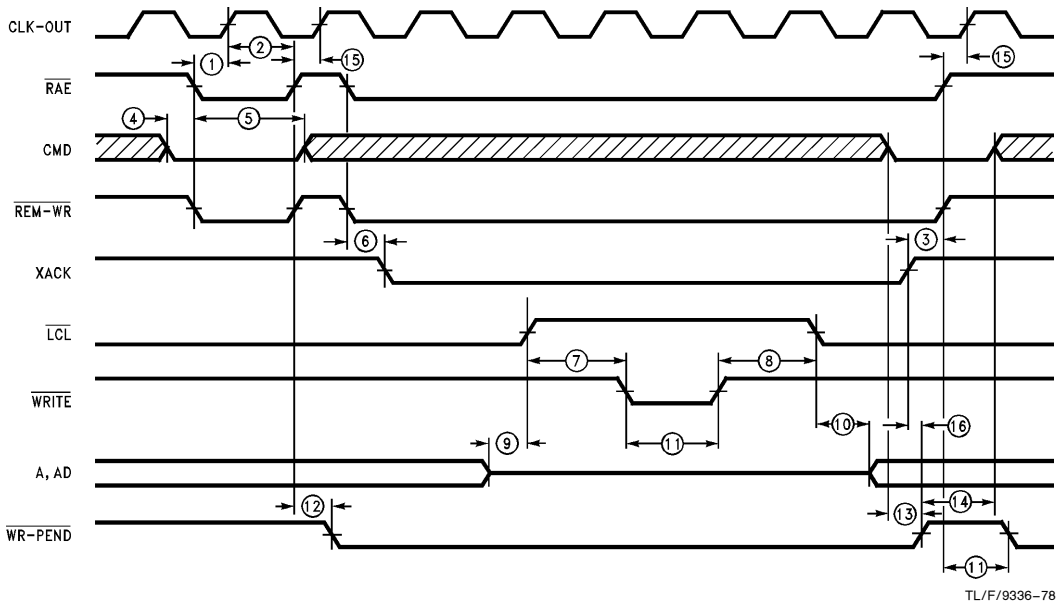


FIGURE 5-24. Latched Write of DMEM

TABLE 5-25. Latched Write of IMEM (Notes 1, 2)

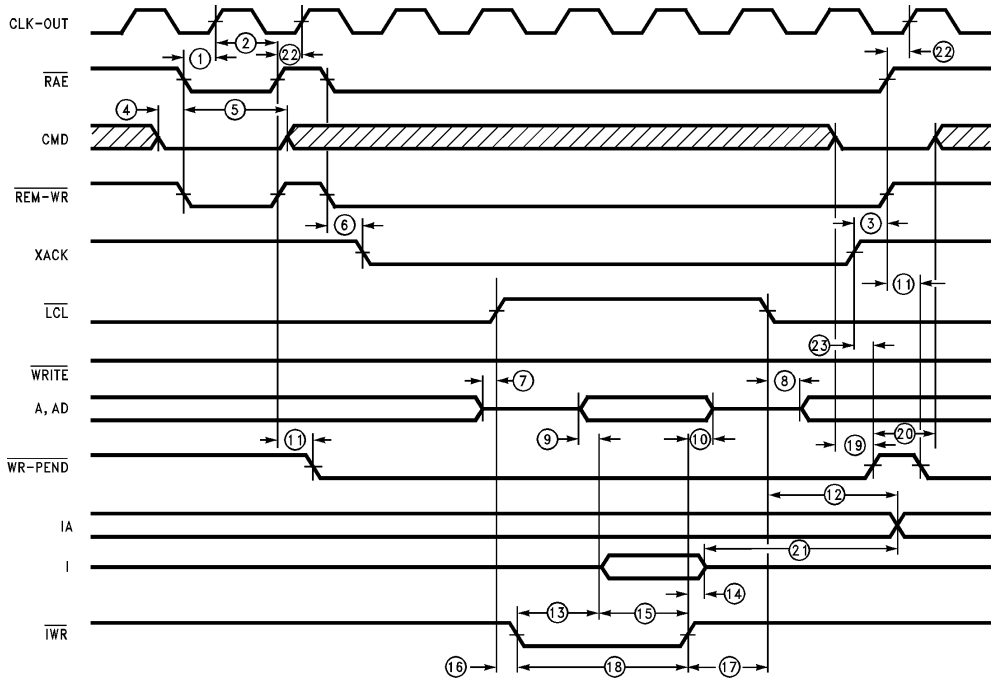
Symbol	ID #	Parameter	Formula	Min	Max	Units
$t_{SU\text{-}RRW\text{-}CO}$	1	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling before CLK-OUT Rising		24		ns
$t_{H\text{-}RRW\text{-}CO}$	2	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Rising after CLK-OUT Rising (Note 3)	T_H+	6		ns
			T+		−20	ns
$t_{H\text{-}RRW\text{-}X}$	3	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Rising after XACK Rising		0		ns
$t_{SU\text{-}CMD\text{-}RRW}$	4	CMD Valid before $\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling		0		ns
$t_{H\text{-}CMD\text{-}RRW}$	5	CMD Invalid after $\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling	T+	26		ns
$t_{PD\text{-}RRW\text{-}X}$	6	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Falling to XACK Falling			26	ns
$t_{AZ\text{-}AAD\text{-}LCL}$	7	A, AD Disabled before $\overline{LCL}$ Rising	T_L+	−20		ns
$t_{ZA\text{-}LCL\text{-}AAD}$	8	A, AD Enabled after $\overline{LCL}$ Falling	T_H+	−10		ns
$t_{PD\text{-}RDAT\text{-}I}$	9	AD (Data) Valid to I Valid			30	ns
$t_{H\text{-}RDAT\text{-}IWR}$	10	AD (Data) Invalid after $\overline{IWR}$ Rising		0		ns
$t_{PD\text{-}RRW\text{-}WPND}$	11	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Rising to $\overline{WR\text{-}PEND}$ Falling		5		
			T+		34	ns
$t_{PD\text{-}LCL\text{-}IA}$	12	$\overline{LCL}$ Falling to Next IA Valid	$T+T_H+$	−19	5	ns
$t_{ZA\text{-}IWR\text{-}I}$	13	$\overline{IWR}$ Falling to I Enabled	T+	−2		ns
$t_{AZ\text{-}IWR\text{-}I}$	14	$\overline{IWR}$ Rising to I Disabled		22	52	ns
$t_{PD\text{-}I\text{-}IWR}$	15	I Valid before $\overline{IWR}$ Rising	$(n_{IW}+1)T+$	−18		ns
$t_{PD\text{-}LCL\text{-}IWR}$	16	$\overline{LCL}$ Rising to $\overline{IWR}$ Falling		−3		ns
$t_{PD\text{-}IWR\text{-}LCL}$	17	$\overline{IWR}$ Rising to $\overline{LCL}$ Falling	T+	−17		ns
$t_{W\text{-}IWR}$	18	$\overline{IWR}$ Low Time	$(n_{IW}+2)T+$	−12		ns
$t_{SU\text{-}CMD\text{-}WPND}$	19	CMD Valid before $\overline{WR\text{-}PEND}$ Rising		16		ns
$t_{H\text{-}CMD\text{-}WPND}$	20	CMD Invalid after $\overline{WR\text{-}PEND}$ Rising		4		ns
$t_{PD\text{-}I\text{-}IA}$	21	I Disabled to IA Invalid	$2T+T_H+$	−70		ns
$t_{SU\text{-}RRWr\text{-}CO}$	22	$\overline{RAE}$, $\overline{REM\text{-}WR}$ Rising before CLK-OUT Rising		20		ns
$t_{PD\text{-}X\text{-}WPND}$	23	XACK Rising to $\overline{WR\text{-}PEND}$ Rising			13	ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

Note 2: Two remote writes to instruction memory are necessary to store a 16-bit instruction word to IMEM—low byte followed by high byte. The timing of the 2nd write is shown in the following diagram. The first write is the same as a write of the PC or RIC as shown in *Figure 5-23*.

Note 3: The maximum value for this parameter is the latest $\overline{RAE}$, $\overline{REM\text{-}WR}$ can be removed without delaying the remote access by one T-state.

5.0 Device Specifications (Continued)



TL/F/9336–79

FIGURE 5-25. Latched Write of IMEM

5.0 Device Specifications (Continued)

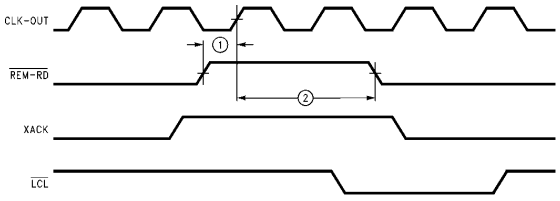
TABLE 5-26. Remote Rest Time (Note 1)

Symbol	ID #	Parameter	Formula	Min	Max	Units
$t_{SU\text{-}BR\text{-}RR\text{-}CO}$	1	$\overline{\text{REM-RD}}$ Rising before CLK-OUT Rising (Buffered Read Mode)		19		ns
$t_{H\text{-}BR}$	2	CLK-OUT Rising after $\overline{\text{REM-RD}}$ Rising to $\overline{\text{REM-RD}}$ or $\overline{\text{REM-WR}}$ Falling (Buffered Read Mode)	$T+T_H+$	10		ns
$t_{SU\text{-}LR\text{-}RR\text{-}CO}$	3	$\overline{\text{REM-RD}}$ Rising before CLK-OUT Rising (Latched Read Mode)		16		ns
$t_{H\text{-}LR}$	4	CLK-OUT Rising after $\overline{\text{REM-RD}}$ Rising to $\overline{\text{REM-RD}}$ or $\overline{\text{REM-WR}}$ Falling (Latched Read Mode)	$T+T_H+$	10		ns
$t_{SU\text{-}SBW\text{-}RW\text{-}CO}$	5	$\overline{\text{REM-WR}}$ Rising before CLK-OUT Rising (Slow Buffered Write Mode)		22		ns
$t_{H\text{-}SBW}$	6	CLK-OUT Rising after $\overline{\text{REM-WR}}$ Rising to $\overline{\text{REM-RD}}$ or $\overline{\text{REM-WR}}$ Falling (Slow Buffered Write Mode)	$T+T_H+$	10		ns
$t_{SU\text{-}FBW\text{-}RW\text{-}CO}$	7	$\overline{\text{REM-WR}}$ Rising before CLK-OUT Rising (Fast Buffered Write Mode)		22		ns
$t_{H\text{-}FBW}$	8	CLK-OUT Rising after $\overline{\text{REM-WR}}$ Rising to $\overline{\text{REM-RD}}$ or $\overline{\text{REM-WR}}$ Falling (Fast Buffered Write Mode)	$T+T_H+$	10		ns
$t_{SU\text{-}LW\text{-}RW\text{-}CO}$	9	$\overline{\text{REM-WR}}$ Rising before CLK-OUT Rising (Latched Write Mode)		20		ns
$t_{H\text{-}LW}$	10	CLK-OUT Rising after $\overline{\text{REM-WR}}$ Rising to $\overline{\text{REM-RD}}$ or $\overline{\text{REM-WR}}$ Falling (Latched Write Mode)		10		ns
$t_{SU\text{-}LW\text{-}RWR\text{-}COa}$	11	$\overline{\text{REM-WR}}$ or$\overline{\text{REM-RD}}$ Falling to CLK-OUT Falling (Latched Write Mode) (Note 2)	T_H+	7		ns
$t_{SU\text{-}LW\text{-}RWR\text{-}COb}$	12	CLK-OUT Rising to $\overline{\text{REM-WR}}$ or $\overline{\text{REM-RD}}$ rising (Latched Write Mode) (Note 2)		8		ns
$t_{PD\text{-}CO\text{-}WP}$	13	CLK-OUT rising to $\overline{\text{WR-PEND}}$ Rising		−1	21	ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

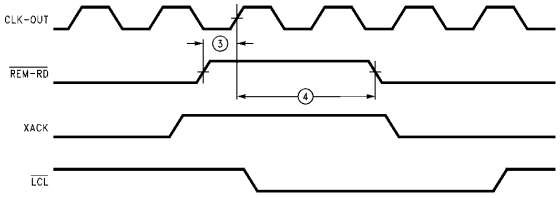
Note 2: Both specifications refer to the CLK-OUT falling edge after $\overline{\text{WR-PEND}}$ rising. See Section 4.2.6, RIAS remote rest time.

5.0 Device Specifications (Continued)



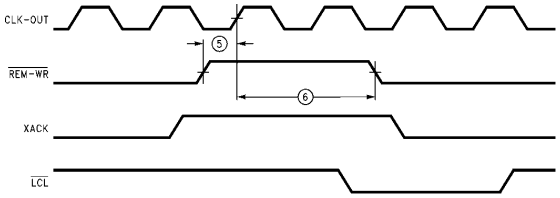
TL/F/9336–B0

(a) REM-RD Rest Time (Buffered Read Mode)



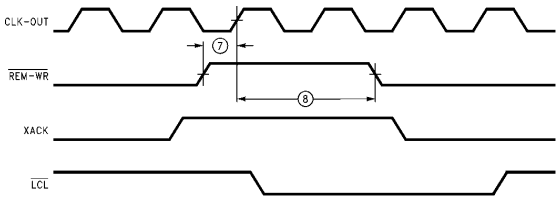
TL/F/9336–B1

(b) REM-RD Rest Time (Latched Read Mode)



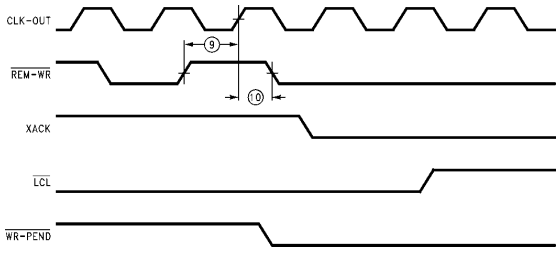
TL/F/9336–B2

(c) REM-WR Rest Time (Slow Buffered Write Mode)



TL/F/9336–B3

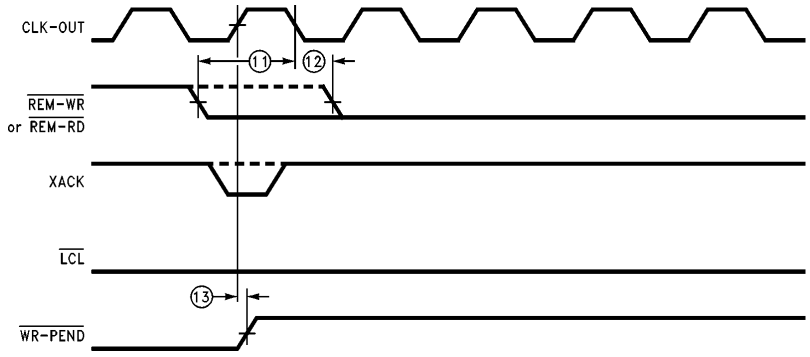
(d) REM-WR Rest Time (Fast Buffered Write Mode)



TL/F/9336–B4

(e) REM-WR Rest Time (Latched Write Mode)

FIGURE 5-26. Remote Rest Time



TL/F/9336–H9

(f) WR-PEND Rising (Latched Write Mode)

FIGURE 5-26. Remote Rest Time (Continued)

TABLE 5-27. Remote Interface WAIT Timing (Note 1)

Symbol	ID #	Parameter	Formula	Min	Max	Units
$t_{SU\text{-}WT\text{-}LCL}$	1	WAIT Falling after LCL Rising to Extend Cycle (Buffered Read, Latched Read, Slow Buffered Write, Fast Buffered Write and Latched Write of PC, RIC)	$T+T_H+$		−28	ns
		WAIT Falling after LCL Rising to Extend Cycle (Buffered Read, Latched Read, Slow Buffered Write, Fast Buffered Write and Latched Write of DMEM)	$(n_{DW}+1)T+T_H+$		−28	ns
		WAIT Falling after LCL Rising to Extend Cycle (Buffered Read, Latched Read, Slow Buffered Write, Fast Buffered Write and Latched Write of IMEM)	$(n_{IW}+1)T+T_H+$		−28	ns
$t_{H\text{-}WT\text{-}LCL}$	2	WAIT Rising after LCL Rising (Buffered Read, Latched Read, Slow Buffered Write, Fast Buffered Write and Latched Write of PC, RIC) (Note 2)	$T+T_H+$	0		ns
			$2T+T_H+$		−27	ns
		WAIT Rising after LCL Rising (Buffered Read, Latched Read, Slow Buffered Write, Fast Buffered Write and Latched Write of DMEM) (Note 2)	$(n_{DW}+1)T+T_H+$	0		ns
			$(n_{DW}+2)T+T_H+$		−27	ns
		WAIT Rising after LCL Rising (Buffered Read, Latched Read, Slow Buffered Write, Fast Buffered Write and Latched Write of IMEM) (Note 2)	$(n_{IW}+1)T+T_H+$	0		ns
			$(n_{IW}+2)T+T_H+$		−27	ns
$t_{SU\text{-}WT\text{-}RD}$	3	WAIT Falling after READ Falling to Extend Cycle (Buffered Read and Latched Read)	$(n_{DW})T+T_H+$		−32	ns
$t_{SU\text{-}WT\text{-}WR}$	3	WAIT Falling after WRITE Falling to Extend Cycle (Slow Buffered Write, Fast Buffered Write and Latched Write)	$(n_{DW})T+T_H+$		−33	ns
$t_{SU\text{-}WT\text{-}IWR}$	3	WAIT Falling after IWR Falling to Extend Cycle (Slow Buffered Write, Fast Buffered Write and Latched Write)	$(n_{IW}+1)T+T_H+$		−38	ns

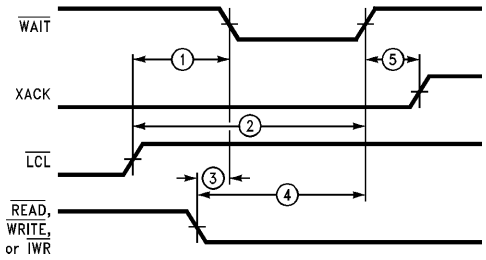
5.0 Device Specifications (Continued)

TABLE 5-27. Remote Interface $\overline{\text{WAIT}}$ Timing (Note 1) (Continued)

Symbol	ID#	Parameter	Formula	Min	Max	Units
$t_{H\text{-}WT\text{-}RD}$	4	$\overline{\text{WAIT}}$ Rising after $\overline{\text{READ}}$ Falling (Buffered Read and Latched Read) (Note 2)	$(n_{DW}T+T_H+$	−4		ns
			$(n_{DW}+1)T+T_H+$		−30	ns
$t_{H\text{-}WT\text{-}WR}$	4	$\overline{\text{WAIT}}$ Rising after $\overline{\text{WRITE}}$ Falling (Slow Buffered Write, Fast Buffered Write and Latched Write) (Note 2)	$(n_{DW})T+T_H+$	−5		ns
			$(n_{DW}+1)T+T_H+$		−34	ns
$t_{H\text{-}WT\text{-}IWR}$	4	$\overline{\text{WAIT}}$ Rising after $\overline{\text{IWR}}$ Falling (Slow Buffered Write, Fast Buffered Write and Latched Write) (Note 2)	$(n_{IW}+1)T+T_H+$	−5		ns
			$(n_{IW}+2)T+T_H+$		−38	ns
$t_{PD\text{-}WT\text{-}X}$	5	$\overline{\text{WAIT}}$ Rising to XACK Rising (Buffered Read, Latched Read, Slow Buffered Write and Fast Buffered Write)	T_L+	0		ns
			$T+T_L+$		24	ns
$t_{PD\text{-}WT\text{-}LCL}$	6	$\overline{\text{WAIT}}$ Rising to $\overline{\text{LCL}}$ Falling (Latched Write)	$T+T_L+$	1		ns
			$2T+T_L+$		26	ns
$t_{PD\text{-}WT\text{-}WR}$	7	$\overline{\text{WAIT}}$ Rising to $\overline{\text{WRITE}}$ Rising (Latched Write)	T_L+	2		ns
			$T+T_L+$		28	ns
$t_{PD\text{-}WT\text{-}IWR}$	7	$\overline{\text{WAIT}}$ Rising to $\overline{\text{IWR}}$ Rising (Latched Write)	T_L+	4		ns
			$T+T_L+$		38	ns

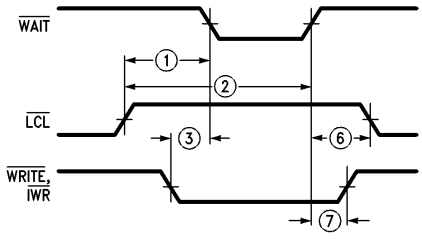
Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

Note 2: The maximum value for this parameter is the latest $\overline{\text{WAIT}}$ can be removed without adding an additional T-state. The formula assumes a minimum external wait of one T-state.



TL/F/9336–B5

(a) Buffered Read, Latched Read, Slow Buffered Write and Fast Buffered Write



TL/F/9336–B6

(b) Latched Write

FIGURE 5-27. Remote Interface $\overline{\text{WAIT}}$ Timing

5.0 Device Specifications (Continued)

TABLE 5-28. Wait Timing After Remote Access (Note 1)

Symbol	ID#	Parameter	Formula	Min	Max	Units
$t_{PD-LCL-AAD}$	1	$\overline{LCL}$ Falling to A, AD (Data Address) Valid	T_H+		11	ns
$t_{PD-LCL-AAD-BR}$		$\overline{LCL}$ Falling to A, AD (Data Address) Valid for Buffered Read of RIC	$2T+$		29	ns
$t_{PD-AAD-ALE}$	2	A, AD (Data Address) Valid to ALE Falling	$T+$	−16		ns
$t_{SU-WT-LCL}$	3	$\overline{LCL}$ Falling to $\overline{WAIT}$ Falling to Extend Local Cycle	$(\max(n_{DW},n_{IW}-1)+1)T+T_H+$		−29	ns
$t_{H-WT-LCL}$	4	$\overline{WAIT}$ Rising after $\overline{LCL}$ Falling	$(\max(n_{DW},n_{IW}-1)+1)T+T_H+$	−3		ns
			$(\max(n_{DW},n_{IW}-1)+2)T+T_H+$		−28	ns
$t_{H-WT-LCL-BR}$		$\overline{WAIT}$ Rising after $\overline{LCL}$ Falling for Buffered read of RIC	$(\max(n_{DW},n_{IW}-1)+3)T+T_H+$	−3		ns
$t_{SU-WT-ALEf}$	5	$\overline{WAIT}$ Low Before ALE Falling to Extend Cycle		22		ns
$t_{H-WT-ALE}$	6	$\overline{WAIT}$ Rising After ALE Falling		0		ns
			$(\max(n_{DW},n_{IW}-1)+1)T+$		−28	ns
$t_{SU-WT-AAD}$	7	A, AD (Data Address) Valid to $\overline{WAIT}$ Falling to Extend Load Cycle	$T+$	−33		ns

Note 1: All parameters are individually tested and guaranteed. Interpreting this data by numerically adding two or more parameters to create a new timing specification may lead to invalid results.

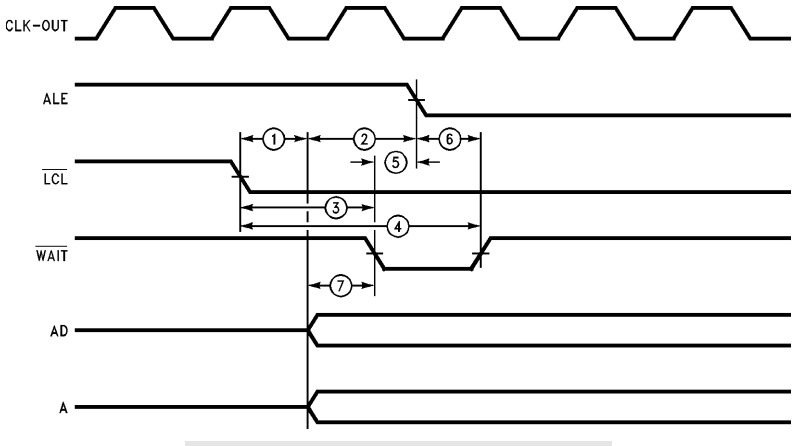


FIGURE 5-28. Wait Timing After Remote Access

6.0 Reference Section

6.1 INSTRUCTION SET REFERENCE

The Instruction Set Reference section contains detailed information on the syntax and operation of each BCP instruction. The instructions are arranged in alphabetical order by mnemonic for easy access. Although this section is primarily intended as a reference for the assembly language programmer, previous assembly language experience is not a prerequisite. The intent of this instruction set reference is to include all the pertinent information regarding each instruction on the page(s) describing that instruction. The only exceptions to this rule concern the instruction addressing modes and the bus timing diagrams. The discussion of the instruction addressing modes occurs at the beginning of the BCP Instruction Set Overview section and, therefore, will not be repeated here. The figures for the bus timing diagrams are located at the end of this introduction rather than constantly repeating them under each instruction. The information that is contained under each instruction is divided into eight categories titled: Syntax, Affected Flags, Description, Example, Instruction Format, T-states, Bus timing, and Operation. The following paragraphs explain what information each category conveys and any special nomenclature that a category may use.

Syntax

This category illustrates the assembler syntax for each instruction. Multiple lines are used when a given instruction supports more than one type of addressing mode, or if it has an optional mnemonic. All capital letters, commas (,), math symbols (+, −), and brackets ([ ]) are entered into the assembler exactly as shown. Braces ({ }) surround an instruction's optional operands and their associated syntax. The text between the braces may either be entered in with or omitted from the instruction. The braces themselves should not be entered into the assembler because they are not part of the assembler syntax. Lower case characters and operands that begin with the capital R represent symbols. These must be replaced with actual register names, numbers, or equated registers and numbers. Table 6-1 lists all the symbols and their associated meanings.

Affected Flags

If an instruction sets or clears any of the ALU flags, (i.e., Negative [N], Zero [Z], Carry [C], and/or Overflow [V]), then those flags affected are listed under this category.

Description

The Description category contains a verbal discussion about the operation of an instruction, the operands it allows, and any notes highlighting special considerations the programmer should keep in mind when using the instruction.

Example

Each instruction has one or more coding examples designed to show its typical usage(s). For clarity, register name abbreviations are often used instead of the register numbers, (i.e., RTR is used in place of R4). Each example assumes that the ''.EQU'' assembler directive has been previously executed to establish these relationships. Information relating register abbreviations to register names, numbers, and purpose is located in the CPU Registers section.

Instruction Format

This category illustrates the formation of an instruction's machine code for each operand variation. Assembly or disassembly of any instruction can be accomplished using these figures.

T-states

The T-state category lists the number of CPU clock cycles required for each instruction, including operand variations and conditional considerations. Using this information, actual execution times may be calculated. For example, if the conditional relative jump instruction's condition is not met, the CPU's clock cycle is 18.867 MHz ([CCS] = 0), and no instruction wait states are requested ([IW1−0] = 00), then Jcc's execution time is calculated as shown below:

$$\begin{aligned} t_{execution} &= 1/(\text{CPU clock frequency}) \times \text{T-states} \\ &= 1/(18.867 \times 10^6\ \text{Hz}) \times 2 \\ &= (53 \times 10^{-9}\text{s}) \times 2 \\ &= 106\ \text{ns} \end{aligned}$$

See the section BCP Timing for more information on calculating instruction execution times.

Bus Timing

This category refers the user to the Bus Timing *Figures 6-1* to *6-6* on the following pages. These figures illustrate the relationship between software instruction execution and some of the BCP's hardware signals.

Operation

The operation category illustrates each instruction's operation in a symbolic coding format. Most of the operand names used in this format come directly from each instruction's syntax. The exceptions to this rule deal with implied operands. Instructions that imply the use of the accumulators use the name ''accumulator'' as an operand. Instructions that manipulate the Program Counter use the symbol ''PC''. Instructions that ''push'' onto or ''pop'' off of the internal Address Stack specify ''Address Stack'' as an operand. Instructions that save or restore the ALU flags and the register bank selections use those terms as operands. Two specialized operator symbols are used in the symbolic coding format, the arrow '' ⟶ '' and the concatenation operator ''&''. The arrow indicates the movement of data from one operand to another. For instance, after the operation ''Rs ⟶ Rd'' is performed the content of Rd has been replaced with the content of Rs. The concatenation operator ''&'' simply indicates that the operands surrounding an ''&'' are attached together forming one new operand. For example, ''PC & [GIE] & ALU flags & register bank selections ⟶ Address Stack'' means that the Program Counter, the Global Interrupt Enable bit, the ALU flags and the register bank selections are combined into one operand and pushed onto the internal Address Stack. Three conditional structures are utilized in the symbolic coding format: the ''Two Line If'' structure, the ''Blocked If'' structure, and the ''Blocked Case'' structure. In the ''Two Line If'' structure, if the *condition* is met then the *operation* is performed, otherwise the *operation* is not performed.

''Two Line If'' structure:

If *condition*
  then *operation*

6.0 Reference Section (Continued)

In the ''Blocked If'' structure, if the *condition* is met then all the *operations* between the ''If'' statement and the ''End if'' statement are performed.

''Blocked If'' structure:

```
If condition then
  operation
  operation
  etc . . .
End if
```

In the ''Blocked Case'' structure, the *operation* preceded by the equivalent numeric value of the *operand* is executed. For example, if the *operand*'s value is equal to ''1'' then the *operation* preceded by ''1:'' is executed.

''Blocked Case'' structure:

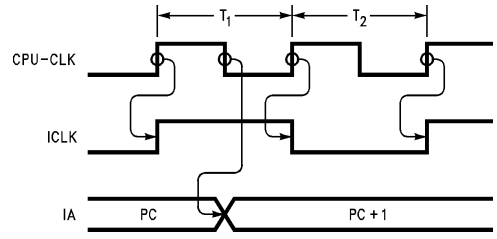
```
Case operand of
  0: operation
  1: operation
  2: etc . . .
End case
```

Two reference tables have been added to the back of the Instruction Set Reference section. The first table, Table 6-2, lists all the instructions with their associated T-states, Affected Flags, and Bus Timing figure numbers in a compact format. The second table, Table 6-3, lists all the instructions in opcode order to facilitate disassembly.

TABLE 6-1. Notational Conventions for Instruction Set

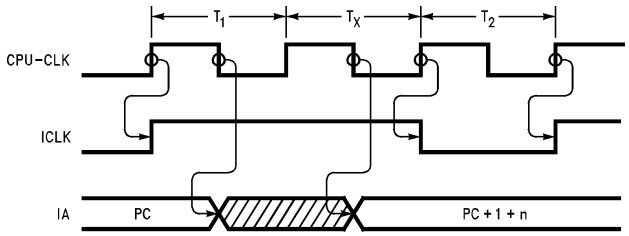
Symbol	Represents	Meaning	Length
n	0 to 255 +127 to −128	Unsigned Number Signed Number	8 Bits
nn	0 to 65535	Unsigned Number	16 Bits
Rs	R0–R31	Source Register	
Rd	R0–R31	Destination Register	
Rsd	R0–R31	Combination Source/Destination Register	
rs	R0–R15	Limited Source Register	
rd	R0–R15	Limited Destination Register	
rsd	R0–R15	Limited Combination Source/Destination Register	
Ir	IW, IX, IY, IZ	Index Register	
mIr	 Ir− Ir Ir+ +Ir	Index Register in One of the Following Address Modes: Post Decrement No Change Post Increment Pre-Increment	
b	0–7	Shift Field	3 Bits
m	0–7	Mask Field	3 Bits
p	0–7	Position Field	3 Bits
s	0–1	State Field	1 Bit
f	0–7	Flag Reference Field	3 Bits
cc		Condition Code Instruction Extensions	
v	0–63	Vector Field	6 Bits
g	0–3	Global Interrupt Enable Flag [GIE] Status Control	2 Bits
g′	0–1	Global Interrupt Enable Flag [GIE] Limited Status Control	1 Bit
rf	0–1	Register Bank and ALU Flag Status Control	1 Bit
ba	0–1	Register Bank A Select	1 Bit
bb	0–1	Register Bank B Select	1 Bit

6.0 Reference Section (Continued)



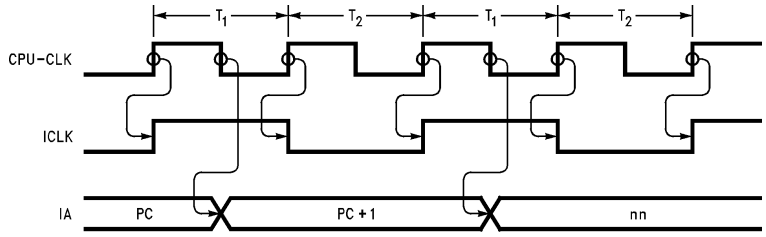
TL/F/9336–21

FIGURE 6-1. Instruction-Memory Bus Timing for 2 T-state Instructions (No Instruction Wait States [IW1–0] = 00, CPU Running at Full Speed [CCS] = 0)



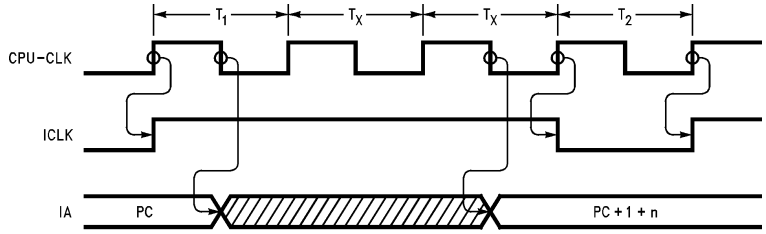
TL/F/9336–22

FIGURE 6-2. Instruction-Memory Bus Timing for 3 T-state Instructions (No Instruction Wait States [IW1–0] = 00, CPU Running at Full Speed [CCS] = 0)



TL/F/9336–23

FIGURE 6-3. Instruction-Memory Bus Timing for (2 + 2) T-state Instructions (No Instruction Wait States [IW1–0] = 00, CPU Running at Full Speed [CCS] = 0)



TL/F/9336–24

FIGURE 6-4. Instruction-Memory Bus Timing for 4 T-state Instructions (No Instruction Wait States [IW1–0] = 00, CPU Running at Full Speed [CCS] = 0)

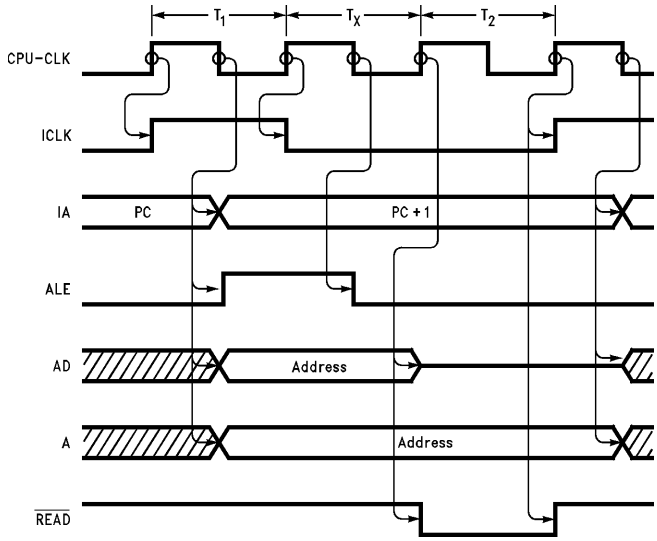


FIGURE 6-5. Instruction/Data Memory Bus Timing for Data Memory Read (No Instruction or Data Memory Wait States, CPU Running at Full Speed [CCS] = 0, [4TR] = 0)

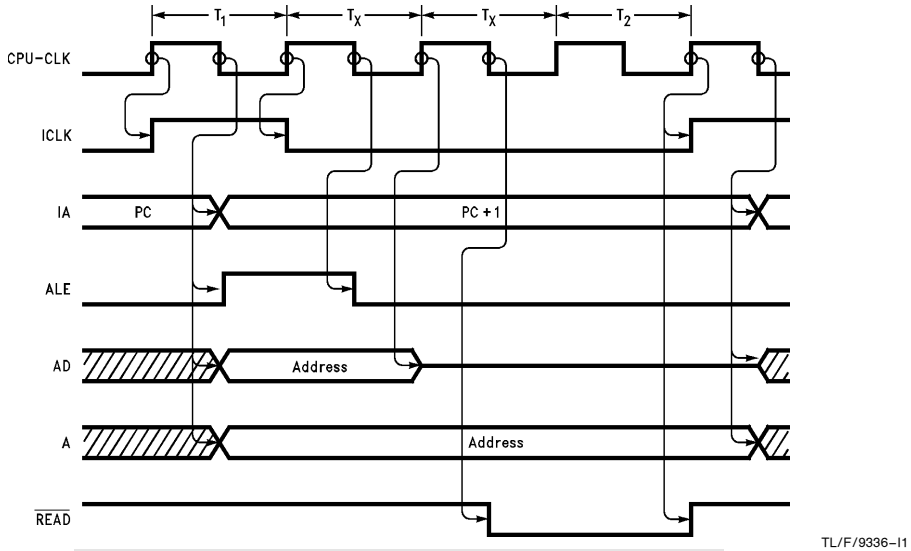


FIGURE 6-6. Instruction/Data Memory Bus Timing for Data Memory Read (No Instruction or Data Memory Wait States, CPU Running at Full Speed [CCS] = 0), [4TR] = 1)

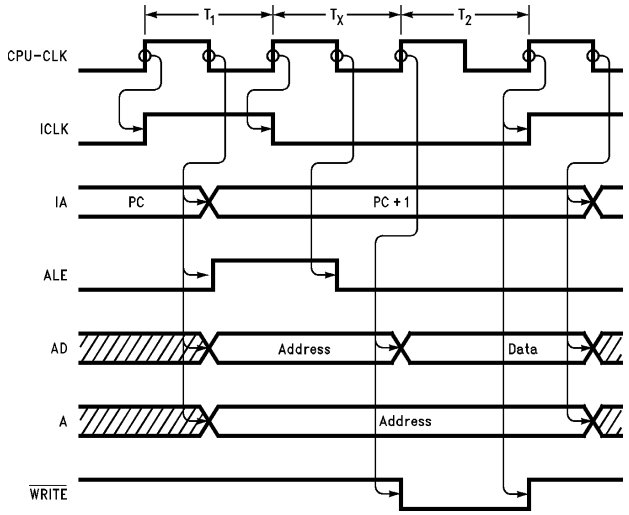


FIGURE 6-7. Instruction/Data Memory Bus Timing for Data Memory Write (No Instruction or Data Memory Wait States, CPU Running at Full Speed [CCS] = 0)

ADCA Add with Carry and Accumulator

Syntax

ADCA Rs, Rd —register, register
ADCA Rs, [mlr] —register, indexed

Affected Flags

N, Z, C, V

Description

Adds the source register Rs, the active accumulator, and the carry flag together, placing the result into the destination specified. The destination may be either a register, Rd, or data memory via an index register mode, [mlr]. Note that register bank selection determines which accumulator is active.

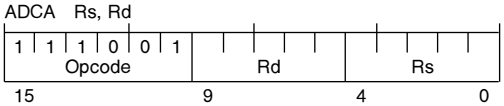
Example

Add the constant 109 to the index register IW, (which is 16 bits wide).

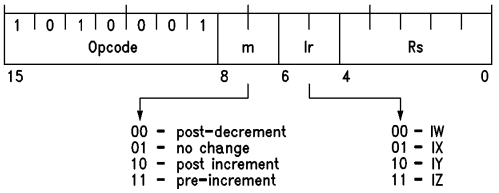
```
SUBA   A, A        ;Clear the accumulator
ADD    109, R12    ;Add 109 to low byte of IW
ADCA   R13, R13    ;Add carry to high byte of IW
```

Instruction Format

ADCA Rs, Rd



ADCA Rs, [mlr]



TL/F/9336–5

T-states

ADCA Rs, Rd —2
ADCA Rs, [mlr] —3

Bus Timing

ADCA Rs, Rd —*Figure 6-1*
ADCA Rs, [mlr] —*Figure 6-6*

Operation

ADCA Rs, Rd
Rs + accumulator + carry bit ⟶ Rd
ADCA Rs, [mlr]
Rs + accumulator + carry bit ⟶ data memory

ADD Add Immediate

Syntax

ADD n, rsd —immediate, limited register

Affected Flags

N, Z, C, V

Description

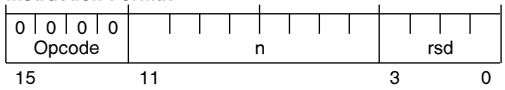
Adds the immediate value n to the register rsd and places the result back into the register rsd. Note that only the active registers R0–R15 may be specified for rsd. The value of n is limited to 8 bits; (unsigned range: 0 to 255, signed range: +127 to −128).

Example

Add the constant −3 to register 10.

```
ADD    −3, R10        ;R10 + (−3) ⟶ R10
```

Instruction Format



T-States

2

Bus Timing

Figure 6-1

Operation

rsd + n ⟶ rsd

ADDA Add with Accumulator

Syntax

ADDA Rs, Rd —register, register
ADDA Rs, [mIr] —register, indexed

Affected Flags

N, Z, C, V

Description

Adds the source register Rs to the active accumulator and places the result into the destination specified. The destination may be either a register, Rd, or data memory via an index register mode, [mIr]. Note that register bank selection determines which accumulator is active.

Example

In the first example, the value 4 is placed into the currently active accumulator, that accumulator is added to the contents of register 20, and then the result is placed into register 21.

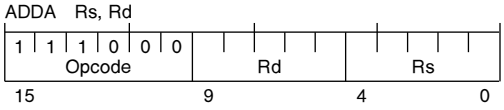
```
MOVE   4, A           ;Place constant into accum
ADDA   R20, R21       ;R20 + accum ⟶ R21
```

In the second example, the alternate accumulator of register bank B is selected and then added to register 20. The result is placed into the data memory pointed to by the index register IZ and then the value of IZ is incremented by one.

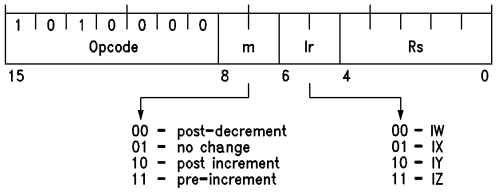
```
EXX    0, 1           ;Select alt accumulator
ADDA   R20, [IZ+]     ;R20 + accum ⟶ data mem
                      ;and increment data pointer
```

Instruction Format

ADDA Rs, Rd



ADDA Rs, [mIr]



TL/F/9336–6

T-states

ADDA Rs, Rd —2
ADDA Rs, [mIr] —3

Bus Timing

ADDA Rs, Rd —*Figure 6-1*
ADDA Rs, [mIr] —*Figure 6-6*

Operation

ADDA Rs, Rd
Rs + accumulator ⟶ Rd

ADDA Rs, [mIr]
Rs + accumulator ⟶ data memory

AND And Immediate

Syntax

AND n, rsd —immediate, limited register

Affected Flags

N, Z

Description

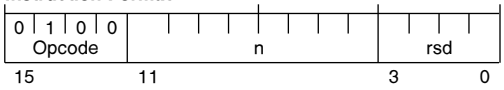
Logically ANDs the immediate value n to the register rsd and places the result back into the register rsd. Note that only the active registers R0–R15 may be specified for rsd. The value of n is 8 bits wide.

Example

Unmask both the Transmitter and Receiver interrupts via the Interrupt Control Register {ICR}, R2. Leave the other interrupts unaffected.

```
EXX    0,0            ;select main register banks
AND    11111100B,R2   ;unmask transmitter and
                      ; receiver interrupts
```

Instruction Format



T-states

2

Bus Timing

Figure 6-1

Operation

rsd AND n ⟶ rsd

ANDA And with Accumulator

Syntax

ANDA Rs, Rd —register, register
ANDA Rs, [mIr] —register, indexed

Affected Flags

N, Z

Description

Logically ANDs the source register Rs to the active accumulator and places the result into the destination specified. The destination may be either a register, Rd, or data memory via an index register mode, [mIr]. Note that register bank selection determines which accumulator is active.

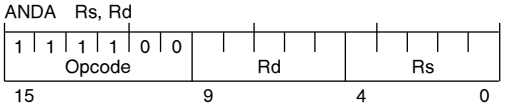
Example

This example demonstrates a way to quickly unload all 11 bits of the three words in the Receiver FIFO when the FIFO is full. The example assumes that the index register IZ points to the location in data memory where the information should be stored.

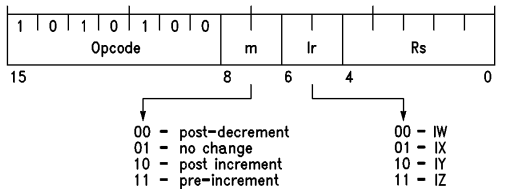
```
    EXX    1,1              ;select alternate banks
    MOVE   00000111B, A     ;place the {TSR} mask
                            ; into the accumulator
;   Pop the first word from the receiver FIFO
    ANDA   TSR, [IZ+]       ;read bits 8, 9, & 10
    MOVE   RTR, [IZ+]       ;pop bits 0–7
;   Pop the second word from the receiver FIFO
    ANDA   TSR, [IZ+]
    MOVE   RTR, [IZ+]
;   Pop the third word from the receiver FIFO
    ANDA   TSR, [IZ+]
    MOVE   RTR, [IZ+]
```

Instruction Format

ANDA Rs, Rd



ANDA Rs,[mIr]



TL/F/9336–7

T-states

ANDA Rs, Rd —2
ANDA Rs, [mIr] —3

Bus Timing

ANDA Rs, Rd —*Figure 6-1*
ANDA Rs, [mIr] —*Figure 6-7*

Operation

ANDA Rs, Rd
Rs AND accumulator ⟶ Rd

ANDA Rs, [mIr]
Rs AND accumulator ⟶ data memory

BIT Bit Test

Syntax

BIT rs, n —limited register, immediate

Affected Flags

N, Z

Description

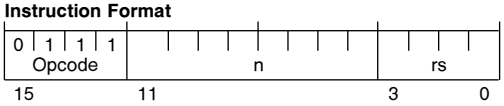
Performs a bit level test by logically ANDing the source register rs to the immediate value n. The affected flags are updated, but the result is not saved. Note that only the active registers R0–R15 may be specified for rs. The value n is 8 bits wide.

Example

Poll the Transmitter FIFO Empty flag [TFE] in the Network Command Flag register {NCF}, R1, waiting for the Transmitter to send the current FIFO data.

```
        EXX   0,1                ;select main A, alt B
Poll:   BIT   NCF,10000000B      ;All data sent yet?
        JZ    Poll               ; No, poll TFE
        . . .                    ; Yes, send next byte(s)
```

Instruction Format



T-states

2

Bus Timing

Figure 6-1

Operation

rs AND n

CALL Unconditional Relative Call

Syntax

CALL n —immediate

Affected Flags

None

Description

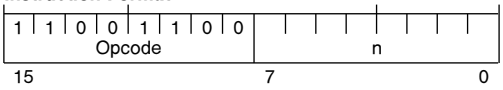
Pushes the Program Counter, the ALU flags, the Global Interrupt Enable bit [GIE], and the current register bank selections onto the internal Address Stack; then unconditionally transfers control to the instruction at the memory address calculated by adding the contents of the Program Counter to the immediate value n, (sign extended to 16 bits). Since the immediate value n is an 8-bit two's complement displacement, the unconditional relative call's range is from +127 to −128 relative to the Program Counter. Note that the Program Counter initially contains the memory address of the next instruction following the call.

Example

Transfer control to the subroutine ''Send.it''. Note that ''Send.it'' must be within +127/−128 words relative to the PC.

```
CALL    Send.it
```

Instruction Format



T-states

3

Bus Timing

Figure 6-2

Operation

PC & [GIE] & ALU flags & register bank selections
⟶ Address Stack
PC + n(sign extended) ⟶ PC

CMP Compare

Syntax

CMP rs, n —limited register, immediate

Affected Flags

N, Z, C, V

Description

Compares the immediate value n with the source register rs by subtracting n from rs. The affected flags are updated, but the result is not saved. Note that only the active registers R0–R15 may be specified for rs. The value of n is limited to 8 bits; (unsigned range: 0 to 255, signed range: +127 to −128).

Example

Compare the data byte in register 11 to the ASCII character ''A''.

```
CMP     R11,''A''           ;If:
JC      Less__than__A       ;     data<''A''
JEQ     Equal__to__A        ;     data=''A''
 . . .                      ;else data>''A''
```

Compare the contents of register 8 to the value 25.

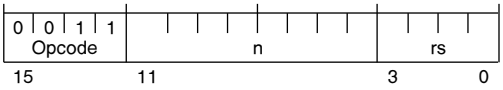
```
CMP     R8,25               ;if:
BIT     CCR,00000011B       ;     data > 25
JZ      Greater__than       ;     Goto Greater__than
```

Comparing of Unsigned Values

Comparison	Flag(s) to Test
LT (<)	C
LEQ (<=)	C|Z
EQ (=)	Z
GEQ (>=)	$\overline{C}$
GT (>)	$\overline{C}$ & $\overline{Z}$

Note: & = logical AND
| = logical OR

Instruction Format



T-states

2

Bus Timing

Figure 6-1

Operation

rs − n

CPL Complement

Syntax

CPL Rsd —register

Affected Flags

N, Z

Description

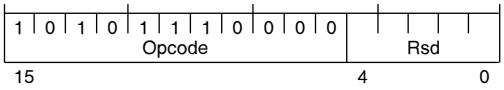
Logically complements the contents of the register Rsd, placing the result back into that register.

Example

Load the fill-bit count passed from the host into the Transmitter's Fill-Bit Register {FBR}, R3, and then perform the required one's complement of the fill-bit count. In this example, register 20 contains the fill-bit count.

```
EXX     1,1       ;select alternate banks
MOVE    R20, FBR  ;load {FBR}
CPL     FBR       ;complement fill-bit count
```

Instruction Format



T-states

2

Bus Timing

Figure 6-1

Operation

$\overline{Rsd} \rightarrow Rsd$

EXX Exchange Register Banks

Syntax

EXX ba, bb {,g}

Affected Flags

None

Description

Selects which CPU register banks are active by exchanging between the main and alternate register sets for each bank. Bank A controls R0–R3 and Bank B controls R4–R11. The table below shows the four possible register bank configurations. Note that deactivated registers retain their current values. The Global Interrupt Enable bit [GIE] can be set or cleared, if desired.

Register Bank Configurations

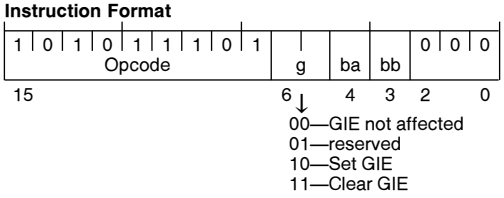
ba	bb	Active Register Banks
0	0	Main A, Main B
0	1	Main A, Alternate B
1	0	Alternate A, Main B
1	1	Alternate A, Alternate B

Example

Activate the main register set of Bank A, the alternate register set of Bank B, and leave the Global Interrupt Enable bit [GIE] unchanged.

```
EXX     0,1       ;select main A, alt B reg banks
```

Instruction Format



T-states

2

Bus Timing

Figure 6-1

Operation

```
Case ba of
  0:  activate main Bank A
  1:  activate alternate Bank A
End case
Case bb of
  0:  activate main Bank B
  1:  activate alternate Bank B
End case
Case g of
  0:  leave [GIE] unaffected, (default)
  1:  (reserved)
  2:  set [GIE]
  3:  clear [GIE]
End case
```

JMP Conditional Relative Jump

Jcc

Syntax

JMP f, s, n —immediate
Jcc n —immediate (optional syntax)

Affected Flags

None

Description

Conditionally transfers control to the instruction at the memory address calculated by adding the contents of the Program Counter to the immediate value n, (sign extended to 16 bits), if the state of the flag referenced by f is equal to the state of the bit s; or, optionally, if the condition cc is met. See the tables below for the flags that f can reference and the conditions that cc may specify. Since the immediate value n is an 8-bit two's complement displacement, the conditional relative jump's range is from +127 to −128 relative to the Program Counter. Note that the Program Counter initially contains the memory address of the next instruction following the jump.

Example

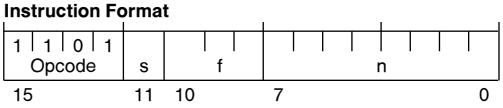
This example demonstrates both syntaxes of the conditional relative jump instruction testing for a non-zero result from a previous instruction; (i.e., [Z] = 0). If the condition is met then control transfers to the instruction labeled "Loop.back"; else the next instruction following the jump is executed.

```
JMP     000B,0,Loop.back     ;jump on not zero
 . . .
JNZ     Loop.back            ;jump on not zero
```

Condition Specification Table for "cc"

cc	Meaning	Condition Tested for
Z	Zero	[Z] = 1
NZ	Not Zero	[Z] = 0
EQ	Equal	[Z] = 1
NEQ	Not Equal	[Z] = 0
C	Carry	[C] = 1
NC	No Carry	[C] = 0
V	Overflow	[V] = 1
NV	No Overflow	[V] = 0
N	Negative	[N] = 1
P	Positive	[N] = 0
RA	Receiver Active	[RA] = 1
NRA	Not Receiver Active	[RA] = 0
RE	Receiver Error	[RE] = 1
NRE	No Receiver Error	[RE] = 0
DA	Data Available	[DAV] = 1
NDA	No Data Available	[DAV] = 0
TFF	Transmitter FIFO Full	[TFF] = 1
NTFF	Transmitter FIFO Not Full	[TFF] = 0

Instruction Format



T-states

2 if condition is not met
3 if condition is met

Bus Timing

Figure 6-1 if condition is not met
Figure 6-2 if condition is met

Operation

JMP f, s, n
If flag f is in state s
  then PC + n(sign extended) ⟶ PC

Jcc n
If cc condition is true
  then PC + n(sign extended) ⟶ PC

Flag Reference Table for "f"

f	(binary)	Flag Reference
0	(000)	[Z] in {CCR}
1	(001)	[C] in {CCR}
2	(010)	[V] in {CCR}
3	(011)	[N] in {CCR}
4	(100)	[RA] in {TSR}
5	(101)	[RE] in {TSR}
6*	(110)	[DAV] in {TSR}
7	(111)	[TFF] in {TSR}

*__Note:__ The value of f for [DAV] differs from the numeric value for the position of [DAV] in {TSR}.

JMP Unconditional Relative Jump

Syntax

JMP n —immediate
JMP Rs —register

Affected Flags

None

Description

Unconditionally transfers control to the instruction at the memory address calculated by adding the contents of the Program Counter to either the immediate value n or the contents of the source register Rs, (both sign extended to 16 bits). Since the immediate value n and the contents of Rs are 8-bit two's complement displacements, the unconditional relative jump's range is from +127 to −128 relative to the Program Counter. Note that the Program Counter initially contains the memory address of the next instruction following the jump.

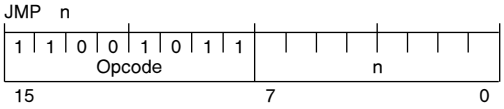
Example

Transfer control to the instruction labeled "Init_Xmit", which is within +127/−128 words relative to the PC.

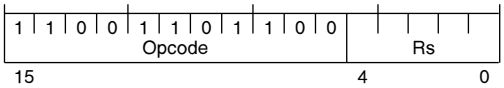
```
JMP   Init_Xmit     ;go initialize Transmitter
```

Instruction Format

JMP n



JMP Rs



T-states

JMP n —3
JMP Rs —4

Bus Timing

JMP n —*Figure 6-2*
JMP Rs —*Figure 6-4*

Operation

JMP n
PC + n(sign extended) ⟶ PC

JMP Rs
PC + Rs(sign extended) ⟶ PC

JRMK Relative Jump with Rotate and Mask on Register

Syntax

JRMK Rs, b, m —register

Affected Flags

None

Description

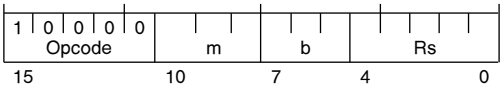
Transfers control to the instruction at the memory address calculated by adding the contents of the Program Counter to a specially formed displacement. The displacement is formed by rotating a copy of the source register Rs the value of b bits to the right, masking (setting to zero) the most significant m bits, masking the least significant bit, and then sign extending the result to 16 bits. Typically, the JRMK instruction transfers control into a jump table. The LSB of the displacement is always set to zero so that the jump table may contain two word instructions, (e.g., LJMP). The range of JRMK is from +126 to −128 relative to the Program Counter. Note that the Program Counter initially contains the memory address of the next instruction following JRMK. The source register Rs may specify any active CPU register. The rotate value b may be from 0 to 7, where 0 causes no bit rotation to occur. The mask value m may be from 0 to 7; where m=0 causes only the LSB of the displacement to be masked, m=1 causes the MSB and the LSB to be masked, m=2 causes bits 7–6 and the LSB to be masked, etc …

Example

This example demonstrates the decoding of the address frame of the 3299 Terminal Multiplexer protocol. In the address frame, only the bits 4–2 contain the address of the Logical Unit.

```
EXX     0,1         ;select main A, alt B
JRMK    RTR,1,4     ;decode device address
LJMP    ADDR.0      ;jump to device handler #0
LJMP    ADDR.1      ;jump to device handler #1
LJMP    ADDR.2      ;jump to device handler #2
 . . .
LJMP    ADDR.7      ;jump to device handler #7
```

Instruction Format



T-states

4

Bus Timing

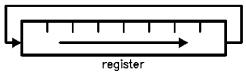
Figure 6-4

Operation

Copy Rs to a temporary register:

Rs ⟶ register

Rotate the register b bits to the right:



TL/F/9336–8

Mask the most significant m bits and the LSB:

register AND 0 . . . 0 1 . . . 1 0 ⟶ register

Modify the Program Counter:

PC + register(sign extended) ⟶ PC

LCALL Conditional Long Call

Syntax

LCALL Rs, p, s, nn —register, absolute

Affected Flags

None

Description

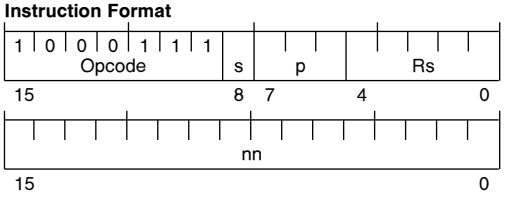
If the bit in position p of register Rs is equal to the bit s, then push the Program Counter, the ALU flags, the Global Interrupt Enable bit [GIE], and the current register bank selections onto the internal Address Stack. Following the push, transfer control to the instruction at the absolute memory address nn. The operand Rs may specify any active CPU register. The value of p may be from 0 to 7, where 0 corresponds to the LSB of Rs and 7 corresponds to the MSB of Rs. The absolute value nn is 16 bits long, (range: 0 to 64k), therefore, all of instruction memory can be addressed.

Example

Call the ''Load.Xmit'' subroutine when the Transmitter FIFO Empty flag, [TFE], of the Network Command Flag register {NCF} is ''1''.

```
EXX     0,0                     ;select main A, alt B
LCALL   NCF,7,1, Load.Xmit      ;If [TFE]=1 call
```

Instruction Format



T-states

(2 + 2)

Bus Timing

Figure 6-3

Operation

If Rs[p] = s then
  PC & [GIE] & ALU flags & register bank selections
    → Address Stack
  nn → PC
End if

LCALL Unconditional Long Call

Syntax

LCALL nn —absolute

Affected Flags

None

Description

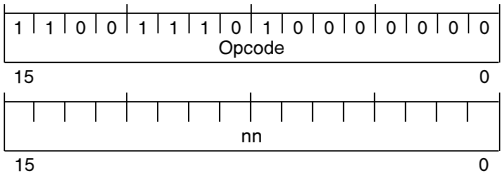
Pushes the Program Counter, the ALU flags, the Global Interrupt Enable bit [GIE], and the current register bank selections onto the internal Address Stack; then unconditionally transfers control to the instruction at the absolute memory address nn. The value of nn is 16 bits long, (range: 0 to 64k), therefore, all of instruction memory can be addressed.

Example

Transfer control to the subroutine ''Send.it.all'', which could be located anywhere in instruction memory.

```
LCALL     Send.it.all
```

Instruction Format



T-states

(2 + 2)

Bus Timing

Figure 6-3

Operation

PC & [GIE] & ALU flags & register bank selections
  → Address Stack
nn → PC

LJMP Conditional Long Jump

Syntax

LJMP Rs, p, s, nn —register, absolute

Affected Flags

None

Description

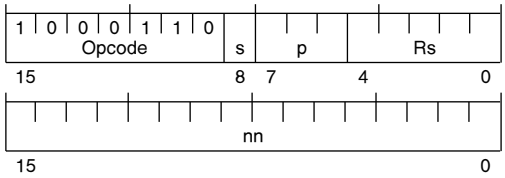
Conditionally transfers control to the instruction at the absolute memory address nn if the bit in position p of register Rs is equal to the state of the bit s. The operand Rs may specify any active CPU register. The value of p may be from 0 to 7, where 0 corresponds to the LSB of Rs and 7 corresponds to the MSB of Rs. The absolute value nn is 16 bits long, (range: 0 to 64k), therefore, all of instruction memory can be addressed.

Example

Long Jump to one of the receiver error handling routines based on the contents of the Error Code Register {ECR}.

```
   EXX     0,1,3              ;select main A, alt B
                              ; and clear [GIE]
   OR      01000000B,TSR      ;set [SEC] in {TSR}
   MOVE    ECR, R11           ;read {ECR}
; Determine error condition
   LJMP    R11, 0, 1, Software__error
   LJMP    R11, 1, 1, Loss__of__Midbit
   LJMP    R11, 2, 1, Invalid__Ending__Seq
   LJMP    R11, 3, 1, Parity__error
   LJMP    R11, 4, 1, Software__error
```

Instruction Format



T-states

(2 + 2)

Bus Timing

Figure 6-3

Operation

If Rs[p] = s
then nn → PC

LJMP Unconditional Long Jump

Syntax

LJMP nn —absolute
LJMP [Ir] —indexed

Affected Flags

None

Description

Unconditionally transfers control to the instruction at the memory address specified by the operand. The operand may either specify an absolute instruction address nn, (16 bits long), or an index register Ir, which contains an instruction address. Long Jump's addressing range is from 0 to 64k; (i.e., all of instruction memory can be addressed).

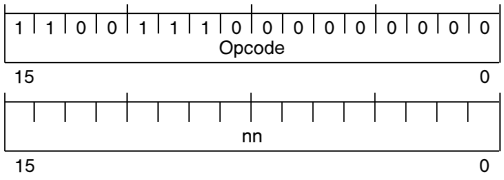
Example

Transfer control to the instruction labeled ''Reset.System'', which may be located anywhere in instruction memory.

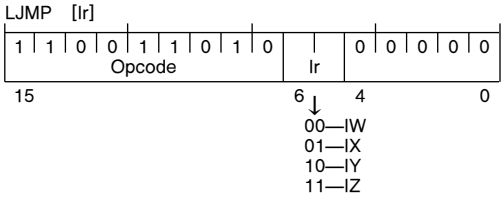
```
   LJMP    Reset.System       ;go reset the system
```

Instruction Format

LJMP nn



LJMP [Ir]



T-states

LJMP nn —(2 + 2)
LJMP [Ir] —2

Bus Timing

LJMP nn —*Figure 6-3*
LJMP [Ir] —*Figure 6-1*

Operation

LJMP nn
nn → PC

LJMP [Ir]
Ir → PC

MOVE Move Data Memory

Syntax

```
MOVE   [mIr], Rd      —indexed, register
MOVE   [Ir+A], Rd     —register-relative, register
MOVE   [IZ+n], rd     —immediate-relative, limited register
```

Affected Flags

None

Description

Moves a data memory byte into the destination register specified. The data memory source operand may specify any one of the index register modes; [mIr], [Ir+A], [IZ+n]. The index register-relative mode, [Ir+A], forms its data memory address by adding the contents of the index register Ir to the unsigned 8-bit value contained in the currently active accumulator. The immediate-relative mode, [IZ+n], forms its data memory address by adding the contents of the index register IZ to the unsigned 8-bit immediate value n. The destination register operand Rd may specify any active CPU register; where as, the destination register operand rd is limited to the active registers R0–R15.

Example

The first example loads the current accumulator by ''popping'' an external data stack, which is pointed to by the index register IX.

```
MOVE     [+IX], A        ;pop accum from ext. stack
```

The second example demonstrates the random access of a data byte within a logical record contained in memory. The index register IY contains the base address of the logical record.

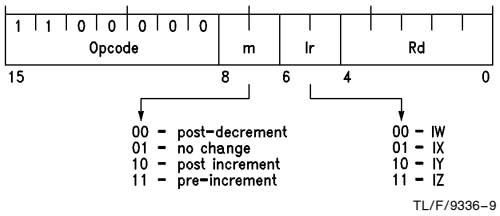
```
ADDA     R9, A           ;calculate offset into record
MOVE     [IY+A], R20     ;get data byte from record
```

In the final example, the 4th element of an Error Count table is transmitted to a host. The index register IZ points to the 1st entry of the table.

```
EXX      0,1             ;select main A, alt B
MOVE     [IZ+3], RTR     ;transmit 4th element
```

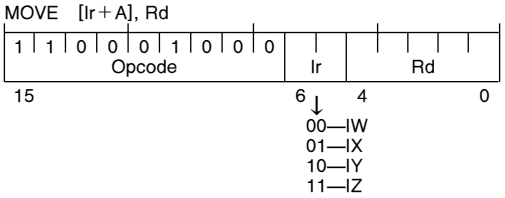
Instruction Format

MOVE [mIr], Rd

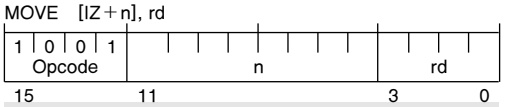


TL/F/9336–9

MOVE [Ir+A], Rd



MOVE [IZ+n], rd



T-states

3 [4TR] = 0
4 [4TR] = 1

Bus Timing

Figure 6-5 [4TR] = 0
Figure 6-6 [4TR] = 1

Operation

MOVE [mIr], Rd
data memory → Rd

MOVE [Ir+A], Rd
data memory → Rd

MOVE [IZ+n], rd
data memory → rd

MOVE Move Immediate

Syntax

MOVE n, rd —immediate, limited register

MOVE n, [Ir] —immediate, indexed

Affected Flags

None

Description

Moves the immediate value n into the destination specified. The destination may be either a register, rd, (limited to the active registers R0–R15), or data memory via an index register, Ir. The value n is 8 bits wide.

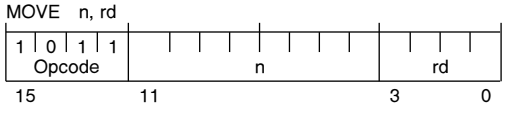
Example

Load the current accumulator with the value of 4.

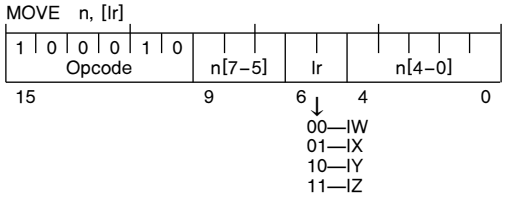
```
MOVE   4, A          ;Load accumulator
```

Instruction Format

MOVE n, rd



MOVE n, [Ir]



T-states

MOVE n, rd —2

MOVE n, [Ir] —3

Bus Timing

MOVE n, rd —*Figure 6-1*

MOVE n, [Ir] —*Figure 6-7*

Operation

MOVE n, rd

n ⟶ rd

MOVE n, [Ir]

n ⟶ data memory

MOVE Move Register

Syntax

MOVE Rs, Rd —register, register
MOVE Rs, [mIr] —register, indexed
MOVE Rs, [Ir+A] —register, register-relative
MOVE rs, [IZ+n] —limited register, immediate-relative

Affected Flags

None

Description

Moves the contents of the source register into the destination specified. The source register operand Rs may specify any active CPU register; where as the source register operand rs is limited to the active registers R0–R15. The destination operand may specify either any active CPU register, Rd, or data memory via one of the index register modes; [mIr], [Ir+A], [IZ+n]. The index register-relative mode, [Ir+A], forms its data memory address by adding the contents of the index register Ir to the unsigned 8-bit value contained in the currently active accumulator. The immediate-relative mode, [IZ+n], forms its data memory address by adding the contents of the index register IZ to the unsigned 8-bit immediate value n.

Example

The first example loads the Transmitter FIFO with a data byte in register 20.

```
EXX     0,1          ;select main A, alt B
MOVE    R20, RTR     ;Load the Transmitter FIFO
```

The second example ''pushes'' the current accumulator's contents onto an external data stack, which is pointed to by the index register IX.

```
MOVE    A, [IX−]     ;push accum to ext. stack
```

The third example demonstrates the random access of a data byte within a logical record contained in memory. The index register IY contains the base address of the logical record.

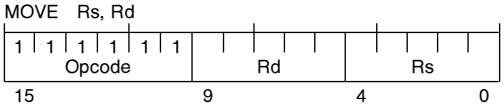
```
ADDA    R9, A        ;calculate offset into record
MOVE    R20, [IY+A]  ;update data byte in record
```

In the final example, the 4th element of an Error Count table is updated with a new value contained in the current accumulator. The index register IZ points to the 1st entry of the table.

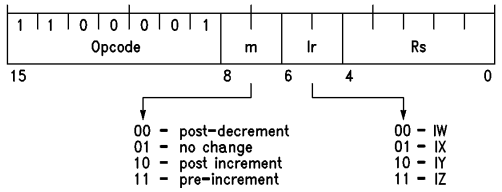
```
MOVE    A, [IZ+3]    ;update 4th element of table
```

Instruction Format

MOVE Rs, Rd

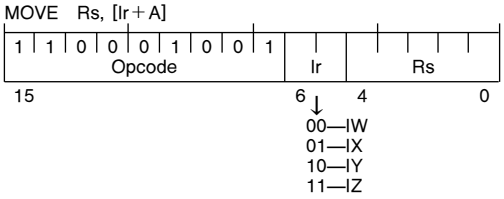


MOVE Rs, [mIr]

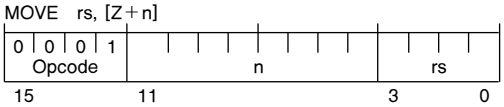


TL/F/9336–10

MOVE Rs, [Ir+A]



MOVE rs, [Z+n]



T-states

MOVE Rs, Rd —2
MOVE Rs, [mIr] —3
MOVE Rs, [Ir+A] —3
MOVE rs, [IZ+n] —3

Bus Timing

MOVE Rs, Rd —*Figure 6-1*
MOVE Rs, [mIr] —*Figure 6-6*
MOVE Rs, [Ir+A] —*Figure 6-6*
MOVE rs, [IZ+n] —*Figure 6-6*

Operation

MOVE Rs, Rd —Rs ⟶ Rd
MOVE Rs, [mIr] —Rs ⟶ data memory
MOVE Rs, [Ir+A] —Rs ⟶ data memory
MOVE rs, [IZ+n] —rs ⟶ data memory

OR OR Immediate

Syntax

OR n, rsd —immediate, limited register

Affected Flags

N, Z

Description

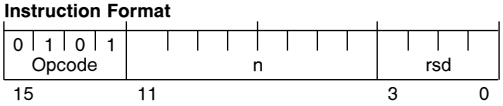
Logically ORs the immediate value n to the register rsd and places the result back into the register rsd. Note that only the active registers R0–R15 may be specified for rsd. The value of n is 8 bits wide.

Example

Mask both the Transmitter and Receiver interrupts via the Interrupt Control Register {ICR}, R2. Leave the other interrupts unaffected.

```
EXX   0,0               ;select main reg banks
OR    00000011B, ICR    ;mask transmitter and
                        ; receiver interrupts
```

Instruction Format



T-states

2

Bus Timing

Figure 6-1

Operation

rsd OR n ⟶ rsd

ORA OR with Accumulator

Syntax

ORA Rs, Rd —register, register

ORA Rs, [mlr] —register, indexed

Affected Flags

N, Z

Description

Logically ORs the source register Rs to the active accumulator and places the result into the destination specified. The destination may be either a register, Rd, or data memory via an index register mode, [mlr]. Note that register bank selection determines which accumulator is active.

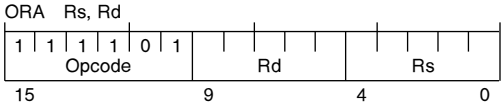
Example

Write an 11-bit word to the Transmitter's FIFO. This example assumes that the index register IZ points to the location of the data in memory.

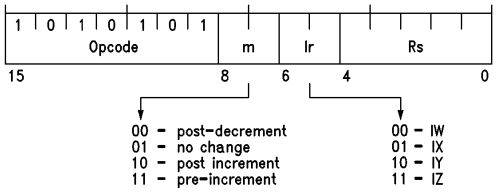
```
TCR.settings:   .EQU   00101000B
   . . .
EXX     1,1             ;select main A, alt B
MOVE    TCR.settings,A  ;load accumulator w/mask
MOVE    [IZ+],R20       ;load bits 8, 9, & 10
ORA     R20,TCR         ;write bits 8, 9, 10 to {TCR}
MOVE    [IZ+],RTR       ;push 11-bit word to FIFO
```

Instruction Format

ORA Rs, Rd



ORA Rs, [mlr]



TL/F/9336–11

T-states

ORA Rs, Rd —2

ORA Rs, [mlr] —3

Bus Timing

ORA Rs, Rd —*Figure 6-1*

ORA Rs, [mlr] —*Figure 6-7*

Operation

ORA Rs, Rd

Rs OR accumulator ⟶ Rd

ORA Rs, [mlr]

Rs OR accumulator ⟶ data memory

RETF Conditional Return

Rcc

Syntax

RETF f, s{,{g} {,rf}}
Rcc {g{,rf}} —(optional syntax)

Affected Flags

If rf = 1 then N, Z, C, and V

Description

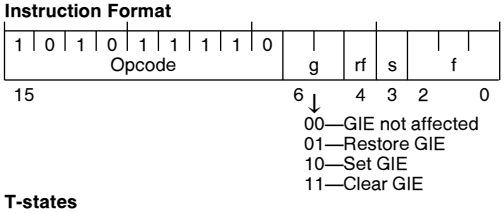
Conditionally returns control to the last instruction address pushed onto the internal Address Stack by popping that address into the Program Counter, if the state of the flag referenced by f is equal to the state of the bit s; or, optionally, if the condition cc is met. See the tables on the following page for the flags that f can reference and the conditions that cc may specify. The conditional return instruction also has two optional operands, g and rf. The value of g determines if the Global Interrupt Enable bit [GIE] is left unchanged (g=0), restored from the Address Stack (g=1), set (g=2), or cleared (g=3). If the g operand is omitted then g=0 is assumed. The second optional operand, rf, determines if the ALU flags and register bank selections are left unchanged (rf=0), or restored from the Address Stack (rf=1). If the rf operand is omitted then rf=0 is assumed.

Example

This example demonstrates both syntaxes of the conditional return instruction testing for a carry result from a previous instruction; (i.e., [C]=1). If the condition is met then the return occurs, else the next instruction following the return is executed. The current environment is left unchanged.

```
RETF   001B,1        ; If [C]=1 then return
 . . .
RC                   ; If [C]=1 then return
```

Instruction Format



T-states

2 if condition is not met

3 if condition is met

Bus Timing

Figure 6-1 if condition is not met
Figure 6-2 if condition is met

Operation

```
If flag f is in state s then
  Case g of
    0: leave [GIE] unaffected, (default)
    1: restore [GIE] from Address Stack
    2: set [GIE]
    3: clear [GIE]
  End case
  If rf=1 then
    restore ALU flags from Address Stack
    restore register bank selection from Address Stack
  End if
  Address Stack ⟶ PC
End if
```

Condition Specification Table for "cc"

cc	Meaning	Condition Tested for
Z	Zero	[Z] = 1
NZ	Not Zero	[Z] = 0
EQ	Equal	[Z] = 1
NEQ	Not Equal	[Z] = 0
C	Carry	[C] = 1
NC	No Carry	[C] = 0
V	Overflow	[V] = 1
NV	No Overflow	[V] = 0
N	Negative	[N] = 1
P	Positive	[N] = 0
RA	Receiver Active	[RA] = 1
NRA	Not Receiver Active	[RA] = 0
RE	Receiver Error	[RE] = 1
NRE	No Receiver Error	[RE] = 0
DA	Data Available	[DAV] = 1
NDA	No Data Available	[DAV] = 0
TFF	Transmitter FIFO Full	[TFF] = 1
NTFF	Transmitter FIFO Not Full	[TFF] = 0

Flag Reference Table for "f"

f	(binary)	Flag Referenced
0	(000)	[Z] in {CCR}
1	(001)	[C] in {CCR}
2	(010)	[V] in {CCR}
3	(011)	[N] in {CCR}
4	(100)	[RA] in {TSR}
5	(101)	[RE] in {TSR}
6*	(110)	[DAV] in {TSR}
7	(111)	[TFF] in {TSR}

*__Note:__ The value of f for [DAV] differs from the numeric value for the position of [DAV] in {TSR}.

RET Unconditional Return

Syntax

RET {g {,rf}}

Affected Flags

If rf = 1 then N, Z, C, and V

Description

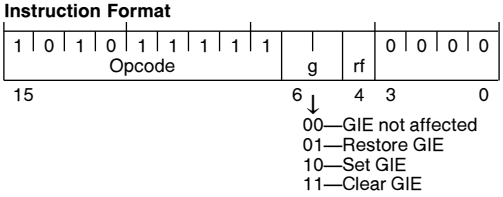
Unconditionally returns control to the last instruction address pushed onto the internal Address Stack by popping that address into the Program Counter. The unconditional return instruction also has two optional operands, g and rf. The value of g determines if the Global Interrupt Enable bit [GIE] is left unchanged (g = 0), restored from the Address Stack (g = 1), set (g = 2), or cleared (g = 3). If the g operand is omitted then g = 0 is assumed. The second optional operand, rf, determines if the ALU flags and register bank selections are left unchanged (rf = 0), or restored from the Address Stack (rf = 1). If the rf operand is omitted then rf = 0 is assumed.

Example

Return from an interrupt.

```
RET     1,1          ;Restore environment & return
```

Instruction Format



T-states

2

Bus Timing

Figure 6-1

Operation

```
Case g of
  0: leave [GIE] unaffected, (default)
  1: restore [GIE] from Address Stack
  2: set [GIE]
  3: clear [GIE]
End case
If rf = 1 then
  restore ALU flags from Address Stack
  restore register bank selection from Address Stack
End if
Address Stack ⟶ PC
```

ROT Rotate

Syntax

ROT Rsd, b —register

Affected Flags

N, Z, C

Description

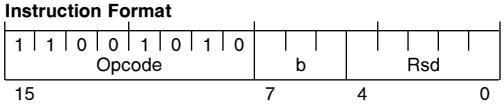
Rotates the contents of the register Rsd b bits to the right and places the result back into that register. The bits that are shifted out of the LSB are shifted back into the MSB, (and copied into the Carry flag). The value b may specify from 0 to 7 bit rotates.

Example

Add 3 to the Address Stack Pointer contained in the Internal Stack Pointer register {ISP}, R30.

```
MOVE    ISP, R8      ;get {ISP}
ROT     R8, 4        ;shift [ASP] to low order nibble
ADD     3, R8        ;add 3 to [ASP]
ROT     R8, 4        ;shift [ASP] to high order nibble
MOVE    R8, ISP      ;store new {ISP}
```

Instruction Format



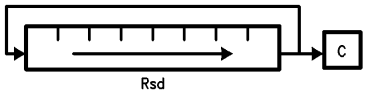
T-states

2

Bus Timing

Figure 6-1

Operation



TL/F9336–12

SBCA Subtract with Carry and Accumulator

Syntax

SBCA Rs, Rd —register, register
SBCA Rs, [mlr] —register, indexed

Affected Flags

N, Z, C, V

Description

Subtracts the active accumulator and the carry flag from the source register Rs, placing the result into the destination specified. The destination may be either a register, Rd, or data memory via an index register mode, [mlr]. Negative results are represented using the two's complement format. Note that register bank selection determines which accumulator is active.

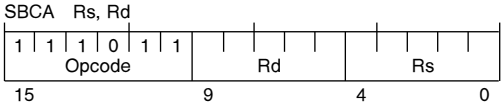
Example

Subtract the constant 109 from the index register IW, (which is 16 bits wide).

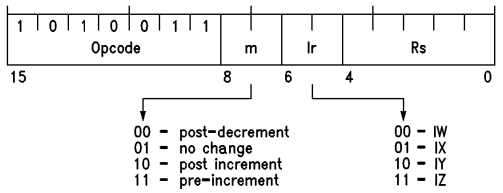
```
SUBA   A, A        ;Clear the accumulator
SUB    109, R12    ;low byte of IW—109
SBCA   R13, R13    ;high byte of IW—borrow
```

Instruction Format

SBCA Rs, Rd



SBCA Rs, [mlr]



TL/F9336–13

T-states

SBCA Rs, Rd —2
SBCA Rs, [mlr] —3

Bus Timing

SBCA Rs, Rd —*Figure 6-1*
SBCA Rs, [mlr] —*Figure 6-7*

Operation

SBCA Rs, Rd
Rs − accumulator − carry bit ⟶ Rd

SBCA Rs, [mlr]
Rs − accumulator − carry bit ⟶ data memory

SHL Shift Left

Syntax

SHL Rsd, b —register

Affected Flags

N, Z, C

Description

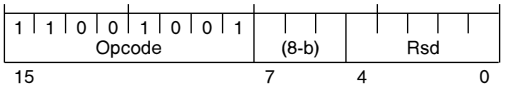
Shifts the contents of the register Rsd b bits to the left and places the result back into that register. Zeros are shifted in from the right, (i.e., from the LSB). The value b may specify from 0 to 7 bit shifts. The Carry flag contains the last bit shifted out.

Example

Place a new internal Address Stack Pointer into the Internal Stack Pointer register {ISP}, R30. Assume that the new [ASP] is located in register 20.

```
MOVE   ISP,R8          ;read {ISP} for [DSP]
AND    00001111B,R8    ;save [DSP] only
SHL    R20,4           ;left justify [ASP]
ORA    R20,ISP         ;combine [ASP] + [DSP],
                       ; then place into {ISP}
```

Instruction Format



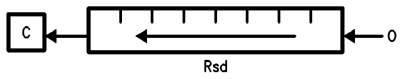
T-states

2

Bus Timing

Figure 6-1

Operation



TL/F/9336–14

SHR Shift Right

Syntax

SHR Rsd, b —register

Affected Flags

N, Z, C

Description

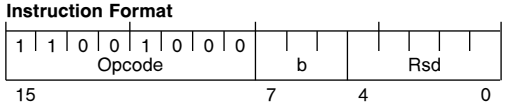
Shifts the contents of the register Rsd b bits to the right and places the result back into that register. Zeros are shifted in from the left, (i.e., from the MSB). The value b may specify from 0 to 7 bit shifts. The Carry flag contains the last bit shifted out.

Example

Right justify the Address Stack Pointer from the Internal Stack Pointer register {ISP}, R30.

```
MOVE   ISP, R20      ;Load [ASP] from {ISP}
SHR    R20,4         ;right justify [ASP]
```

Instruction Format



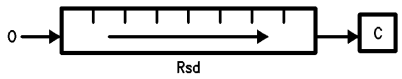
T-states

2

Bus Timing

Figure 6-1

Operation



TL/F/9336–15

SUB Subtract Immediate

Syntax

SUB n, rsd —immediate, limited register

Affected Flags

N, Z, C, V

Description

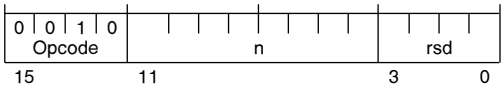
Subtracts the immediate value n from the register rsd and places the result back into the register rsd. Note that only the active registers R0–R15 may be specified for rsd. The value of n is limited to 8 bits; (signed range: +127 to −128). Negative numbers are represented using the two's complement format.

Example

Subtract the constant 3 from register 10.

```
SUB   3, R10      ; R10 − 3 ⟶ R10
```

Instruction Format



T-states

2

Bus Timing

Figure 6-1

Operation

rsd − n ⟶ rsd

SUBA Subtract with Accumulator

Syntax

SUBA Rs, Rd —register, register
SUBA Rs, [mlr] —register, indexed

Affected Flags

N, Z, C, V

Description

Subtracts the active accumulator from the source register Rs and places the result into the destination specified. The destination may be either a register, Rd, or data memory via an index register mode, [mlr]. Negative numbers are represented using the two's complement format. Note that register bank selection determines which accumulator is active.

Example

In the first example, the value 4 is placed into the currently active accumulator, that accumulator is subtracted from the contents of register 20, and then the result is placed into register 21.

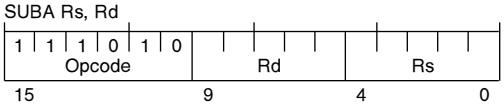
```
MOVE   4, A          ;Place constant into accum
SUBA   R20, R21      ;R20 − accum ⟶ R21
```

In the second example, the alternate accumulator of register bank B is selected and then subtracted from register 20. The result is placed into the data memory pointed to by the index register IZ and then the value of IZ is incremented by one.

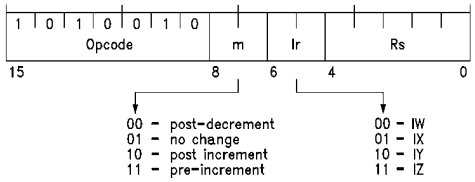
```
EXX    0, 1          ;Select alt accumulator
SUBA   R20, [IZ+]    ;R20 − accum ⟶ data mem
                     ;and increment data pointer
```

Instruction Format

SUBA Rs, Rd



SUBA Rs, [mlr]



TL/F/9336–16

T-states

SUBA Rs, Rd —2
SUBA Rs, [mlr] —3

Bus Timing

SUBA Rs, Rd —*Figure 6-1*
SUBA Rs, [mlr] —*Figure 6-7*

Operation

SUBA Rs, Rd
Rs − accumulator ⟶ Rd

SUBA Rs, [mlr]
Rs − accumulator ⟶ data memory

TRAP Software Interrupt

Syntax

TRAP v {,g′}

Affected Flags

None

Description

Pushes the Program Counter, the Global Interrupt Enable bit [GIE], the ALU flags, and the current register bank selections onto the internal Address Stack; then unconditionally transfers control to the instruction at the memory address created by concatenating the contents of the Interrupt Base Register {IBR} to the value of v extended with zeros to 8 bits. If the value of g′ is equal to "1" then the Global Interrupt Enable bit [GIE] will be cleared. If the g′ operand is omitted, then g′ = 0 is assumed. The vector number v points to one of 64 Interrupt Table entries; (range: 0 to 63). Since some of the Interrupt Table entries are used by the hardware interrupts, the TRAP instruction can simulate hardware interrupts. The following table lists the hardware interrupts and their associated vector numbers:

Hardware Interrupt Vector Table

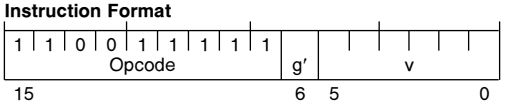
Interrupt	v	(Binary)
NMI	28	(011100)
RFF/DA/RA	4	(000100)
TFE	8	(001000)
LTA	12	(001100)
BIRQ	16	(010000)
TO	20	(010100)

Example

Simulate the Transmitter FIFO Empty interrupt.

```
TRAP     8, 1      ;TFE interrupt simulation
```

Instruction Format



T-states

2

Bus Timing

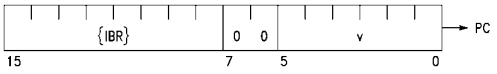
Figure 6-1

Operation

PC & [GIE] & ALU flags & register bank selections
⟶ Address Stack

if g′ = 1
then clear [GIE]

Create PC address by concatonating the {IBR} register to the vector number v as shown below:



TL/F/9336–17

XOR Exclusive OR Immediate

Syntax
XOR n, rsd —immediate, limited register

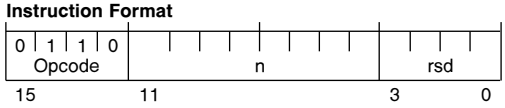
Affected Flags
N, Z

Description
Logically exclusive ORs the immediate value n to the register rsd and places the result back into the register rsd. Note that only the active registers R0–R15 may be specified for rsd. The value of n is 8 bits wide.

Example
Encode/decode a data byte in register 15.

```
XOR   code_pattern, R15     ;encode/decode
```

Instruction Format



T-states
2

Bus Timing
Figure 6-1

Operation
rsd XOR n ⟶ rsd

XORA Exclusive OR with Accumulator

Syntax
XORA Rs, Rd —register, register
XORA Rs, [mlr] —register, indexed

Affected Flags
N, Z

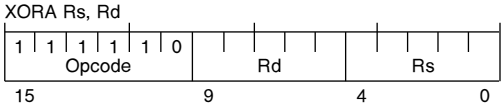
Description
Logically exclusive ORs the source register Rs to the active accumulator and places the result into the destination specified. The destination may be either a register, Rd, or data memory via an index register mode, [mlr]. Note that register bank selection determines which accumulator is active.

Example
Decode the data byte just received and place it into data memory. This example assumes that the accumulator contains the ''key'' and that the index register IY points to the location where the information should be stored.

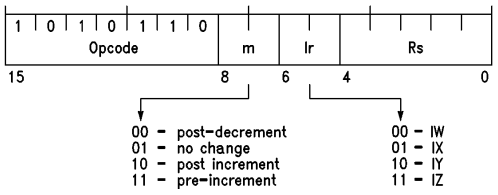
```
EXX     1,1             ;select alternate banks
XORA    RTR, [IY+]      ;decode received byte and
                        ; save it
```

Instruction Format

XORA Rs, Rd



XORA Rs, [mlr]



TL/F/9336–18

T-states
XORA Rs, Rd —2
XORA Rs, [mlr] —3

Bus Timing
XORA Rs, Rd —*Figure 6-1*
XORA Rs, [mlr] —*Figure 6-7*

Operation
XORA Rs, Rd
Rs XOR accumulator ⟶ Rd

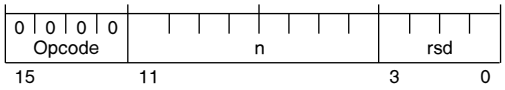
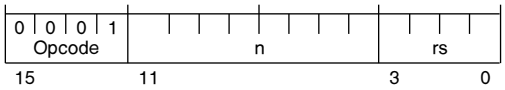
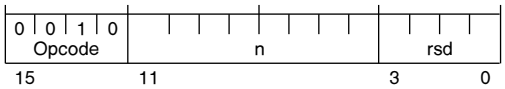
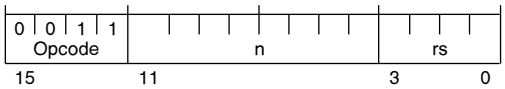
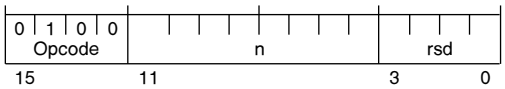
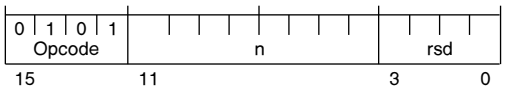
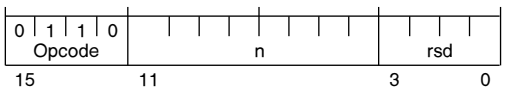
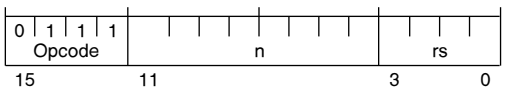
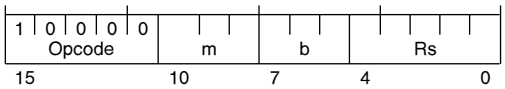
XORA Rs, [mlr]
Rs XOR accumulator ⟶ data memory

TABLE 6-2. Instructions Versus T-states, Affected Flags, and Bus Timing

Instruction		T-states	Affected Flags	Timing Figure
ADCA	Rs, Rd	2	N,Z,C,V	6-1
ADCA	Rs, [mlr]	3	N,Z,C,V	6-7
ADD	n, rsd	2	N,Z,C,V	6-1
ADDA	Rs, Rd	2	N,Z,C,V	6-1
ADDA	Rs, [mlr]	3	N,Z,C,V	6-7
AND	n, rsd	2	N,Z	6-1
ANDA	Rs, Rd	2	N,Z	6-1
ANDA	Rs, [mlr]	3	N,Z	6-7
BIT	rs, n	2	N,Z	6-1
CALL	n	3		6-2
CMP	rs, n	2	N,Z,C,V	6-1
CPL	Rsd	2	N,Z	6-1
EXX	ba, bb {,g}	2		6-1
Jcc	n	2 false 3 true		6-1 6-2
JMP	f, s, n	2 false 3 true		6-1 6-2
JMP	n	3		6-2
JMP	Rs	4		6-4
JRMK	Rs, b, m	4		6-4
LCALL	nn	(2+2)		6-3
LCALL	Rs, p, s, nn	(2+2)		6-3
LJMP	nn	(2+2)		6-3
LJMP	[lr]	2		6-1
LJMP	Rs, p, s, nn	(2+2)		6-3
MOVE	n, rd	2		6-1
MOVE	n, [lr]	3		6-7
MOVE	Rs, Rd	2		6-1
MOVE	Rs, [mlr]	3		6-7
MOVE	Rs, [lr + A]	3		6-7
MOVE	rs, [lZ + n]	3		6-7
MOVE	[mlr], Rd	3 [4TR] = 0 4 [4TR] = 1		6-5 6-6
MOVE	[lr + A], Rd	3 [4TR] = 0 4 [4TR] = 1		6-5 6-6
MOVE	[lZ + n], rd	3 [4TR] = 0 4 [4TR] = 1		6-5 6-6
OR	n, rsd	2	N,Z	6-1
ORA	Rs, Rd	2	N,Z	6-7
ORA	Rs, [mlr]	3	N,Z	6-7
Rcc	{g{,rf}}	2 false 3 true	 N,Z,C,V*	6-1 6-2
RET	{g{,rf}}	2	N,Z,C,V*	6-1
RETF	f, s {,{g} {,rf}}	2 false 3 true	 N,Z,C,V*	6-1 6-2
ROT	Rsd, b	2	N,Z,C	6-1
SBCA	Rs, Rd	2	N,Z,C,V	6-1
SBCA	Rs, [mlr]	3	N,Z,C,V	6-7
SHL	Rsd, b	2	N,Z,C	6-1
SHR	Rsd, b	2	N,Z,C	6-1
SUB	n, rsd	2	N,Z,C,V	6-1
SUBA	Rs, Rd	2	N,Z,C,V	6-1
SUBA	Rs, [mlr]	3	N,Z,C,V	6-7
TRAP	v {,g'}	2		6-1
XOR	n, rsd	2	N,Z	6-1
XORA	Rs, Rd	2	N,Z	6-1
XORA	Rs, [mlr]	3	N,Z	6-7

*__Note:__ If rf = 1 then N, Z, C, and V are affected.

TABLE 6-3. Instruction Opcodes

Hex	Opcode	Instruction
0000–0FFF		ADD n, rsd
1000–1FFF		MOVE rs, [IZ + n]
2000–2FFF		SUB n, rsd
3000–3FFF		CMP rs, n
4000–4FFF		AND n, rsd
5000–5FFF		OR n, rsd
6000–6FFF		XOR n, rsd
7000–7FFF		BIT rs, n
8000–87FF		JRMK Rs, b, m

KEY

mlr

00	lr−
01	lr
10	lr+
11	+lr

lr

00	IW
01	IX
10	IY
11	IZ

g

00	NCHG
01	RI
10	EI
11	DI

g′

0	NCHG
1	DI

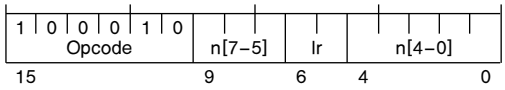
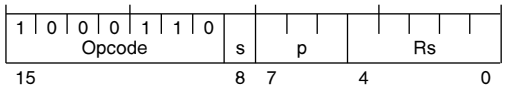
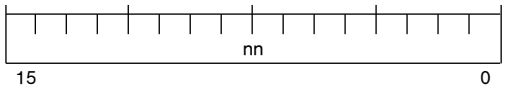
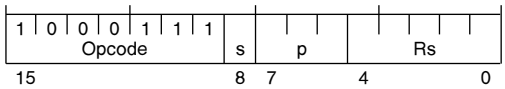
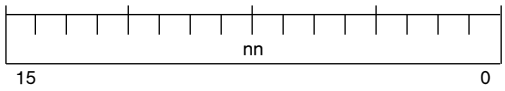
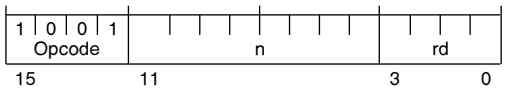
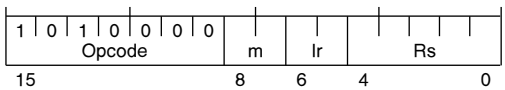
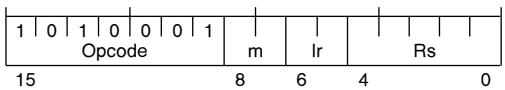
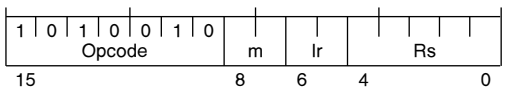
ba/bb

0	MAIN
1	ALT

f

000	[Z]
001	[C]
010	[V]
011	[N]
100	[RA]
101	[RE]
110	[DAV]
111	[TFF]

TABLE 6-3. Instruction Opcodes (Continued)

Hex	Opcode	Instruction
8800–8BFF		MOVE n, [lr]
8C00–8DFF		LJMP Rs, p, s, nn
0000–FFFF		
8E00–8FFF		LCALL Rs, p, s, nn
0000–FFFF		
9000–9FFF		MOVE [IZ+n], rd
A000–A1FF		ADDA Rs, [mlr]
A200–A3FF		ADCA Rs, [mlr]
A400–A5FF		SUBA Rs, [mlr]

KEY

mlr

00	lr–
01	lr
10	lr+
11	+lr

lr

00	IW
01	IX
10	IY
11	IZ

g

00	NCHG
01	RI
10	EI
11	DI

g′

0	NCHG
1	DI

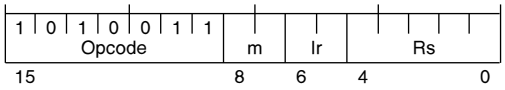
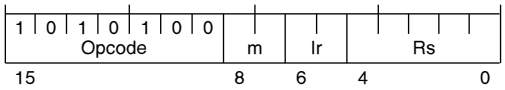
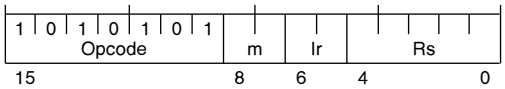
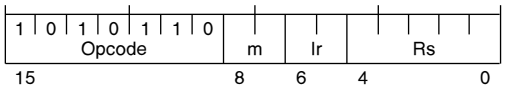
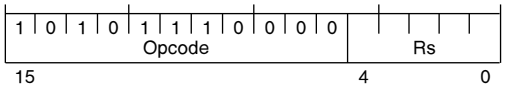
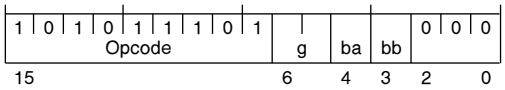
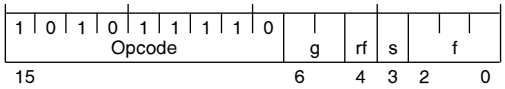
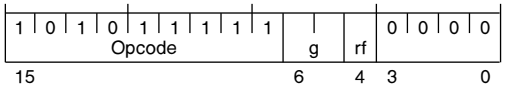
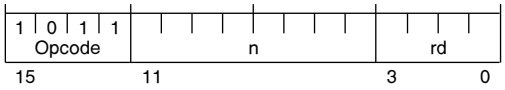
ba/bb

0	MAIN
1	ALT

f

000	[Z]
001	[C]
010	[V]
011	[N]
100	[RA]
101	[RE]
110	[DAV]
111	[TFF]

TABLE 6-3. Instruction Opcodes (Continued)

Hex	Opcode	Instruction
A600–A7FF		SBCA Rs, [mlr]
A800–A9FF		ANDA Rs, [mlr]
AA00–ABFF		ORA Rs, [mlr]
AC00–ADFF		XORA Rs, [mlr]
AE00–AE1F		CPL Rsd
AE80–AEF8		EXX ba, bb {,g}
AF00–AF7F		RETF f,s{,{g}{,rf}} Rcc {g{,rf}}
AF80–AFF0		RET {g{,rf}}
B000–BFFF		MOVE n, rd

KEY

mlr

00	lr−
01	lr
10	lr+
11	+lr

lr

00	IW
01	IX
10	IY
11	IZ

g

00	NCHG
01	RI
10	EI
11	DI

g′

0	NCHG
1	DI

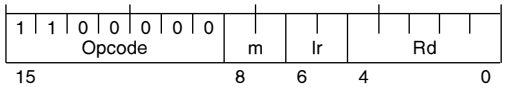
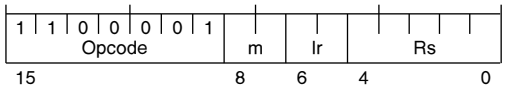
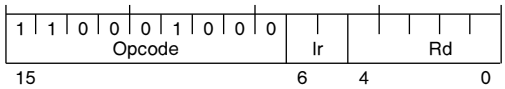
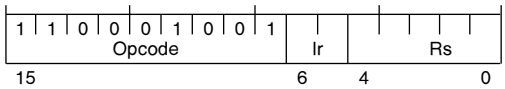
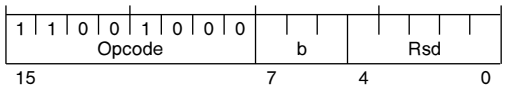
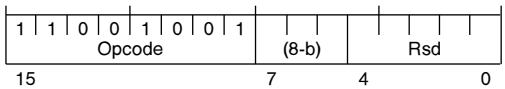
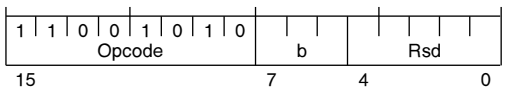
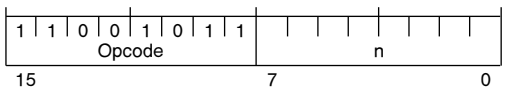
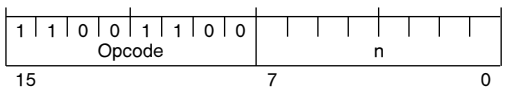
ba/bb

0	MAIN
1	ALT

f

000	[Z]
001	[C]
010	[V]
011	[N]
100	[RA]
101	[RE]
110	[DAV]
111	[TFF]

TABLE 6-3. Instruction Opcodes (Continued)

Hex	Opcode	Instruction
C000–C1FF		MOVE [mIr], Rd
C200–C3FF		MOVE Rs, [mIr]
C400–C47F		MOVE [Ir+A], Rd
C480–C4FF		MOVE Rs, [Ir+A]
C800–C8FF		SHR Rsd, b
C900–C9FF		SHL Rsd, b
CA00–CAFF		ROT Rsd, b
CB00–CBFF		JMP n
CC00–CCFF		CALL n

KEY

mIr

00	Ir−
01	Ir
10	Ir+
11	+Ir

Ir

00	IW
01	IX
10	IY
11	IZ

g

00	NCHG
01	RI
10	EI
11	DI

g′

0	NCHG
1	DI

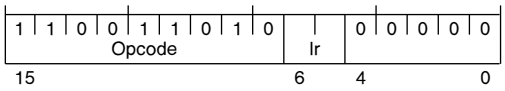
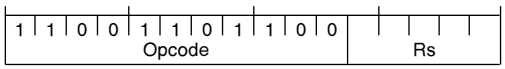
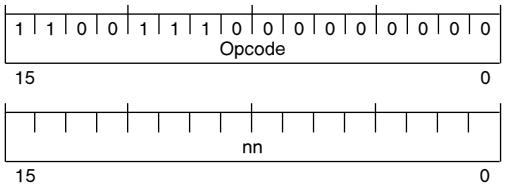
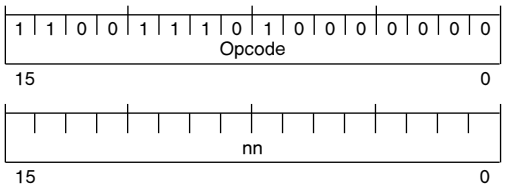
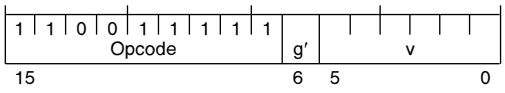
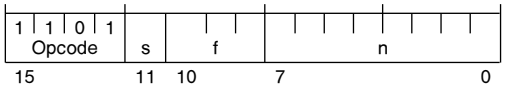
ba/bb

0	MAIN
1	ALT

f

000	[Z]
001	[C]
010	[V]
011	[N]
100	[RA]
101	[RE]
110	[DAV]
111	[TFF]

TABLE 6-3. Instruction Opcodes (Continued)

Hex	Opcode	Instruction
CD00–CD60		LJMP [lr]
CD80–CD9F		JMP Rs
CE00 0000–FFFF		LJMP nn
CE80 0000–FFFF		LCALL nn
CF80–CFFF		TRAP v{,g′}
D000–DFFF		JMP f, s, n Jcc n

KEY

mlr

00	lr−
01	lr
10	lr+
11	+lr

lr

00	IW
01	IX
10	IY
11	IZ

g

00	NCHG
01	RI
10	EI
11	DI

g′

0	NCHG
1	DI

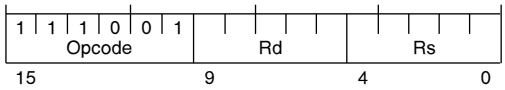
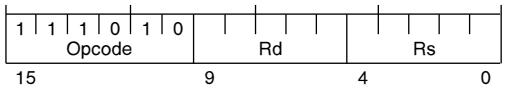
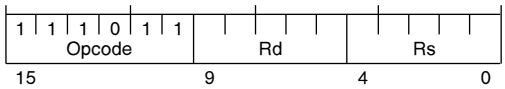
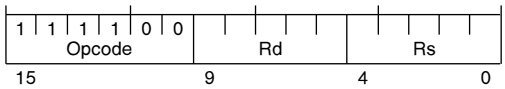
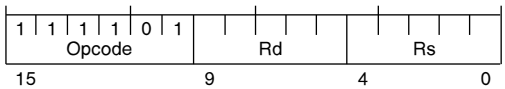
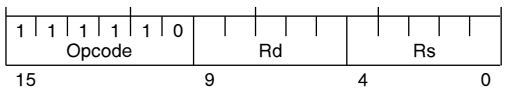
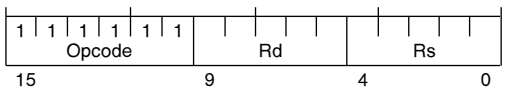
ba/bb

0	MAIN
1	ALT

f

000	[Z]
001	[C]
010	[V]
011	[N]
100	[RA]
101	[RE]
110	[DAV]
111	[TFF]

TABLE 6-3. Instruction Opcodes (Continued)

Hex	Opcode	Instruction
E000–E3FF		ADDA Rs, Rd
E400–E7FF		ADCA Rs, Rd
E800–EBFF		SUBA Rs, Rd
EC00–EFFF		SBCA Rs, Rd
F000–F3FF		ANDA Rs, Rd
F400–F7FF		ORA Rs, Rd
F800–FBFF		XORA Rs, Rd
FC00–FFFF		MOVE Rs, Rd

KEY

mlr

00	lr−
01	lr
10	lr+
11	+lr

lr

00	IW
01	IX
10	IY
11	IZ

g

00	NCHG
01	RI
10	EI
11	DI

g′

0	NCHG
1	DI

ba/bb

0	MAIN
1	ALT

f

000	[Z]
001	[C]
010	[V]
011	[N]
100	[RA]
101	[RE]
110	[DAV]
111	[TFF]

6.0 Reference Section (Continued)

6.2 REGISTER SET REFERENCE

The register set reference contains detailed information on the bit definitions of all special function registers that are addressable in the CPU. This reference section presents the information in three forms: a bit index, a register description and bit definition tables. The bit index is an alphabetical listing of all status/control bits in the CPU-addressable function registers, with a brief summary of the function. The register description is a list of all CPU-addressable special function registers in alphabetical order. The bit definition tables describe the location and function of all control and status bits in the various CPU-addressable special function registers. These tables are arranged by function.

6.2.1 Bit Index

An alphabetical listing of all status/control bits in the CPU-addressable special function registers, with a brief summary of function. Detailed definitions are provided in Section 6.2.3, Bit Definition Tables.

Bit	Name	Location	Function
4TR	Four T-State Read	ACR [3]	Timing Control
ACK	poll/**ACK**nowledge	NCF [1]	Receiver Status
ASP3–0	**A**ddress **S**tack **P**ointer	ISP [7–4]	Stacks
AT7–0	**A**uxilliary **T**ransceiver control	ATR [7–0]	Receiver Control
ATA	**A**dvance **T**ransmitter **A**ctive	TCR [4]	Transmitter Control
BIC	**B**i-directional **I**nterrupt **C**ontrol	ACR [4]	Interrupt Control
BIRQ	**B**i-directional **I**nterrupt **ReQ**uest	CCR [4]	Interrupt Control
C	**C**arry	CCR [1]	Arithmetic Flag
CCS	**C**PU **C**lock **S**elect	DCR [7]	Timing Control
COD	**C**lock **O**ut **D**isable	ACR [2]	Timing Control
DAV	**D**ata **AV**ailable	TSR [3]	Receiver Status
DEME	**D**ata **E**rror or **M**essage **E**nd	NCF [3]	Receiver Status
DS7–0	**D**ata **S**tack	DS [7–0]	Stacks
DSP3–0	**D**ata **S**tack **P**ointer	ISP [3–0]	Stacks
DW2–0	**D**ata memory **W**ait-state select	DCR [2–0]	Timing Control
FB7–0	**F**ill **B**its	FBR [7–0]	Transmitter Control
GIE	**G**lobal **I**nterrupt **E**nable	ACR [0]	Interrupt Control
IES	**I**nvalid **E**nding **S**equence	ECR [2]	Receiver Error Code
IM4–0	**I**nterrupt **M**ask select	ICR [4–0]	Interrupt Control
IV15–8	**I**nterrupt **V**ector	IBR [7–0]	Interrupt Control
IW1,0	**I**nstruction memory **W**ait-state select	DCR [4,3]	Timing Control
LA	**L**ine **A**ctive	NCF [5]	Receiver Status
LMBT	**L**oss of **M**id **B**it **T**ransition	ECR [1]	Receiver Error Code
LOR	**L**ock **O**ut **R**emote	ACR [1]	Remote Interface
LOOP	internal **LOOP**-back	TMR [6]	Transceiver Control
LTA	**L**ine **T**urn **A**round	NCF [4]	Receiver Status
N	**N**egative	CCR [3]	Arithmetic Flag
OVF	receiver **OV**er**F**low	ECR [4]	Receiver Error Code
OWP	**O**dd **W**ord **P**arity	TCR [3]	Transmitter Control
PAR	**PAR**ity error	ECR [3]	Receiver Error Code
POLL	**POLL**	NCF [0]	Receiver Status
PS2–0	**P**rotocol **S**elect	TMR [2–0]	Transceiver Control
RA	**R**eceiver **A**ctive	TSR [4]	Receiver Status
RAR	**R**eceived **A**uto-**R**esponse	NCF [2]	Receiver Status
RDIS	**R**eceiver **DIS**abled while active	ECR [0]	Receiver Error Code
RE	**R**eceiver **E**rror	TSR [5]	Receiver Status
RF10–8	**R**eceive **F**IFO	TSR [2–0]	Receiver Control
RFF	**R**eceive **F**IFO **F**ull	NCF [6]	Receiver Status
RIN	**R**eceiver **IN**vert	TMR [4]	Receiver Control
RIS1,0	**R**eceiver **I**nterrupt **S**elect	ICR [7,6]	Interrupt Control
RLQ	**R**eceive **L**ine **Q**uiesce	TCR [7]	Receiver Control
RPEN	**R**e**P**eat **EN**able	TMR [5]	Receiver Control
RR	**R**emote **R**ead	CCR [6]	Remote Interface
RTF7–0	**R**eceive/**T**ransmit **F**IFO	RTR [7–0]	Transceiver Control
RW	**R**emote **W**rite	CCR [5]	Remote Interface
SEC	**S**elect **E**rror **C**odes	TCR [6]	Receiver Control
SLR	**S**elect **L**ine **R**eceiver	TCR [5]	Receiver Control
TA	**T**ransmitter **A**ctive	TSR [6]	Transmitter Status
TCS1,0	**T**ransceiver **C**lock **S**elect	DCR [6,5]	Transceiver Control
TF10–8	**T**ransmit **F**IFO	TCR [2–0]	Transmitter Control

6.0 Reference Section (Continued)

6.2.1 Bit Index (Continued)

An alphabetical listing of all status/control bits in the CPU-addressable special function registers, with a brief summary of function. Detailed definitions are provided in Section 6.2.3, Bit Definition Tables.

Bit	Name	Location	Function
TFE	**T**ransmit **F**IFO **E**mpty	NCF [7]	Transmitter Status
TFF	**T**ransmit **F**IFO **F**ull	TSR [7]	Transmitter Status
TIN	**T**ransmitter **IN**vert	TMR [3]	Transmitter Control
TLD	**T**imer **L**oa**D**	ACR [6]	Timer
TM7–0	**TiM**er	TRL [7–0]	Timer
TM15–8	**TiM**er	TRH [7–0]	Timer
TMC	**TiM**er **C**lock select	ACR [5]	Timer
TO	**T**ime **O**ut flag	CCR [7]	Timer
TRES	**T**ransceiver **RES**et	TMR [7]	Transceiver Control
TST	**T**imer **S**tar**T**	ACR [7]	Timer
V	o**V**erflow	CCR [2]	Arithmetic Flag
Z	**Z**ero	CCR [0]	Arithmetic Flag

6.2.2 Register Description

A list of all CPU-addressable special function registers, in alphabetical order.

The Remote Interface Configuration register {RIC}, which is addressable only by the remote system, is not included. See Section 6.3, Remote Interface Reference for details of the function of this register.

Each register is listed together with its address, the type of access available, and a functional description of each bit. Further details on each bit can be found in Section 6.2.3, Bit Definition Tables.

ACR AUXILIARY CONTROL REGISTER
[Main R3; read/write]

7	6	5	4	3	2	1	0
TST	TLD	TMC	BIC	rsv	COD	LOR	GIE

rsv . . . state is undefined at all times.

TST — **Timer StarT** . . . When high, the timer is enabled and will count down from it's current value.

When low, timer is disabled. Timer is stopped by writing a 0 to [TST].

TLD — **Timer LoaD** . . . When high, generates timer load pulse. Cleared when load complete.

TMC — **TiMer Clock select** . . . Selects timer clock frequency. Should not be written when [TST] is high. Can be written at same time as [TST] and [TLD].

TMC	Timer Clock
0	(CPU-CLK)/16
1	(CPU-CLK)/2

BIC — **Bi-directional Interrupt Control** . . . Controls direction of BIRQ.

BIC	BIRQ
0	Input
1	Output

COD — **Clock Out Disable** . . . When high, CLK-OUT output is at TRI-STATE.

LOR — **Lock Out Remote** . . . When high, a remote system is prevented from accessing the BCP.

GIE — **Global Interrupt Enable** . . . When low, disables all maskable interrupts. When high, works with [IM4–0] to enable maskable interrupts.

4TR — **4 T-state Read** . . . When high, $\overline{\text{READ}}$ strobe timing is changed to allow more time between the TRI-STATE of the AD lines by the BCP and the falling of the $\overline{\text{READ}}$ strobe. All data memory reads take four T-states when this bit is set. See Section 2.2.2 for more information.

6.0 Reference Section (Continued)

ATR AUXILIARY TRANSCEIVER REGISTER

[Alternate R2; read/write]

7	6	5	4	3	2	1	0
AT7	AT6	AT5	AT4	AT3	AT2	AT1	AT0

AT7–0 — **Auxiliary Transceiver** . . . In 5250 protocol modes, bits 2–0 define the receive station address, and bits 7–3 control the amount of time TX-ACT stays asserted after the last fill bit.

In 8-bit protocol modes, bits 7–0 define the receive station address.

For further information, see Section 3.0 Transceiver.

ATR 7–3	TX-ACT Hold Time (μs) (if TCLK = 8 MHz)
0 0 0 0 0	0
0 0 0 0 1	0.5
0 0 0 1 0	1.0
0 0 0 1 1	1.5
↓	↓
1 1 1 1 1	15.5

CCR CONDITION CODE REGISTER

[Main R0; bits 0–3, 5–7 read/write, bit 4 read only]

7	6	5	4	3	2	1	0
TO	RR	RW	BIRQ	N	V	C	Z

TO — **Time Out flag** . . . Set high when timer counts to zero. Cleared by writing a 1 to this location or by stopping timer (by writing a 0 to [TST]).

RR — **Remote Read** . . . Set on the trailing edge of a $\overline{\text{REM-RD}}$ pulse, if $\overline{\text{RAE}}$ is asserted and {RIC} is pointing to Data Memory. Cleared by writing a 1 to this location.

RW — **Remote Write** . . . Set on the trailing edge of a $\overline{\text{REM-WR}}$ pulse, if $\overline{\text{RAE}}$ is asserted and {RIC} is pointing to Data Memory. Cleared by writing a 1 to this location.

BIRQ — **Bi-directional Interrupt ReQuest** . . . [Read only]. Reflects the logic level of the Bi-directional interrupt pin, $\overline{\text{BIRQ}}$. Updated at the beginning of each instruction cycle.

N — **Negative** . . . A high level indicates a negative result generated by an arithmetic, logical or shift instruction.

V — **oVerflow** . . . A high level indicates an overflow condition generated by an arithmetic instruction.

C — **Carry** . . . A high level indicates a carry or borrow generated by an arithmetic instruction. During a shift/rotate operation the state of the last bit shifted out appears in this location.

Z — **Zero** . . . A high level indicates a zero result generated by an arithmetic, logical or shift instruction. Further information: Section 2.2.1 ALU, Section 2.2.3 Interrupts.

DCR DEVICE CONTROL REGISTER

[Alternate R0; read/write]

7	6	5	4	3	2	1	0
CCS	TCS1	TCS0	IW1	IW0	DW2	DW1	DW0

CCS — **CPU Clock Select** . . . Selects CPU clock frequency. OCLK represents the frequency of the on-chip oscillator, or the externally applied clock on input X1.

CCS	CPU CLK
0	OCLK
1	OCLK/2

TCS1,0 — **Transceiver Clock Select** . . . Selects transceiver clock, TCLK, frequency.

OCLK represents the frequency of the on-chip oscillator, or the externally applied clock on input X1. X-TCLK is the external transceiver clock input.

TCS1,0	TCLK
0 0	OCLK
0 1	OCLK/2
1 0	OCLK/4
1 1	X-TCLK

IW1,0 — **Instruction memory Wait-state select** . . . Selects from 0 to 3 wait states for accessing instruction memory.

DW2–0 — **Data memory Wait-state select** . . . Selects from 0 to 7 wait states for accessing data memory.

DS DATA STACK

[Main R31; read/write]

7	6	5	4	3	2	1	0
DS7	DS6	DS5	DS4	DS3	DS2	DS1	DS0

DS7–0 — **Data Stack** . . . Data stack input/output port. Stack is 16 bytes deep. Further information: Section 2.1.1.8 Stack Registers.

rsv . . . state is undefined at all times.

ECR ERROR CODE REGISTER
[Alternate R4 with [SEC] high; read only]

7	6	5	4	3	2	1	0
rsv	rsv	rsv	OVF	PAR	IES	LMBT	RDIS

OVF — **Receiver oVerFlow** . . . Set when the receiver has processed 3 words and another complete frame is received before the FIFO is read by the CPU. Cleared by reading {ECR} or by asserting [TRES].

PAR — **PARity error** . . . Set when bad (odd) overall word parity is detected in any receive frame. Cleared by reading {ECR} or by asserting [TRES].

IES — **Invalid Ending Sequence** . . . Set when the ''mini-code violation'' is not correct during a 3270, 3299, or 8-bit ending sequence. Cleared by reading {ECR} or by asserting [TRES].

LMBT — **Loss of Mid-Bit Transition** . . . Set when the expected Manchester Code mid-bit transition does not occur within the allowed window. Cleared by reading {ECR} or by asserting [TRES].

RDIS — **Receiver DISabled while active** . . . Set when transmitter is activated while receiver is active, without RPEN being asserted. Cleared by reading {ECR} or by asserting [TRES]. Further information: Section 3.2 Transceiver Functional Description.

FBR FILL-BIT REGISTER
[Alternate R3; read/write]

7	6	5	4	3	2	1	0
FB7	FB6	FB5	FB4	FB3	FB2	FB1	FB0

FB7–0 — **Fill Bits** . . . 5250 fill-bit control. Further information: Section 3.0 Transceiver.

IBR INTERRUPT BASE REGISTER
[Alternate R1; read/write]

7	6	5	4	3	2	1	0
IV15	IV14	IV13	IV12	IV11	IV10	IV9	IV8

IV15–8— **Interrupt Vector** . . . High byte of interrupt and trap vectors. Further information: Section 2.2.3, Interrupts.

Interrupt Vector

IBR	0	0	vector address
15	8	5	0

The interrupt vector is obtained by concatenating {IBR} with the vector address:

Interrupt	Vector Address	Priority
$\overline{\text{NMI}}$	0 1 1 1 0 0	—
Receiver	0 0 0 1 0 0	1 high
Transmitter	0 0 1 0 0 0	2 ↑
Line Turn Around	0 0 1 1 0 0	3
Bi-directional	0 1 0 0 0 0	4 ↓
Timer	0 1 0 1 0 0	5 low

ICR INTERRUPT CONTROL REGISTER
[Main R2; read/write]

7	6	5	4	3	2	1	0
RIS1	RIS0	rsv	IM4	IM3	IM2	IM1	IM0

rsv . . . state is undefined at all times

RIS1,0 — **Receiver Interrupt Select** . . . Defines the source of the Receiver Interrupt.

RIS1,0	Interrupt Source
0 0	RFF + RE
0 1	DAV + RE
1 0	(unused)
1 1	RA

"+" indicates logical "or"

Further information: Section 3.2.3 Transceiver Interrupts.

IM4–0 — **Interrupt Masks** . . . Each bit, when set high, masks an interrupt. IM3 functions as an interrupt mask only if BIRQ is defined as an input. When BIRQ is defined as an output, IM3 controls the state of BIRQ.

IM4–0	Interrupt
0 0 0 0 0	No Mask
X X X X 1	Receiver
X X X 1 X	Transmitter
X X 1 X X	Line Turn-Around
X 1 X X X	Bi-Directional
1 X X X X	Timer

Further information: Section 2.2.3 Interrupts.

ISP INTERNAL STACK POINTER
[Main R30; read/write]

7	6	5	4	3	2	1	0
ASP3	ASP2	ASP1	ASP0	DSP3	DSP2	DSP1	DSP0

ASP3–0 — **Address Stack Pointer** . . . Input/output port of the address stack pointer. Further information: Section 2.1.1.8 Stack Registers.

DSP3–0 — **Data Stack Pointer** . . . Input/output port of the data stack pointer. Further information: Section 2.1.1.8 Stack Registers.

NCF NETWORK COMMAND FLAG REGISTER
[Main R1; read only]

7	6	5	4	3	2	1	0
TFE	RFF	LA	LTA	DEME	RAR	ACK	POLL

TFE — **Transmit FIFO Empty** . . . Set high when the FIFO is empty. Cleared by writing to {RTR}.

RFF — **Receive FIFO Full** . . . Set high when the Receive FIFO contains 3 received words. Cleared by reading to {RTR}.

LA — **Line Active** . . . Indicates activity on the receiver input. Set high on any transition; cleared after detecting no input transitions for 16 TCLK periods.

LTA — **Line Turn Around** . . . Set high when end of message is received. Cleared by writing to {RTR}, writing a "1" to this location, or by asserting [TRES].

DEME — **Data Error or Message End** . . . In 3270 & 3299 modes, asserted when a byte parity error is detected. In 5250 modes, asserted when the [111] station address is decoded and [DAV] is asserted. Cleared by reading {RTR}. Undefined in 8-bit modes and in the first frame of 3299 modes.

RAR — **Received Auto-Response** . . . Set high when a 3270 Auto-Response message is decoded and [DAV] is asserted. Cleared by reading {RTR}. Undefined in 5250 and 8-bit modes and in the first frame of 3299 modes.

ACK — **Poll/ACKnowledge** . . . Set high when a 3270 poll/ack command is decoded and [DAV] is asserted. Cleared by reading {RTR}. Undefined in 5250 and 8-bit modes and in the first frame of 3299 modes.

POLL — **POLL** . . . Set high when a 3270 poll command is decoded and [DAV] is asserted. Cleared by reading {RTR}. Undefined in 5250 and 8-bit modes and in the first frame of 3299 modes. Further information: Section 3.0 Transceiver.

RTR RECEIVE/TRANSMIT REGISTER
[Alternate R4; read/write]

7	6	5	4	3	2	1	0
RTF7	RTF6	RTF5	RTF4	RTF3	RTF2	RTF1	RTF0

RTF7–0 — **Receive Transmit FIFO's** . . . Input/output port to the least significant eight bits of receive and transmit FIFO's. [OWP], [TF10–8] and [RTF7–0] are pushed onto the transmit FIFO on moves into {RTR}. [RF10–8] and [RTF7–0] are popped from receiver FIFO on moves out of {RTR}. Further information: Section 3.0 Transceiver.

TCR TRANSCEIVER COMMAND REGISTER
[Alternate R6; read/write]

7	6	5	4	3	2	1	0
RLQ	SEC	SLR	ATA	OWP	TF10	TF9	TF8

RLQ — **Receive Line Quiesce** . . . Selects number of line quiesce bits the receiver looks for.

RLQ	Number of Quiesces
0	2
1	3

SEC — **Select Error Codes** . . . When high {ECR} is switched into {RTR} location.

SLR — **Select Line Receiver** . . . Selects the receiver input source.

SLR	Source
0	DATA-IN
1	On-chip analog line receiver

ATA — **Advance Transmitter Active** . . . When high, TX-ACT is advanced one half bit time so that the transmitter can generate 5.5 line quiesce pulses.

OWP — **Odd Word Parity** . . . Controls transmitter word parity.

OWP	Word Parity
0	Even
1	Odd

TF10–8 — **Transmit FIFO** . . . [OWP], [TF10–8] and [RTF7–0] are pushed onto transmit FIFO on moves into {RTR}.

Further information: Section 3.0 Transceiver.

TMR TRANSCEIVER MODE REGISTER
[Alternate R7; read/write]

7	6	5	4	3	2	1	0
TRES	LOOP	RPEN	RIN	TIN	PS2	PS1	PS0

TRES — **Transceiver RESet** . . . Resets transceiver when high. Transceiver can also be reset by $\overline{\text{RESET}}$, without affecting [TRES].

LOOP — **Internal LOOP-back** . . . When high, TX-ACT is disabled (held at 0) and transmitter serial data is internally directed to the receiver serial data input.

RPEN — **RePeat ENable** . . . When high, the receiver can be active at the same time as the transmitter.

RIN — **Receiver INvert** . . . When high, the receiver serial data is inverted.

TIN — **Transmitter INvert** . . . When high the transmitter serial data outputs are inverted.

PS2–0 — **Protocol Select** . . . Selects protocol for both transmitter and receiver.

PS2–0	Protocol
0 0 0	3270
0 0 1	3299 multiplexer
0 1 0	3299 controller
0 1 1	3299 repeater
1 0 0	5250
1 0 1	5250 promiscuous
1 1 0	8-bit
1 1 1	8-bit promiscuous

Further information: Section 3.0 Transceiver.

TRH TIMER REGISTER — HIGH
[Main R29; read/write]

7	6	5	4	3	2	1	0
TM15	TM14	TM13	TM12	TM11	TM10	TM9	TM8

TM15–8 — **TiMer** . . . Input/output port of high byte of timer. Further information: Section 2.1.1.4 Timer Registers.

TRL TIMER REGISTER—LOW
[Main R28; read/write]

7	6	5	4	3	2	1	0
TM7	TM6	TM5	TM4	TM3	TM2	TM1	TM0

TM7–0— **TiMer** . . . Input/output port of low byte of timer. Further information: Section 2.1.1.4 Timer Registers.

TSR TRANSCEIVER STATUS REGISTER
[Alternate R5; read only]

7	6	5	4	3	2	1	0
TFF	TA	RE	RA	DAV	RF10	RF9	RF8

TFF — **Transmit FIFO Full** . . . Set high when the transmit FIFO is full. {RTR} must not be written to when [TFF] is high.

TA — **Transmitter Active** . . . Reflects the state of TX-ACT, indicating that data is being transmitted. Unlike TX-ACT, however, [TA] is not disabled by [LOOP].

RE — **Receiver Error** . . . Set high when a receiver error is detected. Cleared by reading {ECR} or by asserting [TRES].

RA — **Receiver Active** . . . Set high when a valid starting sequence is received. Cleared when either an end of message or an error is detected. In 5250 modes, [RA] is cleared at the same time as [LA].

DAV — **Data AVailable** . . . Set high when valid data is available in {RTR} and {TSR}. Cleared by reading {RTR}, or when an error is detected.

RF10–8— **Receive FIFO** . . . [RF10–8] and [RTF7–0] reflect the state of the top word of the receive FIFO.

Further information: Section 3.0 Transceiver.

6.2.3 Bit Definition Tables

The following tables describe the location and function of all control and status bits in the various BCP addressable special function registers. The Remote Interface Configuration register, {RIC}, which is addressable only by a remote processor is not included.

6.2.3.1 Processor

	Bit	Name	Location	Reset State	Function
Timing/ Control	CCS	**C**PU **C**lock **S**elect	DCR [7]	1	Selects CPU clock frequency. CCS | CPU CLK 0 | OCLK 1 | OCLK/2 Where OCLK is the frequency of the on-chip oscillator, or the externally applied clock on input X1.
	DW2–0	**D**ata memory **W**ait-state select	DCR [2–0]	111	Selects from 0 to 7 wait states for accessing data memory.
	IW1,0	**I**nstruction memory **W**ait-state select	DCR [4,3]	11	Selects from 0 to 3 wait states for accessing instruction memory.
	COD	**C**lock **O**ut **D**isable	ACR [2]	0	When high, CLK-OUT is at TRI-STATE.
	4TR	4 T-state Read	ACR[3]	0	When high, data memory reads take four T-states.
Remote Interface	LOR*	**L**ock **O**ut **R**emote	ACR [1]	0	When high, a remote processor is prevented from accessing the BCP or its memory.
	RR*	**R**emote **R**ead	CCR [6]	0	Set on the trailing edge of a $\overline{\text{REM-RD}}$ pulse, if $\overline{\text{RAE}}$ is asserted and {RIC} is pointing to Data Memory. Cleared by writing a 1 to [RR].
	RW*	**R**emote **W**rite	CCR [5]	0	Set on the trailing edge of a $\overline{\text{REM-WR}}$ pulse, if $\overline{\text{RAE}}$ is asserted and {RIC} is pointing to Data Memory. Cleared by writing a 1 to [RW].
Interrupt Control	BIC	**B**i-directional **I**nterrupt **C**ontrol	ACR [4]	0	Controls the direction of $\overline{\text{BIRQ}}$. BIC | $\overline{\text{BIRQ}}$ 0 | Input 1 | Output
	BIRQ	**B**i-directional **I**nterrupt **R**e**Q**uest	CCR [4]	X	[Read Only]. Reflects the logic level of the $\overline{\text{BIRQ}}$ input. Updated at the beginning of each instruction cycle.
	GIE	**G**lobal **I**nterrupt **E**nable	ACR [0]	0	When low, disables all maskable interrupts. When high, works with [IM4–0] to enable maskable interrupts.
	IM4–0	**I**nterrupt **M**ask select	ICR [4–0]	11111	Each bit, when set high, masks an interrupt. IM4–0 | Interrupt | Priority 0 0 0 0 0 | No Mask | — X X X X 1 | Receiver | 1 High X X X 1 X | Transmitter | 2 ↑ X X 1 X X | Line Turn-Around | 3 X 1 X X X | Bi-Directional | 4 ↓ 1 X X X X | Timer | 5 Low IM3 functions as an interrupt mask only when $\overline{\text{BIRQ}}$ is defined as an input. When $\overline{\text{BIRQ}}$ is defined as an output, IM3 controls the state of $\overline{\text{BIRQ}}$.

*These bits represent the only visibility and control that the processor has into the operation of the remote interface controller. The Remote Interface Configuration register, {RIC}, accessible only by a remote processor, provides further control functions. See Remote Interface section for more information.

The following tables describe the location and function of all control and status bits in the various BCP addressable special function registers. The Remote Interface Configuration register, {RIC}, which is addressable only by a remote processor is not included.

6.2.3.1 Processor (Continued)

	Bit	Name	Location	Reset State	Function
Interrupt Control (Continued)	IV15–8	**I**nterrupt **V**ector	IBR [7–0]	0000 0000	High byte of interrupt and trap vectors. The interrupt vector is obtained by concatenating {IBR} with the vector address: **Interrupt** / **Vector Address** $\overline{NMI}$ 0 1 1 1 0 0 Receiver 0 0 0 1 0 0 Transmitter 0 0 1 0 0 0 Line Turn Around 0 0 1 1 0 0 Bi-Directional 0 1 0 0 0 0 Timer 0 1 0 1 0 0 **Interrupt Vector** IBR | 0 0 | vector address 15 8 5 0
	RIS1,0	**R**eceiver **I**nterrupt **S**elect	ICR [7,6]	11	Defines the source of the receiver interrupt. **RIS1,0** / **Interrupt Source** 0 0 RFF + RE 0 1 DAV + RE 1 0 (unused) 1 1 RA
Address and Data Stacks	ASP3–0	**A**ddress **S**tack **P**ointer	ISP [7–4]	0000	Address stack pointer. Writing to this location changes the value of the pointer.
	DSP3–0	**D**ata **S**tack **P**ointer	ISP [3–0]	0000	Data stack pointer. Writing to this location changes the value of the pointer.
	DS7–0	**D**ata **S**tack	DS [7–0]	XXXX XXXX	Data Stack Input/Output port. Stack is 16 bytes deep.
Arithmetic Flags	C	**C**arry	CCR [1]	0	A high level indicates a carry or borrow, generated by an arithmetic instruction. During a shift/rotate operation the state of the last bit shifted out appears in this location.
	N	**N**egative	CCR [3]	0	A high level indicates a negative result generated by an arithmetic, logical, or shift instruction.
	V	o**V**erflow	CCR [2]	0	A high level indicates an overflow condition, generated by an arithmetic instruction.
	Z	**Z**ero	CCR [0]	0	A high level indicates a zero result generated by an arithmetic, logical, or shift instruction.

6.2.3. Bit Definition Tables (Continued)

The following tables describe the location and function of all control and status bits in the various BCP addressable special function registers. The Remote Interface Configuration register, {RIC}, which is addressable only by a remote processor is not included.

6.2.3.1 Processor (Continued)

	Bit	Name	Location	Reset State	Function
Timer	TLD	**T**imer **L**oa**D**	ACR [6]	0	Set high to load timer. Cleared automatically when load complete.
	TM15–8	**Ti**M**e**r	TRH [7–0]	XXXX XXXX	Input/output port of high byte of timer.
	TM7–0	**Ti**M**e**r	TRL [7–0]	XXXX XXXX	Input/output port of low byte of timer.
	TMC	**Tim**er **C**lock select	ACR [5]	0	Selects timer clock frequency. Must not be written when [TST] high. Can be written at same time as [TST] and [TLD]. **TMC** / **Timer Clock** 0 / CPU-CLK/16 1 / CPU-CLK/2
	TO	**T**ime **O**ut flag	CCR [7]	0	Set high when timer counts down to zero. Cleared by writing a 1 to [TO] or by stopping the timer (by writing a 0 to [TST]).
	TST	**T**imer **S**tar**T**	ACR [7]	0	When high, timer is enabled and will count down from its current value. Timer is stopped by writing a 0 to this location.

6.2.3.2 Transceiver

Table includes control and status bits only. It does not include definitions of bit fields provided for the formatting (de-formatting) of data frames. For further information see the Transceiver section.

	Bit	Name	Location	Reset State	Function
Transceiver Control	LOOP	internal **LOOP**-back	TMR [6]	0	When high, TX-ACT is disabled (held at 0) and transmitter serial data is internally directed to the receiver serial data input.
	PS2–0	**P**rotocol **S**elect	TMR [2–0]	000	Selects protocol for both transmitter and receiver. **PS2–0** / **Protocol** 0 0 0 / 3270 0 0 1 / 3299 Multiplexer 0 1 0 / 3299 Controller 0 1 1 / 3299 Repeater 1 0 0 / 5250 1 0 1 / 5250 Promiscuous 1 1 0 / 8-bit 1 1 1 / 8-bit Promiscuous
	RTF7–0	**R**eceive/**T**ransmit **F**IFOs	RTR [7–0]	XXXX XXXX	Input/output port of the least significant 8 bits of receive and transmit FIFOs. [OWP], [TF10–8] and [RTF7–0] are pushed onto the transmit FIFO on moves to {RTR}. [RF10–8] and [RTF7–0] are popped from receive FIFO on moves from {RTR}.

Table includes control and status bits only. It does not include definitions of bit fields provided for the formatting (de-formatting) data frames. For further information see the Transceiver section.

	Bit	Name	Location	Reset State	Function
Transceiver Control (Continued)	TCS1,0	**T**ransceiver **C**lock **S**elect	DCR [6,5]	10	Selects transceiver clock, TCLK, source. **TCS1,0** / **TCLK** 0 0 / OCLK 0 1 / OCLK/2 1 0 / OCLK/4 1 1 / X-TCLK OCLK is the frequency of the on-chip oscillator, or the externally applied clock on input X1. X-TCLK is the external transceiver clock input.
	TRES	**T**ransceiver **RES**et	TMR [7]	0	Resets transceiver when high. Transceiver can also be reset by $\overline{\text{RESET}}$, without affecting [TRES].
Transmitter Control	ATA	**A**dvance **T**ransmitter **A**ctive	TCR [4]	0	When high, TX-ACT is advanced one half bit time so that the transmitter can generate 5.5 line quiesce pulses.
	AT7–3	**A**uxiliary **T**ransceiver control	ATR [7–3]	XXXXX	In 5250 modes. Controls the time TX-ACT is held after the last fill bit. **AT7–3** / **TX-ACT Hold Time (μs) (If TCLK = 8 MHz)** 0 0 0 0 0 / 0 0 0 0 0 1 / 0.5 0 0 0 1 0 / 1 ↓ / ↓ 1 1 1 1 1 / 15.5
	FB7–0	**F**ill **B**it select	FBR [7–0]	XXXX XXXX	The value in this register contains the 1's complement of the number of additional 5250 fill bits selected.
	OWP	**O**dd **W**ord **P**arity	TCR [3]	0	Controls transmitter word parity. **OWP** / **Word Parity** 0 / Even 1 / Odd
	TF10–8	**T**ransmit **F**IFO	TCR [2–0]	000	[OWP], [TF10–8] and [RTF7–0] are pushed onto the transmit FIFO on moves to {RTR}.
	TIN	**T**ransmitter **IN**vert	TMR [3]	0	When high, the transmitter serial data outputs are inverted.
Receiver Control	AT7–0	**A**uxiliary **T**ransceiver control	ATR [7–0]	XXXX XXXX	In 5250 modes, [AT2–0] contains the station address. In 8-bit modes, [AT7–0] contains the station address.
	RF10–8	**R**eceive **F**IFO	TSR [2–0]	XXX	Reflects the state of the most significant 3 bits in the top location of the receive FIFO.
	RIN	**R**eceiver **IN**vert	TMR [4]	0	When high, the receiver serial data is inverted.
	RLQ	**R**eceive **L**ine **Q**uiesce	TCR [7]	1	Selects number of line quiesce bits the receiver requires before it will indicate receipt of a valid start sequence. **RLQ** / **Number of Line Quiesce Pulses** 0 / 2 1 / 3
	RPEN	**R**e**P**eat **EN**able	TMR [5]	0	When high, the receiver can be active at the same time as the transmitter.
	SEC	**S**elect **E**rror **C**odes	TCR [6]	0	When high, {ECR} is switched into {RTR} location.

Table includes control and status bits only. It does not include definitions of bit fields provided for the formatting (de-formatting) data frames. For further information see the Transceiver section.

	Bit	Name	Location	Reset State	Function
Receiver Control (Continued)	SLR	**S**elect **L**ine **R**eceiver	TCR [5]	0	Selects the receiver input source. SLR: 0 = DATA-IN; 1 = On-Chip Analog Line Receiver
Transmitter Status	TA	**T**ransmitter **A**ctive	TSR [6]	0	Reflects the state of TX-ACT, indicating that data is being transmitted. Is not disabled by [LOOP].
	TFE	**T**ransmit **F**IFO **E**mpty	NCF [7]	1	Set high when the FIFO is empty. Cleared by writing to {RTR}.
	TFF	**T**ransmit **F**IFO **F**ull	TSR [7]	0	Set high when the FIFO is full. {RTR} must not be written when [TFF] is high.
Receiver Status	ACK	poll/ **ACK**nowledge	NCF [1]	0	Set high when a 3270 poll/ack command is decoded and [DAV] is asserted. Cleared by reading {RTR}. Undefined in 5250 and 8-bit modes and in the first frame of 3299 modes.
	DAV	**D**ata **AV**ailable	TSR [3]	0	Set high when valid data is available in {RTR} and {TSR}. Cleared by reading {RTR}, or when an error is detected.
	DEME	**D**ata **E**rror or **M**essage **E**nd	NCF [3]	0	In 3270 or 3299 modes, asserted when a byte parity error is detected. In 5250 modes, asserted when the [111] station address is decoded and [DAV] is asserted. Undefined in 8-bit modes and first frame of 3299 modes.
	LA	**L**ine **A**ctive	NCF [5]	0	Indicates activity on the receiver input. Set high on any transition; cleared after no input transitions are detected for 16 TCLK periods.
	LTA	**L**ine **T**urn **A**round	NCF [4]	0	Set high when an end of message is detected. Cleared by writing to {RTR}, writing a ‘‘1’’ to [LTA] or by asserting [TRES].
	POLL	**POLL**	NCF [0]	0	Set high when a 3270 Poll command is decoded and [DAV] is asserted. Cleared by reading {RTR}. Undefined in 5250 and 8-bit modes and in the first frame of 3299 modes.
	RA	**R**eceiver **A**ctive	TSR [4]	0	Set high when a valid start sequence is received. Cleared when either an end of message or an error is detected.
	RAR	**R**eceived **A**uto-**R**esponse	NCF [2]	0	Set high when a 3270 Auto-Response message is decoded and [DAV] is asserted. Cleared by reading {RTR}. Undefined in 5250 and 8-bit modes and in the first frame of 3299 modes.
	RE	**R**eceiver **E**rror	TSR [5]	0	Set high when an error is detected. Cleared by reading {ECR} or by asserting [TRES].
	RFF	**R**eceive **F**IFO **F**ull	NCF [6]	0	Set high when the receive FIFO contains 3 received words. Cleared by reading {RTR}.

Table includes control and status bits only. It does not include definitions of bit fields provided for the formatting (de-formatting) data frames. For further information see the Transceiver section.

	Bit	Name	Location	Reset State	Function
Receiver Error Codes	IES	**I**nvalid **E**nding **S**equence	ECR [2]	0	Set when the first mini-code violation is not correct during a 3270, 3299 or 8-bit ending sequence. Cleared by reading {ECR} or asserting [TRES].
	LMBT	**L**oss of **M**id-**B**it **T**ransition	ECR [1]	0	Set when the expected Manchester Code mid-bit transition does not occur within the allowed window. Cleared by reading {ECR} or by asserting [TRES].
	OVF	receiver **OV**er**F**low	ECR [4]	0	Set when the receiver has processed 3 words and another complete frame is received before the FIFO is read by the CPU. Cleared by reading {ECR} or asserting [TRES].
	PAR	**PAR**ity error	ECR [3]	0	Set when bad (odd) overall word parity is detected in any receive frame. Cleared by reading {ECR} or asserting [TRES].
	RDIS	**R**eceiver **DIS**abled while active	ECR [0]	0	Set when transmitter is activated by writing to {RTR} while receiver is still active, without [RPEN] first being asserted. Cleared by reading {ECR} or asserting [TRES].

6.3 REMOTE INTERFACE CONFIGURATION REGISTER

This register can be accessed only by the remote system. To do this, CMD and $\overline{\text{RAE}}$ must be asserted and the [LOR] bit in the {ACR} register must be low.

7	6	5	4	3	2	1	0	
BIS	SS	FW	LR	LW	STRT	MS1	MS0	RIC

BIS **B**idirectional **I**nterrupt **S**tatus . . . Mirrors the state of IM3 ({ICR} bit 3), enabling the remote system to poll and determine the status of the $\overline{\text{BIRQ}}$ I/O. When $\overline{\text{BIRQ}}$ is an output, the remote system can change the state of this output by writing a one to BIS. This can be used as an interrupt acknowledge, whenever $\overline{\text{BIRQ}}$ is used as a remote interrupt. For complete information on the relationship between BIS, IM3 and $\overline{\text{BIRQ}}$, refer to Section 2.2.3 Interrupts.

SS **S**ingle-**S**tep . . . Writing a 1 with STRT low, the BCP will single-step by executing the current instruction and advancing the PC. On power up/reset this bit is low.

FW **F**ast **W**rite . . . When high, with LW low, selects fast write mode for the buffered interface. When low selects slow write mode. On power up/reset this bit is low (LW will also be low, so buffered write mode is selected).

LR **L**atched **R**ead . . . When high selects latched read mode, when low selects buffered read mode. On power up/reset this bit is low. (Buffered read mode is selected.)

LW **L**atched **W**rite . . . When high selects latched write mode, when low selects buffered write mode. On power up/reset this bit is low (FW will also be low, so slow buffered write mode is selected).

STRT **STaRT** . . . The remote system can start and stop the BCP using this bit. On power-up/reset this bit is low (BCP stopped). When set, the BCP begins executing at the current Program Counter address. When cleared, the BCP finishes executing the current instruction, then halts to an idle mode.

In some applications, where there is no remote system, or the remote system is not an intelligent device, it may be desirable to have the BCP power-up/reset running rather than stopped at address 0000H. This can be accomplished by asserting $\overline{\text{REM-RD}}$, $\overline{\text{REM-WR}}$ and $\overline{\text{RESET}}$, with $\overline{\text{RAE}}$ de-asserted. (Refer to Electrical Specification Section for the timing information needed to start the BCP in stand alone mode.)

MS1,0 **M**emory **S**elect **1,0** . . . These two bits determine what the remote system is accessing in the BCP system, according to the following table:

MS1	MS0	Selected Function
0	0	Data Memory
0	1	Instruction Memory
1	0	Program Counter (Low Byte)
1	1	Program Counter (High Byte)

The BCP must be idle for the remote system to read/write Instruction memory or the Program Counter.

All remote accesses are treated the same (independent of where the access is directed using MS0 and MS1), as defined by the configuration bits LW, LR, FW.

If the remote system and the BCP request data memory access simultaneously, the BCP will win first access. If the locks ([LOR], $\overline{\text{LOCK}}$) are not set, the remote system and BCP will alternate access cycles thereafter.

On power-up/reset, MS1,0 points to instruction memory.

Power-up/Reset state of {RIC[7-0]} is |000 000|.

6.4 DEVELOPMENT TOOLS

National Semiconductor provides tools specifically created for the development of products that use the DP8344. These tools consist of the DP8344 BCP Assembler System, the DP8344 BCP Demonstration/Development Kit, and the DP8344 BCP Multi-Protocol Adapter (MPA) Design/Evaluation Kit.

6.4.1 Assembler System

The Assembler System is an MS-DOS compatible program used to translate the DP8344's instruction set into a directly executable machine language. The system contains a macro cross assembler, link editor and librarian. The macro cross assembler provides nested macro definitions and expansions, to automate common instruction sequences, and source file inclusion nested conditional assembly, which allows the assembler to make intelligent decisions concerning instruction sequence based on user directives. The linker allows relocatable object sections to be combined in any desired order. It can also generate a load map which details each section's contribution to the linked module. The librarian allows for the creation of libraries from frequently accessed object modules, which the linker can automatically include to resolve references.

6.4.2 Demonstration/Development Kit

The Demonstration/Development kit is a cost effective development tool that performs functions similar to an in-circuit emulator. The kit, developed by Capstone Technology, Inc., Fremont, California, consists of a DP8344 based development board, a monitor/debugger software package, National Semiconductor's DP8344 video training tapes, and all required documentation. The development board is a full size PC card that contains a 22 square inch area for logic prototype wiring. The monitor/debugger program displays internal register contents and status information. It also provides functions such as execution break points and single stepping.

6.4.3 Multi-Protocol Adapter (MPA) Design/Evaluation Kit

The Multi-Protocol Adapter (MPA) is a PC expansion card that emulates a 3270 or 5250 display terminal and supports industry standard PC emulation software. The MPA comes in a design/evaluation kit that includes the hardware, schematics and PAL equations, and software including all the DP8344 source code. This kit was produced to provide a blueprint for PC emulation products and a cornerstone for all 3270 and 5250 product development using the DP8344. The code was developed in a modular fashion so it can be adapted to any 3270 or 5250 application.

6.4.4 DP8344 BCP Inverse Assembler

The DP8344 BCP Inverse Assembler is a software package for use in an HP 1650A or HP1651A Logic Analyzer, or in an HP16500A Logic Analysis System with an HP 16510A State/Timing Card installed. The inverse Assembler was developed by National Semiconductor to allow disassembly of the DP8344 op-code mnemonics. This allows one to determine the actual execution flow that occurs in the system being developed with the DP8344.

6.5 THIRD PARTY SUPPLIERS

The following section is intended to make the DP8344 Customer aware of products, supplied by companies other than National Semiconductor, that are available for use in developing DP8344 systems. While National Semiconductor has supported these ventures and has become familiar with many of these products, we do not provide technical support, or in any way guarantee the functionality of these products.

6.5.1 Crystal Supplier

The recommended crystal parameters for operation with the DP8344 are given in Section 2.2.4. Any crystal meeting these specifications will work correctly with the DP8344. NEL Frequency Controls, Inc., Burlington, Wisconsin, has developed crystals, the NEL C2570N and NEL C2571N, specifically for the DP8344 which meet these specifications. The C2570N and C2571N are both 18.8696 MHz fundamental mode AT cut quartz crystals. The C2571N has a hold down pin for case ground and a third mechanical tie down. NEL Frequency Controls, Inc. is located at:

NEL Frequency Controls, Inc.
357 Beloit Street
Burlington, Wisconsin 53105
(414) 763-3591

6.5.2 System Development Tools

The DP8344, with its higher level of integration and processing power, has opened the IBM mainframe connectivity market to a wider range of product manufacturers, who until now found the initial cost and time to market prohibitive. This wider base of manufacturers created the opportunity for a more extensive line of development tools that dealt not only with the use of the DP8344 but also with the implementation of the 3270 and 5250 protocols. While National Semiconductor is dedicated to providing the Customer with the proper tools in both areas, we also have aided and encouraged a number of third party suppliers to offer additional development tools. This has further provided an avenue for faster and more reliable product development in this product area. The development tools discussed in this section are controller emulators and line monitors for the IBM 3270/3299 and 5250 protocols.

A controller emulator is a device that emulates an IBM 3x74 cluster controller or a System 3x controller. With the DP8344 both of these controllers can be emulated with the same piece of hardware. The controller emulator allows the designer to issue individual commands or sequences of commands to a peripheral. This is very useful in characterizing existing equipment and testing of products under development. Capstone Technology offers such a product. Their Extended Interactive Controller, part #CT-109, is a single PC expansion card that can emulate both 3270 and 5250 control devices (the 3x74 and System 3X, respectively). Newleaf Technologies, Ltd., Cobham, Surrey, England, and Azure Technology, Inc., Franklin, Mass., also supply products in this area. Newleaf Technology offers the COLT52, a twinax controller emulator, and Azure Technology offers a controller made with their CoaxScope and TwinaxScope line monitors.

A line monitor is a device that monitors all the activity on the coax or twinax cable. The activity includes both the commands from the controller and the responses from the peripheral. These devices typically decode the commands and present them in an easy to read format. The individual transmissions are time stamped to provide the designer with response time information. The line monitors are very useful in characterizing communications traffic and in determining the source of problems during development or in the field. Azure Technology offers both a 3270/3299 (Coax) and 5250 (Twinax) line monitor. Their Coax Scope and Twinax

6.0 Reference Section (Continued)

Scope are single PC expansion cards that can record, decode and display activity on the 3270 coax and 5250 twinax line respectively. These devices also allow the play back of the recorded controller information. Capstone Technology also supplies a line monitor. The CT101C, Network Analysis Monitor (NAM), is a coax line monitor.

These companies can be contacted at the following locations:

Azure Technology, Inc.
38 Pond Street
Franklin, Massachusettes 02038
(508) 520-3800

Capstone Technology
853 Brown Rd., Suite 207
Fremont, California 94539
(415) 438-3500

New Leaf Technology, Ltd.
24A High Street
Cobham
Surrey
KT113EB
ENGLAND
(0932) 66466

For technical assistance in using the DP8344B, contact the BCP Hot Line (817) 468-6676.

TABLE 6-4. DP8344 Application Notes

App Note No.	Title
AN-623	Interfacing Memory to the DP8344B
AN-624	A Combined Coax-Twisted Pair 3270 Line Interface for the DP8344 Biphase Communications Processor
AN-516	Interfacing the DP8344 to Twinax
AN-504	DP8344 BCP Stand-Alone Soft-Load System
AN-499	''Interrupts''-A Powerful Tool of the Biphase Communications Processor
AN-625	JRMK Speeds Command Decoding
AN-627	DP8344 Remote Processor Interfacing
AN-626	DP8344 Timer Application
AN-641	MPA - A Multi-Protocol Terminal Emulation Adapter Using the DP8344
AN-688	The DP8344 BCP Inverse Assembler

6.6 DP8344A AND DP8344B COMPATIBILITY GUIDE

The DP8344B is an enhanced version of the DP8344A, exhibiting improved switching performance and additional functionality. The device has been characterized in a number of applications and found to be a compatible replacement for the DP8344A. Differences between the DP8344A and DP8344B are detailed in this section.

6.6.1 Timing Changes to the CPU

Relative to the DP8344A, the DP8344B incorporates a number of timing changes designed to improve the system interface. These timing changes are improvements in the timing specifications and therefore should allow the DP8344B to drop into existing DP8344A designs without any hardware modifications.

The DP8344A exhibits a small amount of contention between certain bus signals as detailed in the Device Specifications section of this data sheet. The DP8344B interface timing improvements are designed to reduce and/or eliminate this bus contention.

- **70 ns Data Memory**

  At a 20 MHz CPU clock rate, the DP8344B can support 70 ns static RAM for data memory with no wait states. The DP8344A was limited to 55 ns static RAM for data memory with no wait states. (See Section 5.0 Device Specifications.)

- **$\overline{\text{READ}}$**

  The timing of the $\overline{\text{READ}}$ strobe has been improved to reduce bus contention during a data memory access. There is now more time between AD disabled and $\overline{\text{READ}}$ falling as well as one-half T-state between $\overline{\text{READ}}$ rising and AD enabled. In addition, a new 4 T-state read option has been provided to eliminate bus contention. (See Section 5.0 Device Specifications for timing changes, and **4 T-state Read** later in this document for more information on the 4 T-state Read option.)

  The user can therefore choose between a fast read mode (3 T-states) with a small amount of contention and a slower read mode (4 T-states) with no contention.

- **A/AD Bus Timing**

  The timing of the A and AD buses has been changed to eliminate bus contention during remote accesses of data memory. There is now a one-half T-state TRI-STATE zone during the bus transfer from local to remote control and vice versa. (See Section 5.0 Device Specifications.)

- **$\overline{\text{IWR}}$**

  The timing of $\overline{\text{IWR}}$ has been changed such that $\overline{\text{IWR}}$ now falls one T-state earlier. This eliminates bus contention during the start of soft loads. (See Section 5.0 Device Specifications.)

- **IA Bus Softload Timing**

  The auto-increment of the IA bus address during soft loads of instruction memory now occurs one T-state later to maintain in-phase data and thereby eliminate bus contention. (See Seection 5.0 Device Specifications.)

- **$\overline{\text{LCL}}$**

  $\overline{\text{LCL}}$ is now removed when $\overline{\text{REM-RD}}$ is taken high on buffered reads of {RIC}, the program counter, and instruction memory, to eliminate bus contention in this mode. (See Section 5.0 Device Specifications.)

- **RIC**

  The hold time on slow buffered writes to {RIC} and the program counter has been improved. (See Section 5.0 Device Specifications.)

- **''Kick-start''**

  The hold time on $\overline{\text{REM-WR}}$ and $\overline{\text{REM-RD}}$ to $\overline{\text{RESET}}$ to ''kick-start'' the CPU has been improved. (See Section 5.0 Device Specifications.)

6.6.2 Additional Functionality of the DP8344B

6.6.2.1 4 T-state Read

To eliminate bus contention during memory accesses, a new optional read mode has been created, controlled by

[4TR] in {ACR}. When a one is written to this bit, all subsequent data memory read operations expand to 4 T-states with an extra one-half T-state between the falling edge of ALE and the falling edge of $\overline{\text{READ}}$. This eliminates bus contention on data memory read operations. After a BCP reset, or when a zero is written to this bit, the DP8344B data memory read operations operate in 3 T-states, as in the DP8344A, in which this bit was unused. (See Section 2.2.2 for more information.)

6.6.2.2 A/AD Reset State

After a BCP reset, the index registers and the A and AD buses will be zero. In the DP8344A, their states were undefined after a reset.

6.6.2.3 RIC

Each time instruction memory is selected via {RIC[1,0]} (i.e., {RIC} is set to XXXX XX01 binary), the next read (or write) of instruction memory by a remote processor will always return (or update) the low order 8 bits of the 16 bit instruction location pointed to by the program counter. In the DP8344A, setting {RIC} had no affect on which instruction memory byte would next be fetched and an algorithm had to be developed to determine this. (See Section 4.1.2 for more information.)

6.6.2.4 Transceiver

When the Transceiver is reset, $\overline{\text{DATA-OUT}}$ now goes into a state equal to [TIN] ⊕ [ATA], which eliminates coincident transitions on $\overline{\text{DATA-OUT}}$ and DATA-DLY with TX-ACT. (See Section 3.2 for more information.

6.7 REPORT BUGS

6.7.1 History

The DP8344 Data Sheet Reference, first published 10/29/87 (rev. 3.6), listed a total of 13 bugs. All these bugs were corrected in the DP8344A, released to production April 1989. Subsequent to this date, an additional bug has been reported. This bug is present in all versions of the BCP: DP8344, DP8344A and DP8344B.

For additional information regarding differences in functionality between the DP8344B and DP8344A, see Section 6.6.

6.7.2 LJMP, LCALL Address Decode

The LJMP and LCALL instructions to the address range $Af00_h$ through $AF7F_h$ do not function correctly. Both conditional and unconditional LCALL or LJMP instructions to this address range will not decode as LCALL or LJMP instructions. Instead the address field will be incorrectly decoded as the instruction. Thus a LJMP or LCALL to an instruction in the address range $AF00_h$ through $AF7F_h$ will be decoded as a RETF instruction.

Example: the instruction		`LJMP AF00`
	will be decoded as	`AF00`
	which is	`RETF 000, 00`

Note that LJMP and LCALL to all other addresses work correctly.

The LJMP or LCALL instruction should therefore not be used to transfer program control to an instruction in the range $AF00_h$ to $AF7F_h$.

6.7.2.1 Suggested Work-around

The simplest work-around is not to place any code necessary for system operation in the affected address range. This can be accomplished by creating a section of ''filler'' code that will occupy the instruction address range $AF00_h$ to $AF7F_h$. As an example, the ''filler'' section of code could be as follows:

```
FILLER: .SECT X     ; Start of ""filler'' code section
        .REPEAT 128 ; Repeat the following
                      instruction 128 times
        JMP $       ; Jump to self
        .ENDR       ; End of repeat block
        .END
```

The JMP $ instruction causes an infinite loop at that instruction. Thus one would be able to determine if the program inadvertently entered the ''filler'' section of code. The repeat 128 instruction causes the section to occupy 128 bytes of instruction memory which is the size of the affected address range.

Next, by using the Linker in the DP8344 BCP Assembler System, one can specify that this ''filler'' section of code must occupy instruction memory starting at address $AF00_h$ by using the -L option. For example, the following commands can be entered at the DOS command line to invoke the Assembler and Linker (this assumes that the ''filler'' section is located in the file FILLER.BCP):

```
NBCPASM FILLER.BCP
NLINK -LFILLER=AF00 FILLR.BC0
```

This will prevent any other section of code from occupying the range which the ''filler'' section of code is located in. Hence, one would not have to be concerned about using labels to specify the address in LJMP and LCALL instruction.

6.8 GLOSSARY

3270—An IBM **communication protocol** originally developed for the 370 class mainframe that implements a star topology using a single **coax** cable per slave device. In this master-slave protocol, all communication is initiated by the **controller** (master) and responses are returned by the terminal or other attached device (slave). The data is transmitted using **biphase encoding** at a bit rate of 2.3587 MHz.

3299—A **communications protocol** that is the **3270** protocol with an eight bit address frame added to the beginning of each controller transmission between the **start sequence** and the first coax word. Currently, IBM only uses three bits of the address field which allows up to eight devices to communicate with the **controller** through a **multiplexer**.

5250—An IBM **communications protocol** originally developed for the Series 3 that became widely used on the System 34/36/38 family of minicomputers and currently the AS/400. It uses a **multidrop** bus topology on **twin-ax** cable. This protocol is a master-slave type. The data is transmitted using **bi-phase encoding** at a bit rate of 1 MHz.

accumulator—The implied source register of one operand for some arithmetic operations. In the **BCP**, R8 in the currently enabled bank acts as the accumulator.

ALU—The Arithmetic Logic Unit, a component of the CPU that performs all arithmetic (addition and subtraction), logical (AND, OR, XOR, compare, bit test, and complement), rotational, and shifting operations.

ALU flags—Bits that indicate the result of certain ALU functions.

6.0 Reference Section (Continued)

banked registers—Two or more sets of CPU registers that occupy the same register space, but only one of which is accessible at a time.

barrel shifter—Dedicated hardware for shifting and rotating.

BCP—An abbreviation for Biphase Communications Processor, the National Semiconductor DP8344.

biphase—In this communications signal encoding technique, the data is divided into discrete bit time intervals denoted by a transition in the center of the bit time. This technique combines the clock and data information into one transmission. In **3270** and **3299** protocols, a **mid-bit** transition from low to high represents a bi-phase 1, and a **mid-bit** transition from high to low represents a bi-phase 0. For the **5250** protocol, the definition of biphase logic levels is reversed. Biphase encoding is also called **Manchester II encoding.**

BIRQ—The Bidirectional Interrupt ReQuest. Without any other notation, BIRQ will refer to the BIRQ interrupt itself. BIRQ with a bar on top of it ($\overline{\text{BIRQ}}$) is used where the pin is referenced. BIRQ in brackets ([BIRQ]) is bit 4 in the {CCR} register.

coax—(1) RG-62A/U 93Ω coaxial cable that is used in **3270** protocol systems. (2) Sometimes, this term is used to refer to the **3270** protocol itself.

code violation—A violation of the **bi-phase** encoding format that is part of the **start sequence**. In **3270, 3299,** and the **general purpose 8-bit mode**, the code violation is 1½ bit times low and then 1½ bit times high. In the **5250** protocol, the signal levels are reversed.

communications protocol—A set of rules which defines the physical, electrical, control, and formatting specifications required to successfully transfer data between two systems.

context switch—Switching between two theoretically independent functions that should not affect each other except under specified circumstances.

controller—The master device that initiates all communication to the slave device and controls the manner in which the slave presents the information. It acts as the interface, both physically and logically, between the slave terminals and printers and a host processor.

CPU-CLK—The clock that the operation of the **BCP**'s CPU is synchronized to. The period of this clock which defines **T-state** boundaries is either that of **OCLK** or one-half of **OCLK** depending on the configuration of the **BCP**. The timer clock is also derived from CPU-CLK.

CUT—Control Unit Terminal. A mode of the **controller** where attached devices have limited intelligence and are perceived to be hardware extensions of the **controller**. The **controller** directs all printer, screen, and keyboard activity.

DFT—Distributed Function Terminal. A **controller** mode that supports multiple logical terminals in the same device. The **controller** communicates in higher level commands via data placed in the buffer. The slave device has a greater amount of intelligence than the **CUT** mode device and is responsible for the terminal operation.

direct coupled—The connection of the **transceiver** to the transmission cable in a manner that does not isolate it from DC voltages. Contrast this with **transformer coupled.**

dual port memory—A memory architecture that allows two different processors to access the same memory range alternately.

ending sequence—A defined sequence of bits signifying the end of a transmission. In **3270** and **3299**, it consists of a **bi-phase** 0 followed by a low to high transition on the bit time boundary and two **mini-code violations.**

FIFO—A section of memory or, as in the case of the **BCP** transceiver, a set of registers that are accessed in a First-In First-Out method. In other words, the first data placed in the FIFO by a write will be the first data removed by a read.

fill bits—Fill bits are **bi-phase** 0's used only in the **5250** protocol. A minimum of three fill bits are required between each frame of a **multi-frame message.** This number may be increased by the controller to approximately 243 per the SetMode command. There are always only three fill bits after the last frame of the transmission.

general purpose 8-bit mode—A generic communications mode similar to **3270** and **5250** frame formatting using 8-bit serial data and **bi-phase** signal encoding. The **BCP** supports both **promiscuous** and **non-promiscuous** modes.

Harvard architecture—A computer architecture where the instruction and data memory are organized into two independent memory banks, each with their own address and data buses.

hold time—The amount of time the line is driven at the end of **5250** transmissions to suppress noise on the cabling system.

ICLK—The clock that identifies the start of each instruction when it rises and indicates when the next instruction address is valid when it falls.

immediate addressing mode—An addressing method where one operand, the data for Move instructions and the address for Jump instructions, is contained in the instruction itself.

immediate-relative addressing mode—An addressing method that adds an unsigned 8-bit immediate number to the index register IZ to form the data memory address of an operand.

indexed addressing mode—An addressing method that uses the contents of an index register as the data memory address for one of the operands in an instruction.

interrupt latency—The time from when an interrupt first occurs until it begins executing at its interrupt vector.

jitter—Timing variations for signals of different harmonic content that move the edges of a transmitted signal in time causing uncertainty in their decoding.

jitter tolerance—The total amount of time an edge of a transmitted bit may move and still have its data bit decoded correctly.

LIFO—A sequence of registers or memory locations that are accessed in a Last-In First-Out method; in other words, the last data written into the LIFO will be the first to be removed by a read. Also known as a **stack.**

limited register set—In the **BCP**, the first 16 register address locations (R0–R11 in both banks and R12–R15) that can be used in all instructions.

6.0 Reference Section (Continued)

line hold—The act of driving the transmission line during **5250** transmissions at the end of a message to allow the receivers to unsync. This insures that the receivers will not see line noise as the start of another frame when the line floats.

line interface—All the circuity between the **BCP** and the communications cable medium.

line reflection—Energy from a transmission that is not absorbed by a load impedance and can cause interference in that signal.

Manchester II encoding—See **bi-phase** encoding.

mask—(1) A mechanism that allows the program to specify whether interrupts will be accepted by the CPU. (2) To disable the accepting of an interrupt by the CPU.

mid-bit—In **bi-phase** encoding, the transition in the center of a bit time.

mini-code violation—A violation of the **bi-phase** encoding format that is part of the **ending sequence** in **3270, 3299,** and the **general purpose 8-bit mode.** The mini-code violation has no **mid-bit** transition being high for the entire bit time. There is no mini-code violation in **5250.**

multidrop—A communication method where all the slave devices are attached to the same cable and respond to **controller** commands and data only when their own address frame precedes the transmitted frame.

multi-frame message—Several bytes of data together in the same uninterrupted message that have only one **start sequence** and one **ending sequence.**

multiplexer—A device that receives **3299** protocol transmissions from a **controller**, strips off the address field, and determines over which of eight ports to transmit the message in **3270** format. The device then directs the response from the terminal back to the **controller.**

non-promiscuous—A receiver mode that only enables a data available interrupt when the address frame of the message matches that previously specified. The **5250** and **general purpose 8-bit modes** of the **BCP** support both **promiscuous** and non-promiscuous modes.

NRZ—Non Return to Zero. A data format that uses a high level to represent a data 1 and a low level to represent a data 0. The signal level does not return to a zero level in each bit time. See also **NRZI.**

NRZI—Non Return to Zero Inverted. A data format similar to **NRZ** but with the signal levels reversed.

OCLK—The external Oscillator CLocK connected to the **BCP**. This frequency, from a crystal or a clock, cannot be changed by the **BCP** itself. **CPU-CLK** is derived from OCLK; in addition, the **transceiver** can be configured so that **TCLK** is derived from OCLK.

parity—A one bit code, usually following data, that makes the total number of 1's in a data word odd or even, including the parity bit itself. It is included as an error checking mechanism.

POLL—A command issued by a **controller** to determine changes in terminal status, such as keyboard activity or keylock.

POLL/ACK (PACK)—A command issued by a **controller** to indicate to the terminal that the controller has recognized the non-zero status response of the terminal to its **POLL**, hence its full name poll/acknowledge.

pop—To remove data from a **stack.**

predistortion—The initial voltage step in a **Manchester encoded** bit used to change frequency components of the signal to limit introducing **jitter.**

promiscuous—A receiver mode that enables a data available interrupt regardless of the contents of the transmission address frame. The **5250** and **general purpose 8-bit** modes of the **BCP** support both promiscuous and **non-promiscuous** modes.

push—To place data onto a **stack.**

quiesce pulse—A **bi-phase** 1 bit that is placed at the beginning of a transmission to charge the cable in preparation for the transmission of data. In addition, the quiesce pulses are used as part of the identifying **start sequence**. Typically, five quiesce pulses are placed there.

register addressing mode—An addressing method that uses only operands contained in registers.

register-relative addressing mode—An instruction addressing mode that adds the unsigned 8-bit value in the current **accumulator** to any one of the index registers forming a data memory address for one of the instruction's operands.

remote access—An access to **dual port memory** by a device other than the **BCP.**

repeater—A device used to extend the communication distance between a **controller** and a slave device by receiving the message and re-transmitting it.

RIAS—The Remote Interface and Arbitration System that allows a remote processor and the **BCP** to share the same memory with arbitration of any conflict while the **BCP** is running. A remote processor may also stop and start the **BCP** as well as read and write the Program Counter.

soft-loadable—A feature of a processor system that allows another processor to provide it with instructions and data.

stack—See **LIFO.**

start sequence—A unique arrangement of bits that begin each transmission to ensure proper frame alignment and synchronization. Each transmission begins with five **bi-phase** encoded 1's **quiesce pulses**, a **code violation**, and the **sync bit** of the first frame.

station address—The identification number of a **5250** terminal or other slave device that will specify which device on a **multidrop** line a message is sent to.

sync bit—A **bi-phase** 1 that is placed as the first bit of a frame.

T-state—The period of **CPU-CLK.**

TCLK—The Transceiver CLocK that runs both the transmitter and receiver at a frequency equal to eight times the required serial data rate. The clock can be obtained from a scaled **OCLK** or from **X-TCLK.**

time-out—An interrupt that occurs when the timer reaches a count of zero.

transceiver—The TRANSmitter used for sending messages and the reCEIVER used for reading messages.

transformer coupled—The isolation of the **transceiver** from the transmission cable through the use of a transformer. Contrast this with **direct coupled.**

trap—A **BCP** instruction that forces a software interrupt.

6.0 Reference Section (Continued)

TT/AR—Transmission Turn-around / Auto Response. An acknowledgement by the terminal or other slave device that a write command has successfully been received or that a **POLL** command status response is all zero.

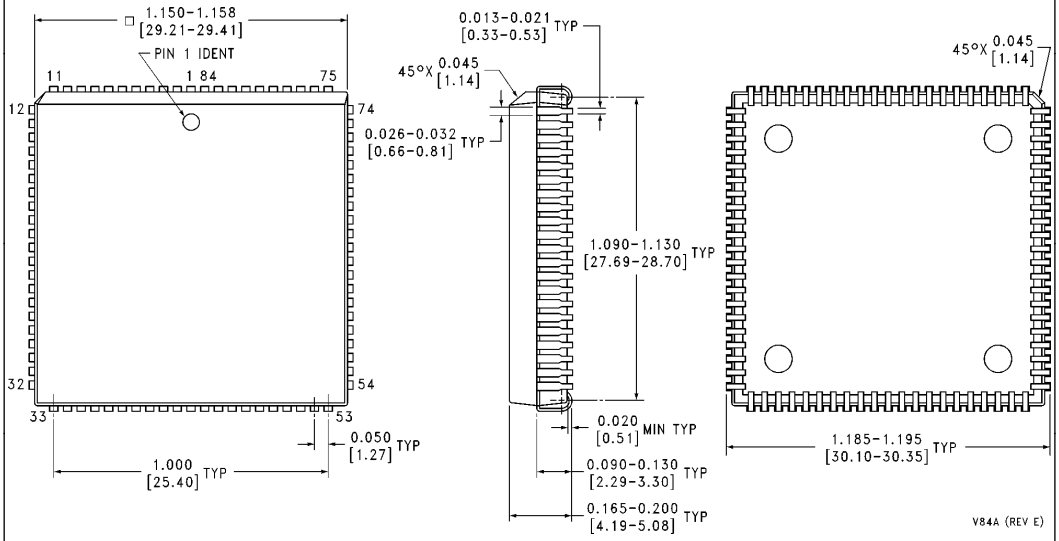
twin-ax—(1) The shielded pair cable that is used in a **5250** communications systems. (2) Sometimes used to refer to the IBM **5250** communications protocol itself.

unmask—Enable the accepting of an interrupt by the CPU.

wait state—Additional **T-states** that may be added to a memory access to increase the time from address generation to the beginning of either a memory read or write. The **BCP** may add as many as seven data wait states and three instruction wait states.

X-TCLK—The eXternal Transceiver CLocK. An independent clock source that the **BCP transceiver** operation may synchronize to rather than from **OCLK.**

Physical Dimensions inches (millimeters)



Plastic Chip Carrier (V)
Order Number DP8344B
NS Package Number V84A

National Semiconductor Corporation
1111 West Bardin Road
Arlington, TX 76017
Tel: 1(800) 272-9959
Fax: 1(800) 737-7018

National Semiconductor Europe
Fax: (+49) 0-180-530 85 86
Email: cnjwge@tevm2.nsc.com
Deutsch Tel: (+49) 0-180-530 85 85
English Tel: (+49) 0-180-532 78 32
Français Tel: (+49) 0-180-532 93 58
Italiano Tel: (+49) 0-180-534 16 80

National Semiconductor Hong Kong Ltd.
13th Floor, Straight Block,
Ocean Centre, 5 Canton Rd.
Tsimshatsui, Kowloon
Hong Kong
Tel: (852) 2737-1600
Fax: (852) 2736-9960

National Semiconductor Japan Ltd.
Tel: 81-043-299-2309
Fax: 81-043-299-2408